An innovative lightweight cryptography system for Internet-of-Things ULP applications

Duy-Hieu Bui

Communauté
UNIVERSITÉ Grenoble Alpes

ĐHQGHN VNU ĐẠI HỌC QUỐC GIA HÀ NỘI Vietnam National University, Hanoi

ĐẠI HỌC CÔNG NGHỆ

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTE UNIVERSITE GRENOBLE ALPES

préparée dans le cadre d'une cotutelle entre *la Communauté Université Grenoble Alpes* et *l'Université Nationale du Vietnam à Hanoi (VNU)*

Spécialité : **Nano Electronique et Nano Technologies**

Arrêté ministériel : le 6 janvier 2005 – 25 mai 2016

Présentée par

Duy-Hieu BUI

Thèse dirigée par **Edith BEIGNE** et **Xuan-Tu TRAN**
Co-encadrée par **Diego PUSCHINI** et **Simone BACLES-MIN**

préparée au sein des **Laboratoires d'Electronique et des Technologies d'Information, CEA Grenoble et Laboratoire du Système Intelligent Intégré (SISLAB), VNU-UET**

dans **les Écoles Doctorales Electronique, Electrotechnique, Automatique et Traitement du Signal (EEATS) et l'Université d'Ingénierie et des Technologies (VNU-UET)**

Système avancé de cryptographie pour l'Internet des objets ultra-basse consommation

Thèse soutenue publiquement le **17/01/2019**,
devant le jury composé de :

M. Régis, LEVEUGLE
Professeur, Université Grenoble Alpes, Grenoble INP, Président
Mme. Nadine, AZÉMARD
Docteure, CNRS, LIRMM, Rapporteuse
M. Jean Max, DUTERTRE
Maître de Conférence, École des Mines de Saint-Etienne, Rapporteur
M. Xuan-Tu, TRAN
Professeur, Université du Vietnam à Hanoi, Directeur de thèse, Membre
Mme. Edith, BEIGNE
Ingénieure de Recherche, CEA Grenoble, Directrice de thèse, Invitée
Mme. Simone, BACLES-MIN
Ingénieure de Recherche, CEA Grenoble, Co-encadrante, Invitée
M. Vincent, BEROULLE
Maître de Conférence, Université Grenoble Alpes, Grenoble INP, Invité

Abstract

The Internet-of-Things (IoT) has come with new standards and new challenges. This has sparked concerns about security constraints, despite its ever-increasing role in business and daily lives. Clearly, IoT contains cloud computing for data processing, communication infrastructure including the Internet, and sensor nodes. During its operations, IoT may collect, transmit and process secret data, which trigger security risks. Nevertheless, implementing security mechanisms for IoT encounters many challenges due to millions of IoT devices integrated at multiple layers, each of which has different computation capabilities and security requirements. In addition to that, sensor nodes in IoT are intended to be battery-based constrained devices with limited power budget, adequate computation, and small memory footprint to reduce costs. This work is therefore motivated to focus on applying data encryption to protect IoT sensor nodes and systems with the consideration of hardware cost, throughput and power/energy consumption. It will firstly implement an ultra-low-power block cipher crypto-accelerator with configurable parameters in $28nm$ FDSOI technology in SNACk testchip with two cryptography modules: AES and PRESENT. AES is a widely used data encryption algorithm for the Internet and currently used for new IoT proposals. PRESENT is a lightweight algorithm which comes up with reduced security level but requires with much smaller hardware area and lower consumption. The AES module is a 32-bit datapath architecture containing multiple optimization strategies supporting multiple security levels from 128-bit keys up to 256-bit keys. The PRESENT module contains a 64-bit round-based architecture to maximize its throughput. The measured results indicate that this crypto-accelerator can provide medium throughput (around $20Mbps$ at $10MHz$), while consumes less than $20\mu W$ at normal condition and sub-pJ of energy per bit. However, the limitation of crypto-accelerator is that the data has to be read into the crypto-accelerator and written back to memory which leads to the increase of the power consumption. Following that, this work looks into an innovative approach to implement the cryptography algorithm which uses the new proposed In-Memory-Computing SRAM. This aims to provide a high level of security with flexibility and configurability to adapt to new standards and to mitigate new attacks. In-Memory Computing SRAM can provide reconfigurable solutions to implement various security primitives by programming the memory's operations. The proposed scheme is to carry out the encryption in the memory using In-Memory-Computing technology. This work evaluates two possible mappings of AES and PRESENT using In-Memory Computing.

Acknowledgment

Undertaking this PhD is a truly challenging and rewarding experience for me and I would like to take this opportunity to thank all the people who in one way or another have contributed to the completion of my work.

First of all, I would like to express my profound gratitude to my supervisors, HDR. Edith Beigné, Research Director, Chief Scientist of IC Design, Architecture and Embedded Software Division (DACLE) at CEA-Leti and Professor Xuan-Tu Tran, Director of the VNU-Key Laboratory for Smart Integrated Systems (SISLAB), at the University of Engineering and Technology (UET), a member university of Vietnam National University, Hanoi (VNU) who have been fantastic mentors to me, continuously supported me with all their patience, motivation, and immense knowledge.

I am also deeply grateful to my advisers, Dr. Diego Puschini and Mrs. Simone Bacles-Min, senior scientists at CEA-Leti. If it had not been for their expertise and constant feedback with insightful comments, valuable suggestions and encouragement, this thesis would not have been achievable.

I would like to thank Professor Regis Leveugle, University of Grenoble Alpes for giving me a great honor by presiding the jury. I would like to express my gratitude to Dr. Nadine Azemard, CNRS, LIRMM and Associate Professor Jean-Max Dutertre, Saint-Etienne School of Mines, for accepting to be reporters. It is my pleasure to have the participation of Associate Professor Vincent Beroulle in the jury.

May I extend my sincere thanks to Dr. HDR. Susanne Lesecq, senior scientist at CEA-Leti for helping me get the French Government scholarship and for her kind support at the beginning of my work in CEA-Leti.

This PhD work was conducted under the collaboration between the VNU University of Engineering and Technology (VNU-UET), and CEA-Leti. I sincerely thank both organizations. My special thanks go to Dr. HDR. Marc Belleville, Former Research Director, Former Chief Scientist of IC Design, Architecture and Embedded Software Division (DACLE) at CEA-Leti and Associate Professor Viet-Ha Nguyen, Rector of VNU-UET for their valuable assistance that made it possible for me to complete this PhD.

I also owe my gratitude to Dr. HDR. Fabien Clermidy and Mr. Jérôme Martin

who gave me their warm-hearted welcome and for providing me a wonderful atmosphere for doing research.

I greatly appreciate the generous support received through the collaborative work with my colleagues at CEA-Leti/DACLE/SCSN/LISAN and SISLAB. Thank you all for sharing with me your excellent advice and for making my PhD fieldwork all the more interesting.

I would never have been able to finish my PhD thesis without the persistent help from my family and my friends. My heartfelt thanks go to my friends who always support me and encourage me with unflagging enthusiasm.

Preface

The exponential growth in Internet-of-Things (IoT) devices has brought the world into a new era of pervasive connectivity where enormous physical things equipped with sensors, actuators, processors, and transceivers communicate and collaborate over the Internet. For this intelligence and interconnection, IoT is creating numerous opportunities to revolutionize the current technologies into smart applications such as smart homes, smart cities, smart grids, and the like which can help unburden human life. On the other hand, IoT is also raising security constraints due to the emergence of new standards and new threats.

IoT has evidently come with new standards, leading to the disparity between the IoT and the existing computer-based systems or embedded systems. Specifically, implementing IoT requires ultra-low-power and ultra-low-cost devices. These are known as highly constrained devices which can operate for a long lifetime using battery or even self-harvested energy. They enable new applications such as implant or wearable devices, environmental monitoring and so forth. Concurrently, IoT has brought into play new challenges and countermeasures, for instance, the cost of devices, the standardization of mechanisms, and the management of millions of lightweight devices, power distribution, security, and privacy. Noticeably, security and low power consumption are important features to be optimized for IoT sensor nodes. Like constrained devices with low resources and limited power supply, IoT sensor nodes need low-power features to lengthen their operations and security mechanisms to protect the secret data of users and their privacy.

In particular, IoT devices and data transmitted through multilayer networks may contain private data or secret data which is threatened by cyber-attacks and organized crimes on the Internet environment. With millions of IoT devices integrated, this can open new attack surfaces which focus on IoT devices in order to use them as new attack tools. Mirai malware which uses vulnerable IoT camera devices is the warning for IoT system. Notwithstanding, the security feature is still left as an option because it increases the power consumption and takes extra time-to-market to IoT products. This leaves rooms for further studies on security to reduce the hardware cost and power consumption.

Security functions are often based on strong cryptography algorithms such as

block ciphers, hash functions, and/or public key cryptography which not only require complicated computations and large power consumption but also reduce the system throughput. Performing these security algorithms on ultra-low-power devices is also a challenging task because these devices are lightweight devices with adequate computation capability, small memory footprint and limited power budget. Therefore, it is critical to optimize cryptographic algorithms in hardware for cost, throughput and especially power and energy consumption. Given these conditions, lightweight security mechanisms such as those based on block ciphers and lightweight block ciphers are considered to be more suitable for constrained devices. To reduce the power consumption, the implementation of security primitives is performed in the crypto-accelerator in hardware. This approach is also applicable to high-performance computing such as Intel CPU or embedded systems such as ARM System-on-Chips. Nonetheless, cost, throughput and power/energy consumption are different features which are hard to achieve at the same time.

The disadvantage of hardware crypto-accelerators is that they have fixed hardware designs, therefore, when there is a flaw in hardware discovered, the only solution is to make a replacement. This urges the research for new approaches for implementing flexible security mechanisms.

Current affordable security implementation for IoT often focuses on application specific integrated circuit (ASIC) with serial processing elements to reduce the hardware cost and power consumption. However, this kind of optimizations, in general, increases the overall latency and reduce energy efficiency. Specific or fixed security functions implemented in ASIC are also the drawbacks in the security point of view because the security standards evolve to adapt to new attacks and to mitigate new threats. ASIC crypto-accelerators are hard to adapt to these changes because of their optimal hardware structure to reduce the hardware cost and power consumption. In addition, to optimize the power consumption, IoT devices often contain a system on chip with multiple hardware modules from different vendors. These hardware modules might include hardware Trojans which might monitor and expose the secret data transported by the system buses. Furthermore, the extra cost is spent to read the data from the memory to the crypto-accelerator and to write the data back to the memory. Therefore, certain trade-offs should be considered to include flexible security solutions for long lifetime constrained devices.

In-Memory Computing is a new advancement of memory technology which can perform logical operations such as AND, OR, NAND, NOR, XOR directly using the memory structure. It is a promising technology to implement different security algorithms because data in the memory can be encrypted in place without being transferred to the processing unit. This minimizes data transfer overhead and the chance of exposing raw data to the system bus. However, because of the serial operations of In-Memory Computing, its flexibility, which can map different algorithms to adapt to new standards or to cope with different attacks, has to be traded off with

the speed and the power/energy consumption of the crypto-accelerators. In-Memory Computing has some important advantages over the traditional methods in terms of security, but it needs in-deep investigation.

To shed some light on the situation, this work firstly focuses on the power/energy consumption optimization of block ciphers which can be used to implement different security primitives to secure the data and communication in IoT systems. The power consumption of both conventional and lightweight algorithms will be carefully evaluated. This work proposes a low-power implementation of two standardized algorithms which can be used for ultra-low-power IoT devices. The first one, Advanced Encryption Standard (AES), is a conventional algorithm widely used to secure the Internet applications with high levels of security. The other is PRESENT, a new lightweight algorithm which uses hardware constructs to reduce the hardware area and the power consumption. Implementation results show that the lightweight implementation can provide lower power consumption than the traditional one but with the sacrifice of security levels. On the other hand, IoT devices might have different security requirements depending on the applications and the available power/energy budget. Therefore, in the next step, this work combines the two modules, AES and PRESENT, into a crypto-accelerator. IoT applications can choose the high secure algorithm with more power consumption or in the critical condition the lightweight one to lengthen its battery life. This crypto-accelerator has been fabricated using 28nm FD-SOI technology in SNACk testchip. The results indicate that this crypto-accelerator can provide sub-pJ/bit operations.

