Efficient Implementation Strategies for Block Ciphers on ARMv8

Bachelorarbeit

Bastian Engel

February 10, 2023

Abstract

Lorem ipsum dolor [1] sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Declaration

I hereby declare that ...

Contents

Chapter 1

Introduction

1.1 Block ciphers

Securing communication channels between different parties has been a long-term subject of study for cryptographers and engineers which is essential to our modern world to cope with ever-increasing amounts of devices producing and sharing data. The main way to facilitate high-throughput, confidential communications nowadays is through the use of symmetric cryptography in which two parties share a common secret, called a key, which allows them to encrypt, share and subsequently decrypt messages to achieve confidentiality against third parties. Ciphers can be divided into two categories; block ciphers, which always encrypt fixed-sized messages called blocks, and stream ciphers, which continuously provide encryption for an arbitrarily long, constant stream of data.

A block cipher can be defined as a bijection between the input block (the message) and the output block (the ciphertext). For any block cipher with block size n, we denote the key-dependent encryption and decryption functions as $E_K, D_K : \mathbb{F}_2^n \to \mathbb{F}_2^n$. The simplest way to characterize this bijection is through a lookup table which yields the highest possible performance as each block can be encrypted by one simple lookup depending on the key and the message. This is not practical though due to most ciphers working with block and key sizes $n, |K| \geq 64$. For a block cipher with $n = 64, |K| = 128$, a space of $2^{64}2^{128} = 2^{192}$ is necessary. Considering modern consumer hard disks being able to store data in the order of 2^{40}, it is easy to see that a lookup table is wholly impractical. We therefore describe block ciphers al-

gorithmically which opens up possibilities for different tradeoffs and security concerns.

1.1.1 GIFT

`GIFT`[1], first presented in the *CHES 2017* cryptographic hardware and embedded systems conference, is a lightweight block cipher based on a previous design called |PRESENT|, developed in 2007. Its goal is to offer maximum security while being extremely light on resources. Modern battery-powered devices like RFID tags or low-latency operations like on-the-fly disc encryption present strong hardware and power constraints. GIFT aims to be a simple, low-energy cipher suited for these kinds of applications.

`GIFT` comes in two variants; `GIFT-64` working with 64-bit blocks and `GIFT-128` working with 128-bit blocks. In both cases, the key is 128 bits long. The design is a very simple, round-based substitution-permutation network (SPN). One round consists in a sequential application of the confusion layer by means of 4-bit S-boxes and subsequent diffusion through bit permutation. After the bit permutation, a round key is added to the cipher state and the single round is complete. `GIFT-64` uses 28 rounds while `GIFT-128` uses 40 rounds.

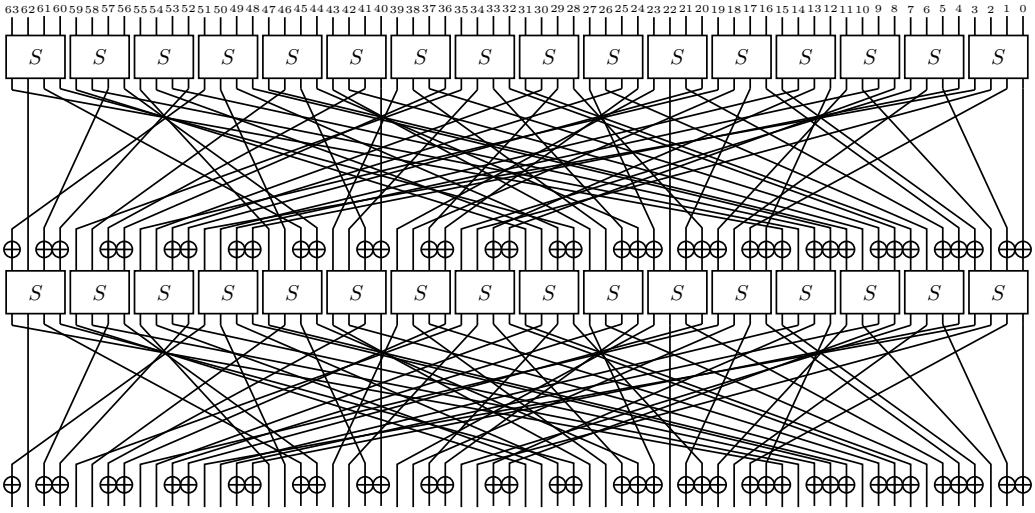

Figure 1.1: Two rounds of GIFT-64

Substitution layer

The input of `GIFT` is split into 4-bit nibbles which are then fed into 16 S-boxes for `GIFT-64` and 32 S-boxes for `GIFT-128`. The S-box $S : \mathbb{F}_2^4 \to \mathbb{F}_2^4$ is defined as follows:

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S(x)$	1	a	4	c	6	f	3	9	2	d	b	7	5	0	8	e

Permutation layer

The permutation P works on individual bits and maps bit b_i to $b_{P(i)}, \forall i \in \{0, 1, \ldots, n-1\}$. The different permutations for `GIFT-64` and `GIFT-128` can be expressed by:

$$P_{64}(i) = 4\left\lfloor \frac{i}{16} \right\rfloor + 16\left(\left(3\left\lfloor \frac{i \bmod 16}{4} \right\rfloor + (i \bmod 4)\right) \bmod 4\right) + (i \bmod 4)$$

$$P_{128}(i) = 4\left\lfloor \frac{i}{16} \right\rfloor + 32\left(\left(3\left\lfloor \frac{i \bmod 16}{4} \right\rfloor + (i \bmod 4)\right) \bmod 4\right) + (i \bmod 4)$$

Round key addition

Round key extraction and key schedule

1.1.2 Camellia

1.2 The ARMv8 platform

With small devices, embedded processors and ASICs becoming ever more ubiquitous and essential in areas like medicine or automotive design, the need for ...

Chapter 2

A simple implementation

Chapter 3

Optimizations through bitslicing

Chapter 4

Leveraging NEON advanced SIMD instructions

Acknowledgements

I want to thank ...

Bibliography

[1] S. Banik, S. Kumar Pandey, T. Peyrin, Y. Sasaki, S. M. Sim, and Y. Todo, "Gift: A small present," pp. 321–345, 08 2017.