Not only optimizing the power consumption, but the security evaluation using the current state-of-the-art methods are also applied to the proposed design. Two notable evaluation methods are employed in this work including Correlation Power Analysis (CPA) and Test Vector Leakage Assessment (TVLA). TVLA can address different information leakages in the proposed designs while CPA can be used to mount the key recovery attack. The evaluation results using the estimated power traces of the post-signoff netlist show that the proposed design with optimization for low power consumption achieved equivalent information leakage in comparison with the reference design on OpenCores.

Last but not least, from the lesson of various security breaches because of the fixed hardware security module, this work finally explores the configurability, the flexibility and the feasibility of different block cipher algorithms using In-Memory Computing. This work proposes the implementation of two algorithms, which are previously designed in the aforementioned crypto-accelerator, using the In-Memory Computing technology. The implementation results using the behavior model of In-Memory Computing show that the conventional algorithm, AES, with byte level transformation has eight times higher throughput than the bit-level permutation, PRESENT. In addition, because of the serial operations, the security functions using this technology have to trade off flexibility and configurability with throughput and

power consumption. The implementation of AES and PRESENT using In-Memory Computing is at least five times slower than the one in the crypto-accelerator. Security functions in memory can also be equipped with the countermeasures for software implementations, but it needs careful investigations.

Motivations and objectives

Many researches have been focusing on reducing the area and power consumption of cryptographic hardware primitives so that they can be used for constrained devices. As a result, a new class of cryptography algorithms named Lightweight Cryptography has emerged to fulfill the new requirements. Lightweight cryptography considers the trade-offs among area, throughput, power consumption and security features by reducing the security levels to achieve small hardware footprint and low power consumption. In general, the lightweight approach is expected to reduce the data block size and the key size to minimize the hardware area and also the power consumption but this leads to the reduction in security level. Regardless of new advancements in cryptography, many new IoT proposals have chosen AES as the main primitive. Therefore, the optimization for AES is considered to be critical not only for IoT applications but also for other products which use AES.

In addition, lightweight cryptography which uses the hardware-friendly construct to reduce the hardware cost and power consumption is a promising candidate for ultra-low-power and ultra-low-cost applications even though they might reduce the level of security. However, lightweight cryptography has not been selected for recent IoT proposals yet.

On the other hand, depending on the applications' profile, ultra-low-power devices might adapt to different security levels by selecting between conventional security algorithms such as AES or the lightweight ones, for instance, PRESENT. Configurable implementations will increase flexibility and configurability.

Furthermore, hardware implementations of security functions for ultra-low-cost and ultra-low-power devices often use fixed architecture with fixed algorithms which cannot adapt to new standards and against new attacks. With new advancement of the memory technologies, In-Memory Computing which can be programmed to execute logical operation directly in the memory banks might open a new solution to address this problem. The memory can be programmed to execute the security primitives in place without data transfer through the system bus. In-Memory Computing can be an innovative solution for security, but it needs careful investigation and evaluations.

The objective of this PhD's work is firstly to investigate different security mechanisms, which are suitable for highly-constrained IoT devices, to search for a good trade-off among security level, hardware cost and power/energy consumption. After that, two algorithms including a highly-secure algorithm – AES and a lightweight al-

gorithm – PRESENT were chosen to be implemented into a crypto-accelerator with multiple levels of security. An optimization strategy for AES and PRESENT was proposed to reduce its power consumption. Secondly, the implementation results of AES and PRESENT are analyzed to extract the power traces for security evaluation. Finally, in the search for an innovative approach to design security functions for IoT, the design of two algorithms using In-Memory Computing is implemented and evaluated. Security mechanisms implemented using In-Memory Computing are more flexible in comparison with the hardware accelerator because they are reconfigurable and can be used to map different algorithms to accelerate the computation directly in the memory array. In-Memory Computing provides flexibility and configurability to adapt to future IoT standards and to cope with new attacks, but it has to trade the flexibility and configurability for higher power consumption and lower throughput than the proposed crypto-accelerator.

Explanation for the different CMOS technologies used in this thesis

This thesis' work has been conducted under the collaboration among VNU University of Engineering and Technology (VNU-UET) – a member university of Vietnam National University, University of Grenoble-Alpes and CEA-Leti/DACLE in France. It involved the participation of two laboratories including the Key Laboratory for Smart Integrated Systems (SISLAB), VNU-UET, and Digital Architecture (LISAN), CEA-Leti/DACLE, France. The two-thirds period of the work was performed in LISAN which has access to $28nm$ FD-SOI technology and TSMC $65nm$. At the beginning of the work, the designs are experimented using TSMC $65nm$ to prepare the simulation and evaluation environment in the local computer in the lab. Some experiments require an extremely large amount of disk space which can only be performed using the local computer. However, the fabricated demonstration was implemented using ST $28nm$ FD-SOI technology. All the measured results are originated from this technology. The remaining of the research including Chapter 4 was completed in SISLAB which does not have access to the technologies as in CEA-Leti. Therefore, the experiments performed in SISLAB use $45nm$ technology from North Carolina State University (NCSU) education platform development kit [FreePDK45nm]. The standard cells are from Nangate [Nangate2011OCL]. The SRAM cells and the memory peripheral were built based on the OpenRAM compiler [Guthaus2016oao].

Contributions of the work

- **Design of a low-power low-cost crypto-accelerator which can be used for ultra-low-power IoT applications**

The current state-of-the-art of cryptography was studied in terms of hardware cost, power/energy consumption, throughput and security features. After that, two block ciphers are selected for implementation including a traditional algorithm – AES, and a lightweight algorithm – PRESENT. An optimization strategy to optimize AES hardware module was proposed based on 32-bit datapath architecture. This crypto-accelerator was integrated as an IP into the SNACk testchip. It was fabricated in $28nm$ FD-SOI technology. The measured power consumption results show that it can achieve very low power consumption down to $20\mu W$ for AES at $10MHz$ and throughput of $28Mbps$, and $10\mu W$ for PRESENT with the throughput of $17Mbps$ at the same operating frequency.

- **Perform security evaluation of AES crypto-core in SNACk using the post-signoff power estimation traces**

  One of the weaknesses in hardware crypto-accelerators is the hardware security attack such as power analysis attacks. Designing countermeasure is not a part of this work because this work mainly focuses on optimizing the power consumption along with the considerations of hardware costs, throughput and security level. Therefore, this work does not include countermeasures. However, the security evaluation framework using Correlation Power Analysis and Test Vector Leakage Assessment is proposed for early design testing based on the post-signoff power estimation to verify that no security weakness was introduced during the optimization. The framework has successfully revealed the secret key of the reference design on Opencores and the design in this work. The design in this work with power optimization has a certain level of resistance to these types of attacks. It means that the proposed optimization did not add security leakage.

- **Design and implementation of AES and PRESENT in an innovative In-Memory Computing architecture**

  Crypto-accelerators have low cost, low power consumption, however, they are fixed after the chip fabrication. In contrast, the attack methods are changing very fast, thus, the flaws are discovered, these hardware structures must be replaced to maintain the security feature. In this work, a configurable crypto-accelerator is proposed using the In-Memory Computing architecture. This work uses the memory bitcells proposed by Akyel *el al.* [Akyel2016ddr] to map AES and PRESENT on this platform. For a 32-bit datapath architecture, AES takes 232 clock cycles to finish one encryption while PRESENT needs 873 clock cycles. Because of the bit based permutation, PRESENT implementation using In-Memory computing are eight times slower than AES.

Organization of the manuscript

The rests of the manuscript are organized into four chapters. Chapter 1 introduces the Internet-of-things (IoT) and the requirements of ultra-low-power IoT applications. It also analyzes the importance of security in the IoT systems and the security challenges in IoT. The IoT applications are expected to be highly-constrained devices to reduce the cost with very low power consumption. Therefore, implementing security into these devices to reduce cost and power consumption is seen as a challenging task.

Chapter 2 summarizes the security features in IoT and reviews the current state of the art of security for constrained devices. It also presents the reason why block ciphers including AES and PRESENT are selected to be implemented for IoT. The chapter provides a scanning of the current hardware implementations of AES and lightweight cryptography. It additionally looks into the hardware security problems. Moreover, this chapter has two sections on the implementations of AES which use the SRAM memory and the configurable cryptographic modules, which are directly related to the manuscript's proposals in Chapter 3 and Chapter 4.

Chapter 3 presents a proposal of a low-power crypto-accelerator containing two block ciphers – AES and PRESENT. The proposal firstly involves the implementation of SNACk testchip in 28nm FD-SOI from STMicrolectronics. Furthermore, the chapter presents these implementation results including the power estimation and the measured results of SNACk. Finally, the security evaluation of AES in SNACk test chip is also presented and evaluated in this chapter.

Chapter 4 proposes an innovative method to map block ciphers into hardware using the In-Memory Computing mechanisms. It begins with the demonstration of the memory constructions to support In-Memory Computing. It then continues with the mapping of the block ciphers including AES and PRESENT using the memory construction which is previously described.

Finally, the author concludes the contributions of the study in this manuscript. Additionally, it raises some perspectives on which further researches would be beneficial.

List of Publications

This is the list of my publications during my thesis:

[1] Marc Belleville, Anca Molnos, Gilles Sicard, Jean-Frédéric Christmann, Dominique Morche, Duy-Hieu Bui, Diego Puschini, Suzanne Lesecq, and Edith Beigné. Adaptive Architectures, Circuits and Technology Solutions for Future IoT Systems. *J. Low Power Electronics*, 13(3):298–309, 2017.

[2] D. Bui, D. Puschini, S. Bacles-Min, E. Beigné, and X. Tran. Ultra low-power and low-energy 32-bit datapath AES architecture for IoT applications. In *2016 International Conference on IC Design and Technology (ICICDT)*, pages 1–4, June 2016.

[3] D. Bui, D. Puschini, S. Bacles-Min, E. Beigné, and X. Tran. AES Datapath Optimization Strategies for Low-Power Low-Energy Multisecurity-Level Internet-of-Things Applications. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(12):3281–3290, Dec 2017.

Contents

Chapter 1

Ultra-Low-Power applications for IoT and security problems

Security has arisen as one of the key areas that developers of Internet-of-Things applications are trying to make it better. On the one hand, the advent of IoT has dramatically enabled miniaturized computing platforms such as implant or wearable devices, health monitoring, environmental monitoring and the likes due to its characteristics as constrained devices which can operate for a long lifetime using batteries or even self-harvested energy. On the other hand, the computing capability of these devices in a pervasively connected environment has also considerably raised the concerns about the misuse of collecting, processing and exchanging of user data. More precisely, the demand for information security and privacy protection comes integrally to the daily operations and the integration over the Internet of these new classes of IoT backed applications. Meanwhile, IoT differs from the existing computer-based systems or embedded systems in terms of both connectivity and the fact that IoT needs to be implemented in ultra-low-power and ultra-low-cost devices. These devices are lightweight devices with limited processing power, small memory footprint and low power budget. On the contrary, strong cryptography algorithms such as block ciphers, hash functions, and/or public key cryptography are usually employed for security functions. However, these firm cryptography algorithms require complicated computations and high power consumption which diminish the system throughput. Consequently, performing cryptography algorithms on ultra-low-power devices is set to be an extremely challenging task.

In addition, IoT ecosystems are evolving with new standards, new security threats and attacks discovered. Therefore, flexible and configurable solutions for IoT security are required to adapt to these factors. Software implementations of security functions have more flexibility than hardware implementations because new standards and bug fixes can be done by updating the software. However, software implementation reduces the overall throughput of the system and cause higher power consumption.

Current affordable security solutions are focusing on implementing security functions as hardware accelerator with serial processing to reduce hardware cost and optimize power consumption. However, they are specific hardware with fixed functionalities which do not provide the flexibility and configurability.

With the new advancement of memory technologies, In-Memory Computing is a new technology which can perform logical and arithmetic computations in place of the memory. It can be used to implement different algorithms including security primitives. In-Memory Computing can provide more flexible solutions than specific hardware accelerator with low power consumption. However, because of the serial operations of the memory, the flexibility has been traded off with the throughput and power consumption. This new mechanism needs in-deep analysis to prove its advantages.

The purpose of this chapter is to firstly review various aspects of IoT systems which lead to the requirement of ultra-low-power features. After that, it analyzes the security problems in IoT and possible security mechanisms for IoT. Consequently, it explores the possible solutions for ultra-low-power devices which are based on the block ciphers and lightweight blocks ciphers. Furthermore, it also addresses the security challenges in IoT in terms of attack surfaces and standardization which leads to the requirements of a flexible and configurable design for security. These ideas are further developed with the proposals of this work in the following chapters.

This chapter begins with Section 1.1 reviewing the organization of IoT systems and addressing the security problems and application requirements for IoT. It further raises the importance of ultra-low-power features for IoT sensor nodes. Section 1.2 embodies various security mechanisms applicable to IoT sensor nodes. Then, it raises the new constraints of IoT applications which can be provided by lightweight cryptography especially in terms of hardware cost and power consumption. This section additionally compares the complexity of different security primitives which can be employed for IoT. In Section 1.3, different challenges regarding security for IoT are discussed. In return, it also analyzes strong security mechanisms which can be applied to IoT in the future if there is a practical implementation. The chapter ends with some conclusions and perspectives presented in Section 1.4.

1.1 Introduction to Internet-of-Things applications and requirements for Ultra-Low-Power features

The Internet-of-Things (IoT) is a new concept of connected objects through the Internet. An object or a thing is a physical thing or a device which has the capabilities of computation and communication through the global network – the Internet. Because of the Internet connectivity and computation capability, a thing can capture data, preprocess them, and then send the results to the cloud through the Internet. In the cloud, the data can be further processed by artificial intelligence (AI) applications.

A thing can also receive data from the Internet to control its actuators to adapt to the changes in its working environment. Figure 1.1 shows a general organization of IoT systems. The IoT sensor/actuator node is often a wireless-based low-power and low-cost system using battery or power-harvesting power supply.

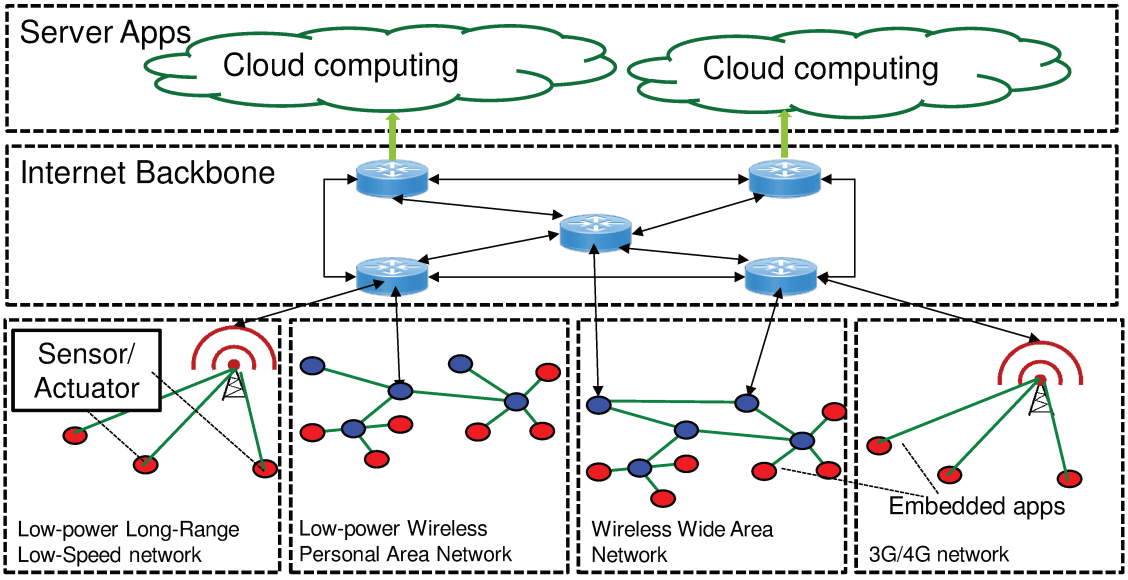


Figure 1.1: The general organization of IoT.

The idea of IoT is getting more and more attention because of the immense benefits it brings to each individual and the society at large [Atzori2010tio]. First of all, IoT opens new ways of collecting and manipulating data from physical objects, which is expected to revolutionize the existing technologies into the smart things such as smart homes, smart transports, smart cities and so on. Data collected by physical objects are stored on the Internet, to be more specific in the cloud; therefore, they can be accessed at any time and from anywhere in the world with an Internet connection. Furthermore, various applications are enabled to access these data to extract useful information hence improve management and usage efficiencies. This leads to many smart applications which can adapt to users' needs. In addition, to each individual, IoT provides new living experiences of smart things – things with computation and communication capabilities. Take a smart refrigerator for example. A smart refrigerator can collect data about its users' consumption behavior for food management purposes such as creating shopping lists, making orders from online marketplaces and optimizing the utilization. This helps save enormous cost, hence possibly affect the economy. There are numerous other examples in which IoT helps improve our lives. Finally, IoT can help better the society management by bringing new living concepts which are not limited to smart cities, smart transportation, or smart grid. Figure 1.2 published by McKinsey Global Institute in June 2015 describes various business potentials of IoT in different settings.

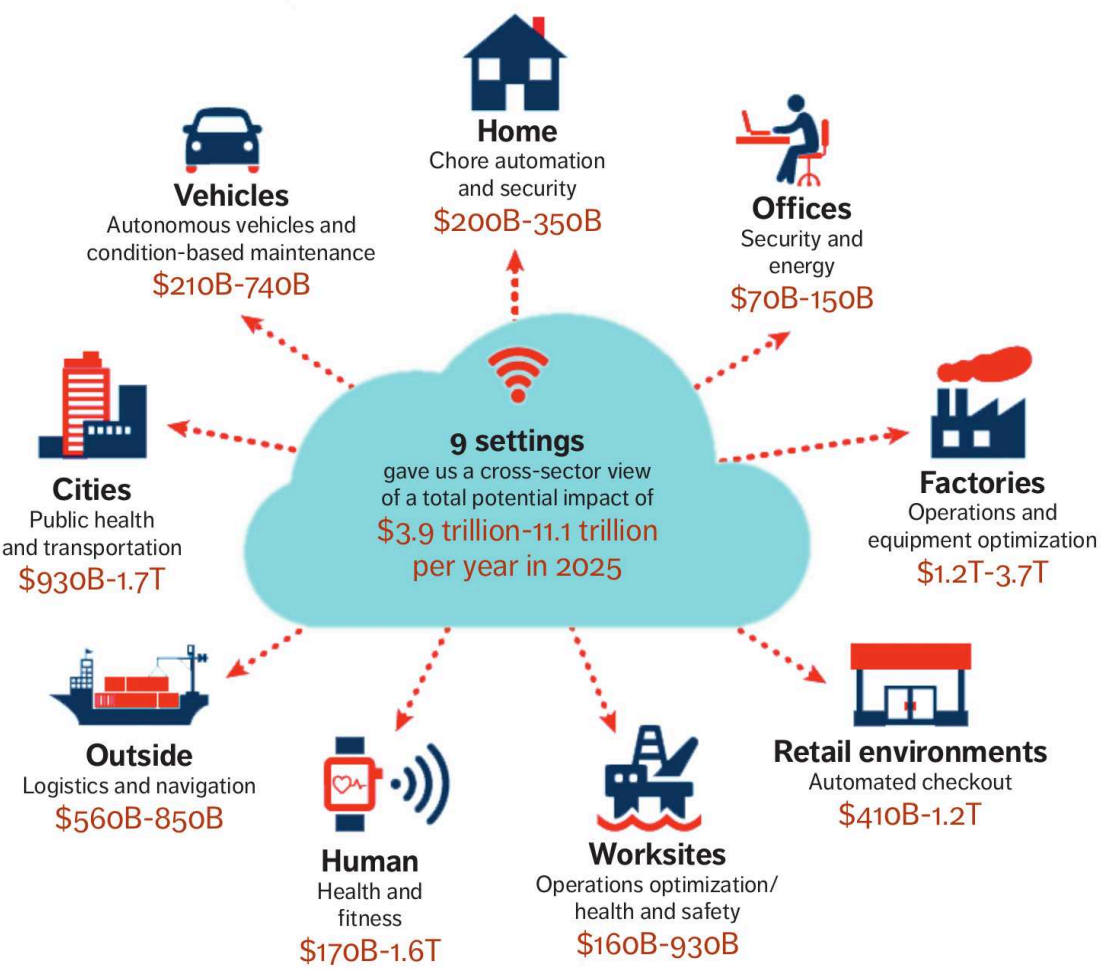


Figure 1.2: IoT applications and their expected market share (*Source: McKinsey Global Institute (June 2015)*).

The success of IoT is based on many enabling technologies. As indicated by *Al-Fuqaha et al.* in [Al-Fuqaha2015iot], IoT is the combination of different technologies including identification technology, sensing technology, communication technology, computation technology and the software parts such as services and semantics. Identification technology is used to address and name the object in the network so that each object can be referred to using a unique identifier. Sensing technology provides the ability to capture the changes in the environment surrounding objects. Two most important technologies enabling IoT are communication technology and computation technology. Communication technology enables the seamless integration of many IoT devices in a small area using wireless technology such as Wi-Fi, Bluetooth BLE, IEEE 802.11.4, Z-wave, Sigfox and LoRaWan, and so forth. The computations in IoT Smart Objects rely on processing units such as an embedded processor or a System-on-Chip (SoC) which does fast processing of the data. Many computations will be done in the cloud which needs supercomputers with acceleration to

do complicated machine learning algorithms or big data applications. Based on the computation capability, different services which will provide data access for different applications. Finally, to provide data for them, the semantics of data should be defined to provide interoperability among applications.

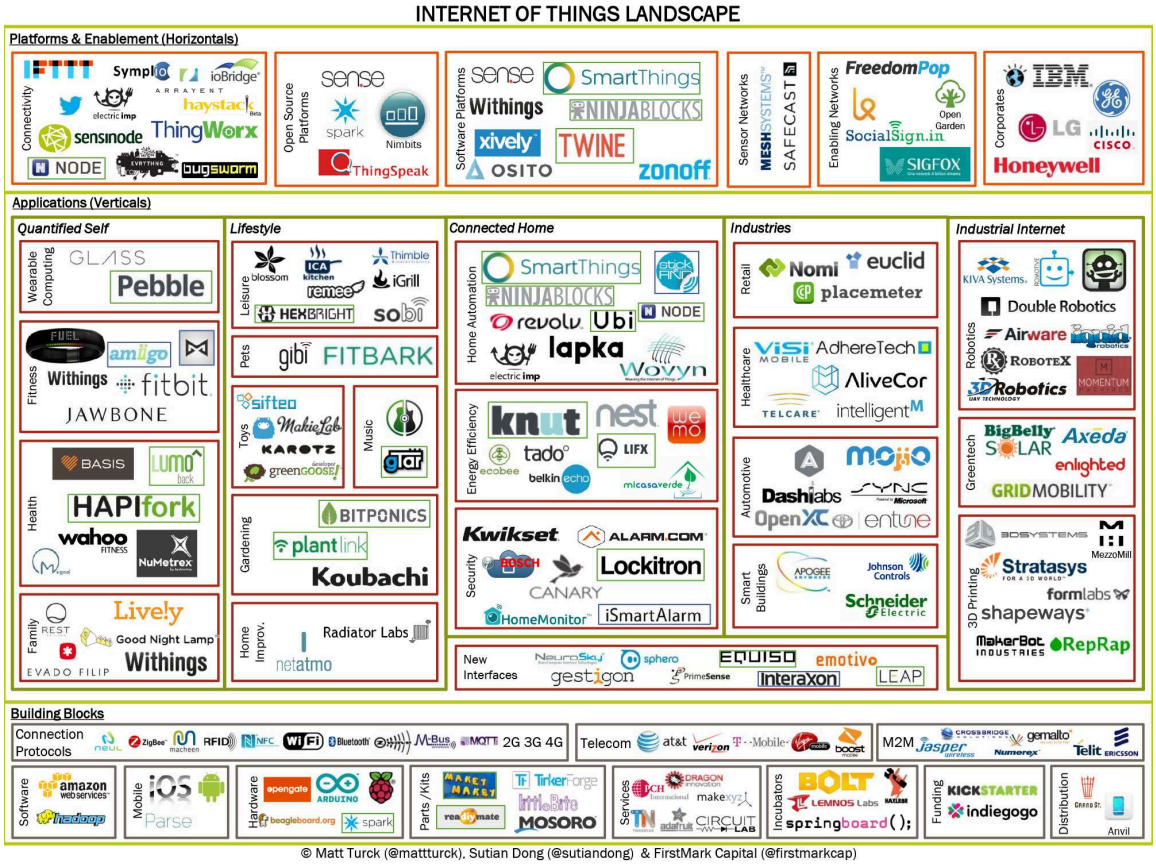


Figure 1.3: Internet-of-Things landscape.

IoT has opened new areas for application development. Figure 1.3 shows an example of different applications, platforms and enabling technologies for IoT. For personal use, IoT can be used in smart homes or personal things such as smart watches or e-health devices. For industry, IoT brings better monitoring and logistic capability which will boost the performance and the efficiency in factories. For our society, IoT with smart cities, smart grid and so on brings a new level of efficiency and can help to build a sustainable environment. In summary, IoT and Artificial Intelligence (AI) create a new world of smart things and objects which is expected to be sold in large quantities in the coming years.

IoT is creating a world of limitless potential, but it is not without challenges. Standardization appears to be a critical issue when it comes to adoption of IoT in different industries such as industrial IoT, agriculture IoT or IoT for healthcare, i.e. which among various enabling technology for IoT to be applied in such fields. This leads to difficulties in developing interoperable applications. In addition to that,

different applications and services might have different requirements and interfaces; making it strenuous to create an interoperable system. What is more, with the integration of thousands of devices, the bandwidth will become an evident bottleneck for IoT. This leads to the development of Fog/Edge computing which will implement the preprocessing tasks at the edge devices. The implementation of heavy computation at the edge or device levels will trigger a problem for power consumption. IoT devices supposed to be low-cost devices with the battery-based power supply or even power harvesting. Therefore, the optimization for low power of IoT systems will be important for the future IoT device. Finally, IoT is posing serious issues of security and privacy as highlighted by the current surveys on IoT such as [Atzori2010tio], [Yang2017aso], [Lin2017aso], [Al-Fuqaha2015iot]. The Internet is exposed to many critical bugs, even with the state-of-the-art security systems. IoT is built on the Internet infrastructure, hence, inherits all the implicit issues of the Internet including security challenges. Furthermore, many IoT devices are supposed to be constrained low-cost devices with adequate processing power, small memory footprint, and even limited power/energy budget; for example, power-harvesting devices and battery-based devices. As a result, dealing with security problems for these constraint devices is even more challenging.

In the light of these conditions, this study focuses on power/energy optimization for security components for IoT devices/sensor nodes. They are low-cost constrained devices with small memory footprint with reduced computation capabilities to minimize the cost and power consumption. Figure 1.4 shows the energy per bit of the most important components of such devices including communication, computation and security. It is clear from Figure 1.4 that security is an additional feature, but it consumes a large amount of energy. Therefore, optimization for power and energy consumption of IoT devices is becoming vital for future IoT applications. It has been shown that for constrained IoT devices, security functions on pure software are not efficient in terms of throughput and power consumption [Zhang2018rar]. Data encryption/decryption using software not only reduces the throughput of the application but also occupies the CPU runtime. This leads to an increase in power consumption and energy consumption of devices. Eventually, most of the works on the optimization of security primitives for low power have been focusing on hardware architectures. In addition, power consumption optimization for security algorithms on hardware is a challenging problem to such extent that the security algorithms especially the encryption and decryption are pseudorandom algorithms. The randomness makes it hard to define a clear strategy for optimizing power consumption. Furthermore, optimization for power consumption might cause other security issues, for example, the leakage of the credential information through the implementation. Side-channel attack techniques such as power analysis attacks [Kocher1999dpa] use power consumption as an attack vector. Consequently, power consumption optimization should consider this issue. This work also seeks for a new method to increase

the security feature and to reduce the power consumption of the system. Part of the work experiments with a new type of memory which is capable of performing logical operations. This can be a new approach to implement security primitives in the future.

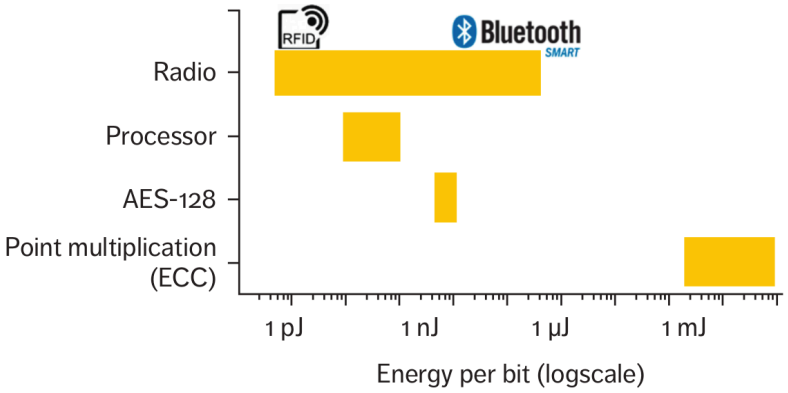


Figure 1.4: Energy per bit of different components in IoT (*Source: [Yang2017hdf]*).

In summary, IoT brings countless benefits not only to individual and industrial uses but also for the society at large. On the other hand, it also presents unique challenges, among which the ultra-low power capability and security will be two areas of concern in this work. Ultra-low power capability will help to deploy IoT applications on a larger scale with battery-based and power-harvesting devices while security will ensure the critical information not to be exposed to attackers. It should also be emphasized that for many applications, security is an important additional feature, but it requires a large amount of power consumption. In addition, IoT standards and proposals have to evolve to adapt to new security requirements and to cope with new threats. Therefore, a flexible and configurable security solution should be taken into account. For this reason, the security functions for ultra-low power IoT applications with flexibility and configurability are worth considering and will be reviewed in the next section.

1.2 Security mechanisms and lightweight cryptography

Attacks on IoT enabled devices and recent hardware attacks lately have brought into play the concerns about the security of IoT applications. As proof, Mirai malware in [Sinanovic2017aom] has affected numerous weak-password IoT cameras. They are later used as a botnet for Distributed Denial of Service (DDoS) attacks on Domain Name Server (DNS) provider Dyn. Mirai malware reveals that weak-security IoT devices can be used not only to collect data but also to mount attacks on another system. Mirai malware also demonstrates new attack vectors. Instead of focusing on

the computers or their users, in the IoT era, hackers might focus on the devices and use them as a mean to attack other parts in the IoT system. In addition, in terms of hardware security, the recent findings of Spectre [Kocher2018sae] and Meltdown [Lipp2018m], which exploited the security hole in the current computer architectures and affected billions of computers and devices, again highlighted the importance of designs for security. A security hole in hardware architectures will create a larger impact because hardware components especially integrated IP cores in systems-on-chip are hard to fix and replace.

A strong-security computer-based system might require security features, for including confidentiality, integrity, availability, accountability, auditability, trustworthiness, non-repudiation and privacy as described in Table 1.1. Confidentiality ensures the secrecy of the data while integrity enforces the non-modification of the data. Availability ensures that the system and service are available when requested by an authorized user. Accountability addresses the users' responsibility for their actions, whereas auditability provides the capability to persistently monitor all actions. Trustworthiness facilitates an ability to verify, identify and establish trust in a third party's environment. Last but not least, privacy puts the system in place to comply with the privacy policies, at the same time, enables individuals to control their personal information. Fulfilling all these security requirements has already been a great challenge, even for a computer-based system.

For computer-based systems and their interconnected networks, the above security requirement can be ensured by using the standardized security mechanisms and recommendations as suggested by the Internet Engineering Task Force (IETF) [Bellovin2003smi] as below.

1. One-Time Password for authentication and identification.
2. HMAC – a preferred shared-secret authentication technique.
3. IPsec – a generic IP-layer encryption and authentication protocol.
4. TLS – an encrypted, authenticated channel that runs on top of TCP.
5. SASL – a framework for negotiating an authentication and encryption mechanism to be used over a TCP stream.
6. GSS-API – a framework for applications to use when they require authentication, integrity, and/or confidentiality.
7. DNSSEC – digitally signs DNS records.
8. Digital Signatures – one of the strongest forms of challenge/response authentication using public key cryptography.
9. OpenPGP and S/MIME – two different secure mail protocols.

10. Firewalls and Topology: Firewalls are a topological defense mechanism.

11. Kerberos – a mechanism for two entities to authenticate each other and exchange keying material.

12. SSH – a secure connection between client and server.

Although the aforementioned mechanisms are widely used in the current Internet applications, they require complicated software and hardware implementations which are not suitable for constrained IoT devices. Constrained IoT devices may provide only adequate supports for requirements such as confidentiality, integrity, identification and authentication. Other requirements could make constrained IoT devices more usable, but they are not mandatory.

Table 1.1: Common security requirements for Internet-based System

Requirement	**Definition**
Confidentiality	Ensuring that only authorized users access the information
Integrity	Ensuring that data is not modified
Availability	Ensuring that all system services are available when requested by an authorized user
Accountability	An ability of a system to hold users' responsibility for their actions
Auditability	An ability of a system to persistently monitor all actions
Trustworthiness	An ability of a system to verify identify and establish trust in a third party
Non-repudiation	An ability of a system to confirm occurrence/non-occurrence of an action
Privacy	Ensuring that the system obeys privacy policies and enabling individuals to control their personal information

Constrained IoT devices need special consideration for security not only because of the cost but also the power/energy consumption as well as the time to market it will incur. Instead of meeting all the security requirements as described in Table 1.1, they might have only a subset of lightweight security mechanisms which provides trade-offs among cost, power consumption and energy consumption. The lightweight security requirements and the associated security tools for IoT are described in Table 1.2. IoT devices might require the authentication and identification of the IoT node

so that only authenticated devices can join the network. IoT nodes may contain secret data; therefore, they need data protection to protect their data from unauthorized access. IoT data will be analyzed to make a decision to react to the changes; therefore, they should not be modified. This feature is protected by data and device integrity. Furthermore, IoT devices may contain wireless communication, which can be easily captured through the air by the attackers. This brings on the importance of protecting the communication channel so that the data will remain secret. In addition, the software and firmware running on IoT devices should be protected from reverse engineering and unintended modification. Finally, the availability of IoT devices should be ensured so that they can run endlessly. These requirements can be met by using current security tools such as data encryption, digital signatures, cryptographic hash functions, and other security protocols. In addition, they are built based on strong secure primitives which are cryptography functions including symmetric cryptography, asymmetric cryptography, cryptographic hash functions and so son.

Table 1.2: Security requirements for constrained IoT devices

Features	**Description**	**Security tools**
Authentication and identification of the IoT nodes	Ensuring the IoT node is not the unwanted one	Digital Signatures Cryptographic Hash Function Physical Unclonable Function
Data protection	Protecting the IoT data from unauthorized parties	Data Encryption
Data/device integrity	Protecting the IoT data and devices from unwanted modification	Cryptographic Hash Function
Communication protection	Creating a secure communication channel among IoT devices/sensor networks	Secure protocols: TLS/SSL DTLS IPSec Data encryption
Firmware and/or Software protection	Do not allow unauthorized modification of the firmware or software	Firmware/software encryption Digital signature
Device/data availability	Preventing the IoT devices from DoS attacks	DoS detection and prevention

Currently, available lightweight cryptographic primitives can be considered solutions for existing security problems on IoT constrained devices. As a matter of

fact, devices and protocols with proper usage of identification, authentication and data encryption will reduce the risk of exposing secret or personal data to attackers. These cryptographic primitives contain two main categories: asymmetric cryptography (or public-key cryptography) and symmetric cryptography. Nonetheless, current asymmetric cryptography has not been yet suitable for constrained devices such as constrained IoT devices because it uses computationally intensive algorithms such as RSA [Rivest1978amf] or Elliptic Curve Cryptography (ECC) [Koblitz1978ecc]; and consequently, it needs complex calculations, large memories to store the keys, and high power consumption. Table 1.3 illustrates the security levels of symmetric cryptography and its equivalent asymmetric cryptography. As obviously seen, asymmetric cryptography requires not only more processing power but also more memory and storage than symmetric cryptography. In addition, IoT applications require multiple security mechanisms such as identification, authentication and data encryption. Asymmetric cryptography is more flexible in the application point of view, but it takes more processing power, more data storage and much more power consumption even when the cryptography modules are implemented in hardware.

Table 1.3: Security level recommended by ECRYPT-II [ECRYPT-II]

Level	Protection	Symmetric cryptography		Asymmetric cryptography			
		Symmetric cipher (key size in bits)	Hash Functions (key size in bits)	Factor Modulus (bit)	Discrete Logarithm		Elliptic curve (key size in bits)
					Key (bits)	Group (bits)	
1	Attacks in "real-time" by individuals; only acceptable for authentication tag size	32	-	-	-	-	-
2	Very short-term protection against small organizations; should not be used for confidentiality in new systems	64	128	816	128	816	128
3	Short-term protection against medium organizations, medium-term protection against small organizations	72	144	1008	144	1008	144
4	Very short-term protection against agencies, long-term protection against small organizations	80	160	1248	160	1248	160
5	Legacy standard level	96	192	1776	192	1776	192
6	Medium-term protection	112	224	2432	224	2432	224
7	Long-term protection	128	256	3248	256	3248	256
8	"Foreseeable future"	256	512	15424	512	15424	512

The asymmetric cryptography system or the public-key cryptography system uses a pair of keys as described in Figure 1.5(b): a public key for encrypting the plaintext and a private key to decrypt the ciphertext. In contrast, the symmetric cryptography system uses only one key on both sides as presented in Figure 1.5(a). This shared

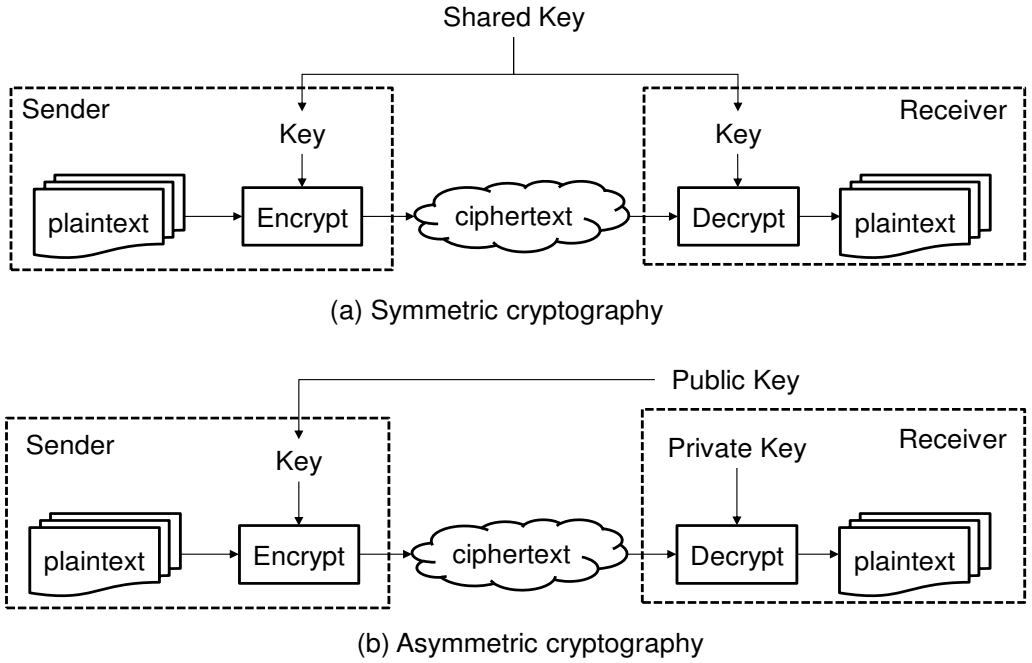


Figure 1.5: Symmetric cryptography scheme and asymmetric cryptography scheme.

key must be known in advance to do the communication. The public key can be distributed to everyone while the private key is known to only the owner. In a secure system using asymmetric cryptography, the sender will use the public key to encrypt the message, while only the owner can decrypt the encrypted one with his private key. Asymmetric cryptography is often based on hard problems such as factorization of big prime numbers or the elliptic curves. However, to keep the system secure, asymmetric cryptography often uses big numbers with the size of thousands of bits. Asymmetric cryptography is more flexible because anyone taking part in the communication can have the public key of the receiver however only the receiver can decrypt the message. By doing this, the sender and the receiver can verify the identity of each other. Consequently, asymmetric cryptography is widely used for key distribution and digital signature. It is also used for high-level security protocols such as Transport Layer Security (TLS), Datagram Transport Layer Security (DTLS), Secure Sockets Layer (SSL), and Pretty Good Privacy (PGP) and so on.

In a practical system, both asymmetric cryptography and symmetric cryptography are used. Asymmetric cryptography provides high-level protocols such as key exchange, key distribution and key update while symmetric cryptography is used as the main data encryption methods. The drawback of the systems which use only symmetric cryptography is that the shared secret key must be known in advance. After that, application key exchange, key distribution and key update can be implemented based on the shared secret key. The shared secret key can also be used as a method for authentication. This shared key can be programmed using a secure

Non-Volatile Memory (NVM) at the time of application development. Compared to asymmetric cryptography, symmetric cryptography is used more often in practical systems because of its high throughput.

On the other hand, symmetric cryptography including block cipher and stream cipher is adapted to data encryption because of its fast operations (mostly XORs and permutations). Between two types of symmetric cryptography algorithms, stream ciphers are capable of generating the encrypted data stream very fast, but they are limited to only stream data encryption. Notwithstanding, block ciphers can be configured for various security functions using the operation modes to be used as a stream cipher, a block cipher or a mechanism for authentication. It is more flexible for applications to use block ciphers for different security purposes. Among block cipher algorithms, Advanced Encryption Standard (AES) [FIPS-197] is considered a well-studied algorithm which is widely used in the current standards not only for IoT but also other applications such as network protocols, data encryption, storage encryption, etc.

New block cipher algorithms which are lightweight in terms of hardware or software implementation and memory footprints have emerged recently, but they have come up with reduced security levels such as PRESENT [Bogdanov2007pau]. They, in fact, have small hardware implementation area but use more encryption rounds and smaller block sizes leading to lower throughput. More importantly, these lightweight algorithms are not adopted in the new IoT proposals yet due to the shortage of their studies in security and protocols. Thus, AES is still selected as the main primitive for security mechanism in the emerging proposals targeting IoT applications, for example, IEEE802.15.4 [LRWPAN], LoraWan [LoRaWan], Zigbee [Zigbee] and in other Internet standards.

The innovation of new lightweight cryptography algorithms [ISO-29192-2] comes from new applications which have limited power budget such as RFID applications [ISO-29167-1, NIST-SP-800-98]. RFID cards do not have their power supply. The energy is harvested through the reader's electromagnetic waves. The harvested power supply is limited while the conventional cryptography algorithms need large area footprint and power consumption. This leads to a new class of cryptography algorithms to support these devices. Figure 1.6 shows the differences between two types of cryptography algorithms. The traditional algorithms such as AES focus on the security feature which has high levels of security; while lightweight cryptography aims attention at the effectiveness of the algorithms not only in security feature but also in terms of performance, power budget, and throughput and so on. Current lightweight algorithms concentrate on reducing power consumption by using the hardware structure such as employing wire permutation. The security feature in lightweight cryptography is traded off with the power consumption and the cost of the area of the implementation in hardware; while the traditional algorithms are optimized for software implementation.

Traditional Algorithms	Lightweight algorithms
Focusing on security	Focusing on minimizing area
Optimizations for software implementations (permutations on bytes or words, less rounds)	Optimizations for HW implementations (permutations on bits, more rounds)
High levels of security	Trade-offs between security and area

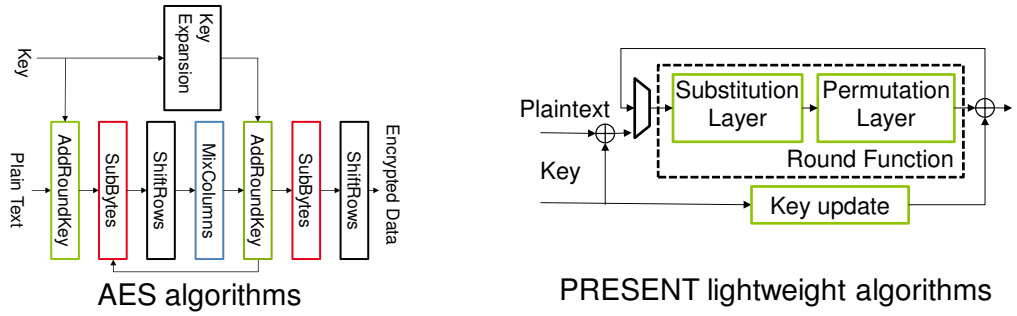


Figure 1.6: Comparison between traditional and lightweight block cipher algorithms.

Overall, a variety of security mechanisms for the Internet and applications could be implemented to ensure a certain level of security. However, they are considered to be more suitable for software on the computer. A new series of problems arising urges the implementation of security for constrained devices with limited power budgets such as RFID tags or power-harvesting devices. This leads to a new class of lightweight cryptography algorithms with small area footprint, ultra-low power consumption, reduced security levels and lower throughput. Finding a good trade-off among security, throughput, area, and power and energy consumption, therefore, will be a key issue for future IoT devices.

1.3 Security challenges in IoT

The first security problem lies in the scale and scope of IoT deployment. As a large variety of devices will be integrated, with varying security supporting mechanisms and communication/computation capabilities. As a result, a weakness in one of the devices in this system can reveal the secrecy of the whole system. Furthermore, the constrained IoT devices with limited computation capability and security mechanisms would become the weak points in the system. These devices are becoming a new attack surface in the IoT era. Figure 1.7 illustrates the number of devices in different layers in IoT systems and their corresponding security levels. In the clouds and the Internet backbone, strong security mechanisms can be applied with well-defined security features and threat models. However, at perception layers, for the constrained devices, lightweight security mechanisms are employed to reduce the cost and power consumption. With millions of devices involved, in the case of some devices containing weak security, the secrecy of the system could be vulnerable to

attack.

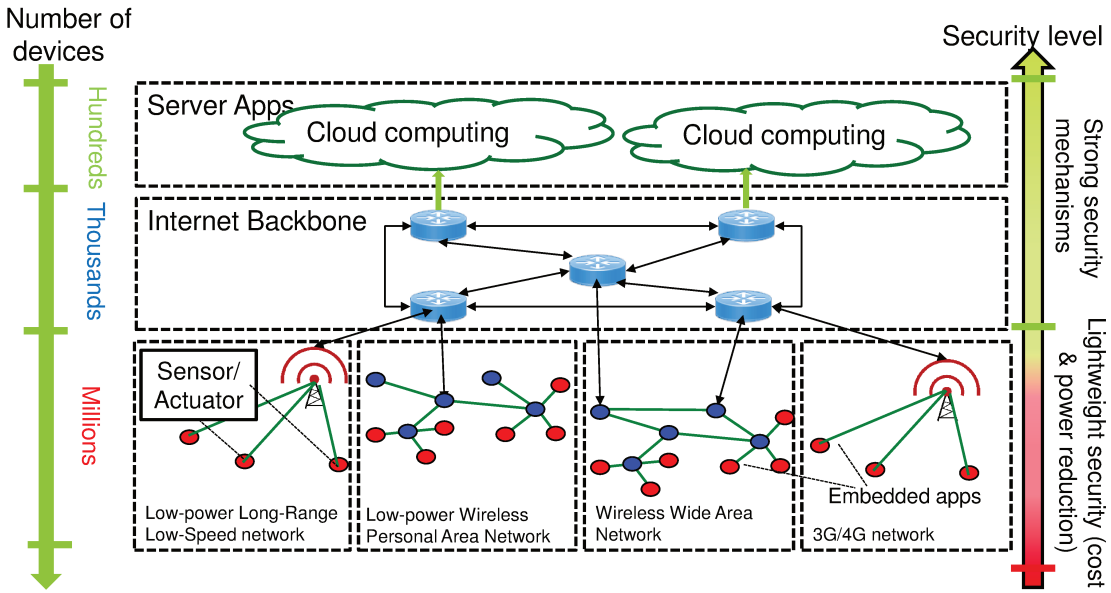


Figure 1.7: IoT security for different layers.

In addition to that, the current security mechanisms and algorithms contain multiple attacks which are also applicable to the IoT system. Figure 1.8 summaries the threats to different IoT components such as IoT devices, IoT communication and edge-level devices. First of all, at the algorithmic levels, cryptanalysis already reveals many weak algorithms. The secret key of the system could be revealed in a reasonable time with a reasonable number of data. An example of cryptanalysis is the failure of RC4 – a stream cipher widely used in GSM network. For communications, many attacks focus on the misuse of the cryptography algorithms such as reuse of the key, modification of the packets, fake packets, and device tag cloning and so on. For hardware implementation, side-channel attacks and fault attacks can be used for key recovery which is dangerous for IoT, because IoT devices are said to be wide-spread, and they are easily captured [Mosenia2017acs].

From technical points of view, system designers or hardware providers have to provide security mechanisms for all possible attacks as shown in Figure 1.8. This not only makes the system design more complicated but also increase the time to market as well as the cost of the final products. However, from the hackers' perspective, they can focus on only one weak point which may lead to the possible exploitation of the system. The design of IoT systems with strong security, therefore, becomes complex and expensive. Consequently, security holes have left current IoT products open to hackers.

Moreover, for constrained IoT devices, it is difficult to maintain a high level of security because of their small memory footprint and adequate computation capabilities. Running complicated security operations on these devices reduces their

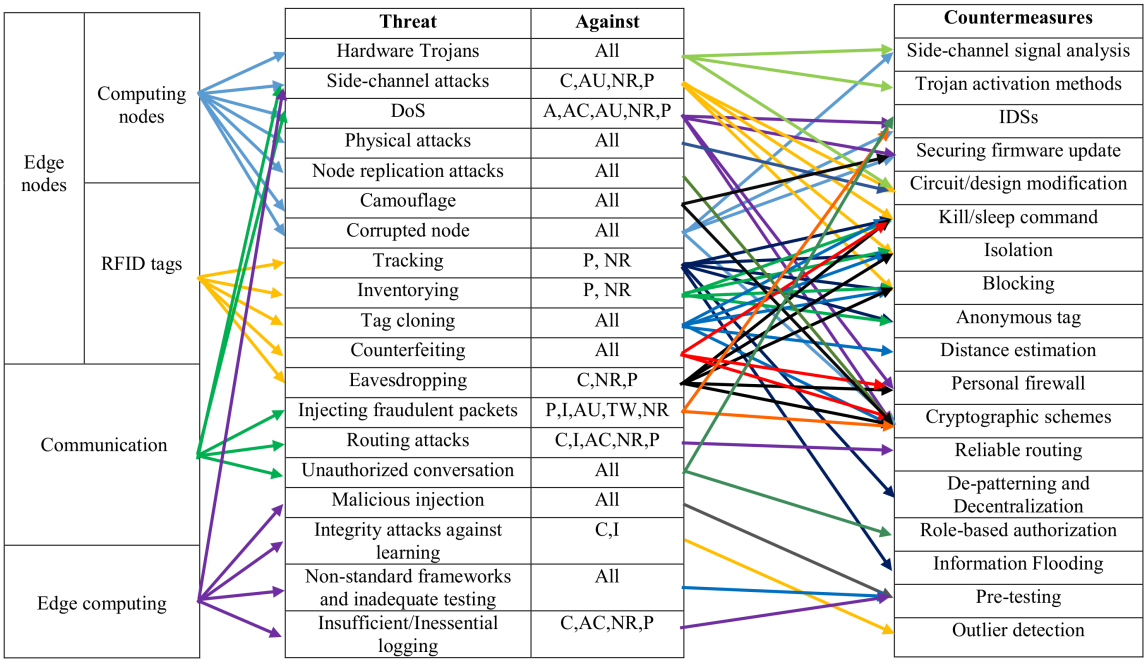


Figure 1.8: IoT security threats and possible countermeasures [Mosenia2017acs].

throughput and usability. Additionally, a high level of security on embedded devices will raise the cost of the system. Moreover, security evaluation with security evaluation lab's certificate is a long process which will add not only the cost but also the time-to-market to the product.

Apart from that, in line with the development of quantum computers and new computation architectures, the current cryptographic schemes are likely to be broken in the future. For example, the current asymmetric cryptography algorithms such as RSA are predicted to be collapsed easily by quantum computers. For that reason, new cryptographic primitives need to be developed to mitigate the new attacks from quantum computers. Some solutions could be practiced including lattice cryptography and homomorphic encryption; however, they are not as efficient in the current computer system. These new cryptographic schemes are even more costly and consume a large amount of power even when running at low throughput. They are not suitable for embedded system and especially constrained IoT devices. Despite the impractical implementations of lattice cryptography and homomorphic encryption at the moment, they open many new interesting application areas. For example, with homomorphic encryption, the data from IoT devices can be encrypted and processed in the encrypted domain. The data do not need to be decrypted in the server or the cloud. This is perfectly fit in a wide range of Internet applications including IoT. Figure 1.9 shows mechanisms which can be used with homomorphic encryption to provide better security and privacy for IoT systems. With homomorphic encryption, computers can work on encrypted data to serve encrypted requests from users. All the computations are in the encrypted domain; therefore, the providers do not know

the secrecy of the data and the request. This would be a far future of security and privacy. At the moment, these concepts are too costly for current computer systems and it is hard to apply to IoT constrained devices.

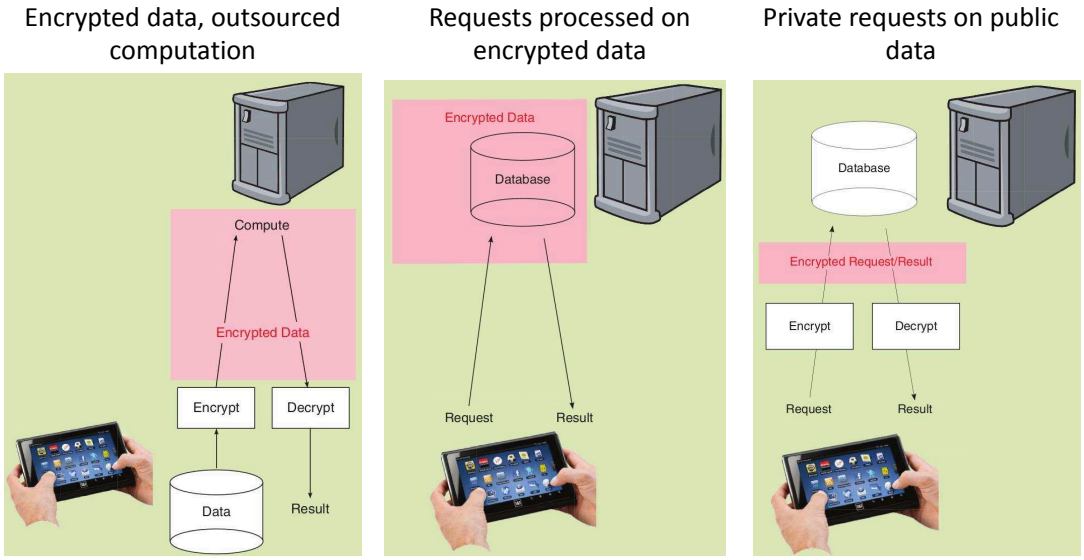


Figure 1.9: Homomorphic encryption and its applications to IoT [Aguilar2013rai].

The implementation of security functions in IoT also hinders in standardization. Different enabling technologies for IoT have different security mechanisms based on their needs. This leads to the problem of supporting multiple security mechanisms in a system with different components. For constrained IoT devices, there are lightweight cryptography algorithms and mechanisms which are already standardized; however, they are not used in the recent IoT proposals. In the recent proposals, AES is still used as the main security mechanisms for data encryption, data integrity and network protections leading to the demand for optimization of AES for cost, performance and power consumption for constrained IoT devices. Table 1.4 shows some recent proposals for IoT systems in which AES is widely used as the main security mechanisms. This is because AES is proved to be secure by a large number of researches on its security features.

New ideas have been proposed in the search for new mechanisms and more efficient computation for the security of IoT constrained devices, for instance, approximate Computing [Gao2017acf] or In-Memory Computing [Wang2016dad] for data encryption. However, further studies are needed to assess the effectiveness of these new mechanisms. In this work, the encryption using In-Memory Computing will be explored in terms of power consumption and security features. Two algorithms: AES – a traditional cryptography algorithm, and PRESENT – a standardized lightweight one will be investigated using the new computing mechanisms. PRESENT was chosen because it is designed for constrained devices which are expected to reduce the hardware cost and power consumption with a reduced security level, while AES is

Table 1.4: Some proposals for IoT with security features

	Sigfox	LoRaWAN	Thread	AllJoyn	Weightless
Startups	Sigfox	IBM & Semtech	Nest Labs (Google)	Qualcom	NWave
Type of License	N/A	Public spec/ patented radio IF (Semtech)	Free for members?	Open source license	Open standard
Type of specification	Low-power, wide range modem (N/A)	MAC/PHY	Network protocol targeting connected home	Wireless Application protocol and framework	MAC/PHY (N/A)
Type of network	Wide range LPWAN	Wide range LPWAN	IEEE 802.15.4 6LoWPAN Zigbee bluetooth 4.0	Wireless LAN Bluetooth	Wide range LPWAN
Reference model	N/A	LoRa WAN in C (IBM LRSC)	N/A	Source code release	N/A
Demos	In Service	LoRa in C	N/A	In production	N/A
Security	AES 128	AES 128 + MIC	AES 128 CCM + ECC	Full SW security stack	AES-128/256

a traditional algorithm with high-level of security. Pushing the encryption into the memory is expected to mitigate the risk of moving unencrypted data through the system bus as well as the risk of unauthorized access from other IP cores in the system. Furthermore, these memories with computing capability can be easily integrated into sensors even without an embedded processor.

Security implementation should be addressed carefully with good trade-offs among hardware cost, system throughput and power/energy consumption. Figure 1.10 describes the trade-offs among different aspects of the hardware implementation of the cryptography algorithms. At the algorithmic level, security is related to block sizes, key sizes and the number of rounds of cryptography algorithms. Algorithms, which have large block sizes, key sizes and numbers of rounds, usually provide high levels of security. However, large block sizes and key sizes require many registers to store them and the intermediate results which, in turn, require more hardware area and power consumption. In addition, a large number of rounds tends to reduce the throughput of algorithms and increase energy consumption as they use more loops to output the encrypted messages. Conventionally, cryptography algorithms are designed for security with software implementation to run in computer systems, however, in order to optimize throughput and power consumption for embedded systems, they are now integrated into a System-on-Chip system which is implemented in hardware. Along with security, the two new aspects, the total occupied area of the hardware implementation and its throughput must be considered. It will not be easy to optimize these aspects with new requirements of constrained devices, i.e. limited power

budget and energy consumption. In practice, most current lightweight cryptography algorithms are focusing on optimizing hardware area by reducing block sizes and key sizes with an increase in the number of rounds to keep an acceptable security level. The throughput can also be optimized by using round-unrolling architectures. This work also looks into the lightweight characteristic but adds other factors in consideration including power consumption and energy consumption. Lightweight to the extents of power consumption and energy consumption should be weighed for future IoT applications.

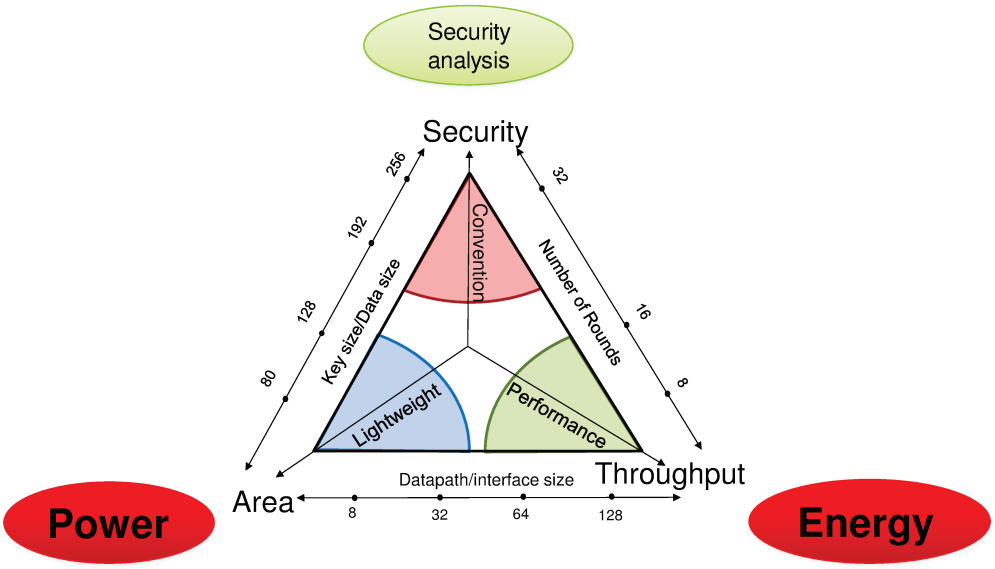


Figure 1.10: Security trade-offs.

In summary, security is a supplement but an important feature in IoT. However, security will increase the cost to the development process and a longer time to markets to cover different kinds of attacks. Furthermore, the development of security mechanisms in constrained devices is getting increasingly challenging as these devices have limited resources and processing power to support complicated security mechanisms. high demand for optimizing AES for security, hardware cost, and power/energy consumption has been frequently reflected through recent IoT proposals using AES as the main security mechanisms for confidentiality, integrity and communication protection. Considering those factors, this study seeks for a good trade-off among security, cost and power/energy consumption in hardware crypto-accelerator by implementing AES – a conventional algorithm, and PRESENT – a lightweight data encryption algorithm. However, hardware crypto-accelerators have limitations of flexibility and configurability to cope with new threats and to adapt to new IoT standards. Therefore, another part of this work is to investigate a new way of data encryption of these two algorithms using In-Memory computing mechanism.

1.4 Conclusion

All things considered, the Internet-of-things presents enormous benefits to our daily lives and society, but it also encounters major concerns about security. Security problems can be solved using standardized security mechanisms such as block ciphers, hash functions, RSA or elliptic curves; however, these mechanisms are not suitable for highly constrained IoT devices/sensor nodes with the ultra-low-power feature. Ultra-low-power devices using battery power supplies or energy harvesting enables new types of applications such as implant or wearable devices, health monitoring, environment monitoring with extended operational time. For these constrained devices, there are trade-offs among security level, hardware cost and throughput. In general, a high-security level often requires costly hardware and high power consumption.

The feasible solution of security for ultra-low-power devices is to use block ciphers or lightweight block ciphers to implement various security task such as data encryption, authentication and identification using pre-shared keys and so on. The recent proposals for IoT use Advanced Encryption Standard (AES) as the main security primitive for the system such as LoRaWan, Sigfox, Z-wave and so forth. They are often implemented in hardware accelerators to reduce the power consumption and increase the system throughput because software implementation cannot provide high throughput and consume the power. However, software implementations of security functions provide flexibility and configurability because new mechanisms can be implemented into the devices using the software update. On the other hand, hardware accelerators have fixed designs, and they cannot adapt to new standards and mitigate new attacks.

Furthermore, different applications might have different security requirements with different power budgets. Therefore, a configurable solution will adapt better to a wide range of applications. Conventional block ciphers such as AES can provide high levels of security while lightweight block ciphers such as PRESENT has smaller hardware area and lower power consumption. The combination of both types of block ciphers can help IoT applications lengthen their operations by selecting high levels of security when there is more stable supply and when in the low power mode, they can use the lightweight one.

In addition, with the new advancement of memory technologies, In-Memory Computing can be used to implement various security primitives in place without the requirement of moving the data out of the memory. This not only provides high flexibility and high configurability but also reduces the risks of exposing secret data through the system bus. However, this new mechanism needs thorough investigations and evaluations.

With a view to improve the current situation, this work first and foremost concentrates on the power/energy consumption optimization of block ciphers which can be used to implement different security primitives to secure the data and communication in IoT systems. The power consumption of both conventional and lightweight

algorithms will be carefully evaluated. Then, the work proposes a low-power implementation of two standardized algorithms which can be used for ultra-low-power IoT devices. The first one, Advanced Encryption Standard (AES), is a conventional algorithm widely used to secure the Internet applications with high levels of security. The other is PRESENT, a new lightweight algorithm which uses hardware constructs to reduce the hardware area and the power consumption. On the other hand, IoT devices might have different security requirement depending on the applications and the available power/energy budget. Therefore, in the next step, this work combines the two modules, AES and PRESENT, into a crypto-accelerator with various key sizes and block sizes. IoT applications can choose the high secure algorithm with more power consumption or in the critical condition the lightweight one to lengthen their operations. Not only optimizing the power consumption, but the security evaluation using the current state-of-the-art methods are also applied to the proposed design. Two notable evaluation methods are employed in this work including Correlation Power Analysis (CPA) and Test Vector Leakage Assessment (TVLA). TVLA can address different information leakage in the proposed designs while CPA can be used to mount the key recovery attack. Last but not least, from the lesson of various security breaches because of the fixed hardware security module, this work finally explores the configurability, the flexibility and the feasibility of different block cipher algorithms using In-Memory Computing. This work puts forward the implementation of two algorithms, which are previously designed in the aforementioned crypto-accelerator, using the In-Memory Computing technology to explore the trade-offs of the flexibility and the configurability along with throughput and power/energy consumption.

Chapter 2

State-of-the-art of security hardware in IoT

Simple cryptography algorithms have been used for a long time to protect private and confidential information such as military or diplomatic communications. For example, Enigma Machine used by German Army during the World War II was equipped with an electro-mechanical rotary cipher which was said to be very hard to break by humans at that time because of an extremely large number of combinations. However, as a result of the development of computer machines, it was then broken by a machine designed by Alan Turing. With the help of the machine designed by Alan Turing, British military was able to read the encrypted messages transmitted by the German. This shows the danger of transmitting secret information in a public environment. In the era of the connected world, information is transmitted over the Internet and cryptography is becoming increasingly important to protect the information secrecy and integrity. Nevertheless, multiple vulnerabilities have been discovered in ciphers such as Data Encryption Standards (DES) [Biham1990dcd, Matsui1994lcm] or RC4 [Mantin2005pad] which were used in the GSM (Global System for Mobile communication) network. Therefore, to reduce the risks of using weak algorithms, new applications should use well-studied and standardized algorithms. This is why Advanced Encryption Standard (AES) is chosen as the main security primitive in the recent IoT proposals.

However, conventional algorithms with strong security levels such as AES are often designed for software implementation with optimization for software. They have been implemented in hardware to increase performance, and they need to be optimized for hardware cost and power consumption, but they have larger area and higher power consumption compared with the new lightweight cryptography algorithms which are designed specifically for hardware implementation. Lightweight algorithms such as PRESENT were designed with the trade-offs among hardware cost, power consumption and security. They are more efficient in hardware than in software and sometimes with reduced security levels to reduce the hardware cost and area power consumption.

To optimize the security functions for IoT, block ciphers and lightweight block ciphers are often implemented in hardware to reduce the cost and power consumption. However, these optimizations lead to the fixed hardware structures which cannot adapt to new standards and mitigate new attacks. To improve flexibility, many algorithms can be combined into a configurable architecture. Different optimizations can be applied to reduce the hardware cost and power consumption. However, flexibility and configurability still have to trade off for high power consumption.

To push one step further, In-Memory Computing and Near-Memory Computing can be deployed to improve the flexibility and configurability of block-cipher implementations. In-Memory Computing provides logical operations and storage while Near-Memory Computing puts the operations closer to the memory. Therefore, the power consumption caused by the communication overheads of transferring data from the memory to IP cores or processing elements the communication overheads can be saved. However, because of the serial operations of the memory, it often requires a large number of clock cycles than in the case of fixed hardware.

This chapter aims to review the recent works on designs of cryptography algorithms and their hardware implementations for constrained devices. It discusses various techniques to implement the block ciphers in hardware for performance, area and power consumption. It also summarizes various optimization techniques using emerging technologies such as In-Memory Computing, Near-Memory Computing, Domain Wall Nano Wires, and so on. Importantly, it summaries the current state-of-the-art of hardware implementations of various cryptography algorithms.

Based on the above analyses, this work then proposes its own crypto-accelerator for ultra-low-power consumption which will be discussed in details in Chapter 3. However, the crypto-accelerator has limited configurable options and flexibility. Therefore, flexibility and configurability are further explored in Chapter 4 using In-Memory Computing. The same algorithms in Chapter 3 are demonstrated in Chapter 4 to explore the trade-offs among flexibility, configurability, throughput and power/energy consumption.

This chapter is organized as follows. Firstly, the hardware architectures of symmetric cryptography algorithms, especially the block ciphers and the power consumption optimizations for CMOS technologies are presented in Section 2.1. Both conventional algorithms and new lightweight algorithms are discussed. After that, the current state-of-the-art of hardware implementations of AES and various lightweight cryptography algorithms, including power optimization is shown in Section 2.2 and Section 2.3, respectively. This chapter then discusses existing reconfigurable hardware implementations of cryptography algorithms in Section 2.4. It includes the previous works on In-Memory Computing and Near-Memory Computing. Section 2.5 presents various accelerations using memory elements and the trade-offs between flexibility and configurability with throughput and power consumption. The hardware security including the hardware evaluation methods is described in Section 2.6. Finally,

Section 2.7 presents some conclusions and perspectives on the state-of-the-art.

2.1 Introduction to symmetric cryptography hardware architecture and its power consumption optimizations

This section firstly assesses the current designs of various symmetric cryptography algorithms and their hardware architectures. There is no specific method to design a highly secure block cipher. However, there are some agreed secure primitives and/or operations which have been used to design most of the symmetric cryptography algorithms. After that, the optimization techniques which can be applied to the hardware design of these algorithms will be discussed.

2.1.1 Symmetric cryptography hardware architecture

Symmetric cryptography algorithms are cipher algorithms which require both the sender and the receiver to share the same key. The sender will use the key to encrypt the data while the receiver will use it to decrypt the encrypted data. Symmetric cryptography contains block ciphers and stream ciphers. Block ciphers work on fixed width bit vectors. For example, Data Encryption Standard (DES) has the block size of 56 bits, while Advanced Encryption Standard (AES) has the block size of 128 bits. The encryption is carried out by transforming the data blocks using permutations and some linear/non-linear transformations. On the other hand, stream ciphers generate a pseudo-random bit stream. The encryption is performed by XORing the plaintext with the generated pseudo-random bit stream. The general structure of a stream cipher is presented in Figure 2.1. To keep the algorithm secure, stream ciphers often require an initialization vector (IV) and an initialization process before generating the pseudo-random key stream. The advantage of stream ciphers is that they can generate a random value very fast. However, stream ciphers are only limited to stream data and cannot be used in other modes of operations.

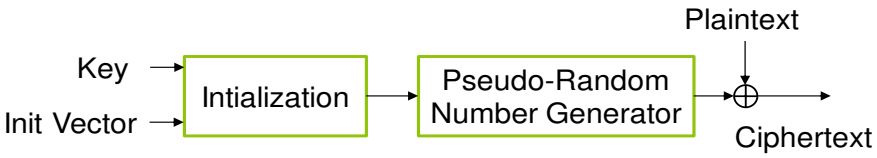


Figure 2.1: Stream cipher structure.

Block ciphers work on fixed-size blocks of data and apply different transformations including linear and non-linear ones to diffuse the input data. Block ciphers and hash

function define round functions which are repeated with different keys to keep it secure. Figure 2.2 shows the general structure of block ciphers. Each round contains non-linear layers and linear permutation. The non-linear substitution layers are often constructed using a look-up table (S-Box). The permutation layers are often the permutations of bytes or bits. It may contain other linear arithmetic operations. The speed of block ciphers is often decided by the speed of its primitive operation such as the substitution boxes (S-Boxes), the linear operations and the number of rounds. The more rounds are executed, the more secure the algorithms are. However, the performance will be degraded. In general, block ciphers are slower in speed when compared with stream ciphers. However, block ciphers can be used for different purposes using the operation modes such as Message Authentication Code (MAC) – a lightweight version of hash functions, or as a stream cipher in counter mode or Cipher Block Chaining (CBC) mode. Figure 2.3 shows the possible modes of operation of block ciphers to be used as a stream cipher or a message authentication code. These configurations are adapted in a lightweight protocol such as Counter with CBC-MAC (CCM) used in LoRaWan [LoRaWan].

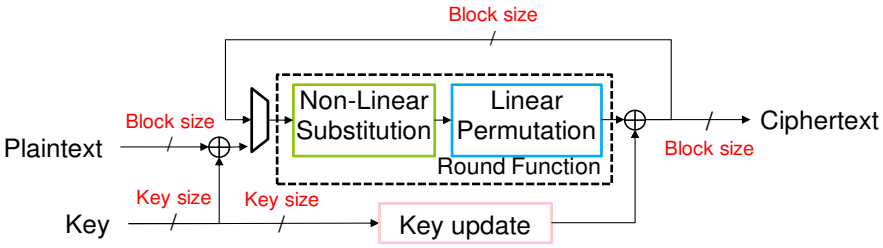
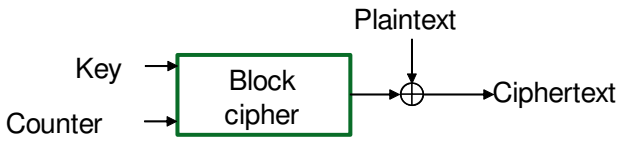
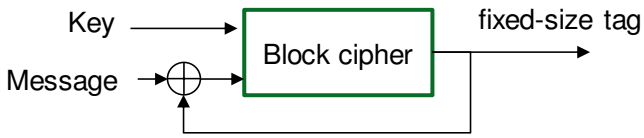


Figure 2.2: Block cipher structure.



(a) Block cipher in counter mode (similar to stream cipher)



(b) Block cipher in feedback mode (similar to hash function)

Figure 2.3: Block cipher in different operation modes.

When personal computers became popular, block cipher algorithms were mostly designed for software implementation on computers such as Advanced Encryption Standard which was designed to be efficiently executed on modern computers. However, the rapid development of many constrained systems such as smart cards or the power harvesting devices makes it necessary to develop a new class of lightweight block ciphers. The design of lightweight block ciphers is similar to the traditional algorithms, but with the optimization for hardware implementation. For example, to reduce the hardware cost, the lightweight block ciphers often reduce the block size and the key size. This reduces the security level but more importantly, it reduces the hardware size and power consumption. The permutation layers use the hardware permutation which is the permutation of wire in the circuit. Instead of large S-Boxes in traditional algorithms, lightweight algorithms use 4-bit S-Boxes which can easily be represented in combinational logic or use a small look-up table. New lightweight block ciphers are friendly to hardware implementation, but they are not efficient in software because they use many bit-level operations which take more time to complete when implemented in software.

Among a large number of block ciphers, in this work, two standardized algorithms are selected for evaluation. The first one is Advanced Encryption Standard (AES) which was designed for efficient software implementation on personal computers. The other is PRESENT – a recent lightweight cryptography algorithm targeting constrained devices such as smart cards and embedded systems.

In summary, new advances in cryptography enable a new class of cryptography for constrained devices which is called lightweight cryptography. Most recent lightweight cryptography algorithms focus on optimizing the implementation area (hardware cost) and power consumption, while the throughput is degraded. Despite no specific guidance according to the recently-announced algorithms, lightweight cryptography has been designed with lightweight primitives such as small S-Boxes, reduced key size and reduced block size. Lightweight cryptography provides decent security level and might be broken in the future. On the other hand, AES is still a strong algorithm and recommended for many applications. It provides from medium to long-term security level. However, its complexity makes it not suitable for constrained devices because of hardware cost and especially the high power and energy consumption.

2.1.2 Power consumption optimizations for CMOS technologies

CMOS technologies replaced previous technologies such as bipolar ones because they can provide faster switching rates with lower power consumption. The current integrated circuits are mostly fabricated in CMOS technologies. This section presents the power consumption characteristics for CMOS technologies which will be used to understand more clearly the different power consumption optimization techniques in the following sections.

In a CMOS circuit, total power consumption is the sum of dynamic power consumption $P_{dynamic}$ and static power consumption P_{static} as in Equation 2.1. Dynamic power consumption is caused by charging and discharging load capacitances when gates switch or by the short circuit current when both NMOS and PMOS are in the open state (Equation 2.2). Switching power consumption is related to load capacitance (C), supply voltage (VDD) and operating frequency (f) of the circuit as in Equation 2.3. On the other hand, static power consumption is caused by the subthreshold leakage through the OFF transistor; gate leakage through gate dielectric; junction leakage through source/drain diffusion; and contention current in ratioed circuits (Equation 2.4).

$$P_{total} = P_{dynamic} + P_{static} \tag{2.1}$$

$$P_{dynamic} = P_{switching} + P_{short-circuit} \tag{2.2}$$

$$P_{switching} = \alpha \times C \times V_{VDD}^2 \times f \tag{2.3}$$

$$P_{static} = (I_{subthreshold} + I_{gate} + I_{junction} + I_{contention}) \times V_{VDD} \tag{2.4}$$

Power consumption can also be divided into different categories. Active power is the power consumption of the circuit when it does something helpful. At the idle time, the circuit can be put into the standby mode to reduce its power consumption. This is called the standby power. When in the standby mode for a long time, the circuit can also be switched into the sleep state so that it would consume even less power, which is called the sleep power. Active power is dominated by dynamic power consumption while standby power consumption is dominated by the leakage. To optimize the power consumption during the active state, it is more important to optimize the dynamic power consumption. The leakage power should also be minimized during the standby and sleep state because the circuit does not do helpful works.

Various techniques can be applied to minimize the power consumption of a CMOS circuit. Power optimization can be performed from architecture levels down to fabrication process. For dynamic power consumption, the focus is on the switching power consumption. As described in Equation 2.3, switching power is related to the load capacitance, supply voltage and operating frequency. Reducing the supply voltage has a high impact on power consumption because switching power has a quadratic relation with supply voltage. However, reducing the supply voltage might affect the performance of the circuit because at lower supply voltage, the transistors in the circuit switch slower and they are more susceptible to noise. For this area, there is a new trend in near-threshold operations range of CMOS circuit. Reducing supply voltage also affects the maximum operating frequency (f) of the circuit. Furthermore, the operating frequency can also be controlled to give enough performance with reduced power consumption. In addition, voltage and frequency can be controlled together to give better performance. Methods such as Dynamic Voltage Frequency

Scaling (DVFS) [LeSueur2010dvf] or Adaptive Voltage Frequency Scaling (AVFS) [Beigne2011ail] are widely applied in the current products. At the microarchitecture level, clock-gating is one of the efficient ways to reduce the dynamic power consumption because it reduces the switching activity of the clock distribution network. The idea of clock gating is to cut off the clock signals connected to sequential elements when they are not activated. New EDA tools support the insertion of an integrated clock gating (ICG) cells to registers with an enable signal [Benini1994spb]. At the microarchitecture level, minimizing the glitches of the circuit can also help to reduce the power consumption. Glitches are created because of the different propagation delays.

To optimize the leakage power during the idle state of the circuit, power gating is widely used. If the circuit is idle for a long enough period, the power distribution to the circuit will be cut off. As a result, the circuit will consume little power because there is a tiny static power consumption and no dynamic power consumption. However, power gating needs careful controls and consideration so that the circuit will work correctly after being powered on again. State retention can be used to maintain the state of the circuit during the power cut-off. The working states will be restored when the circuit is powered on. Due to the state retention and isolation, power-on and power-off sequences takes time and have to be activated carefully. Therefore, power gating is only effective when the power is cut off in a long period. Bad controls of power-on and power-off sequences might leads to the increase of power and energy consumption. Other techniques can also be applied such as using multiple threshold voltage transistors (multi-V_{th}), Reserve/Forward Back-Biasing (RBB/FBB) [Pagliari2018fbb] and optimization in the fabrication process.

In summary, optimizing power consumption of integrated circuits is important for constrained devices. There are multiple methods to optimize different aspects of the circuit. The optimization at the architecture level often has the most impact on the results.

2.2 AES hardware implementation

Advanced Encryption Standard (AES) is a round-based block cipher with the block size of 128 bits supporting the key size of 128 bits, 192 bits, and 256 bits with 10 rounds, 12 rounds and 14 rounds, respectively. It has been standardized in 2001 under the name FIPS-197 by U.S National Institute of Standard and Technology (NIST) [FIPS-197] and then included in ISO/IEC 18033-3 [ISO-18033-3]. AES encryption procedure is shown in Figure 2.4. Firstly, the 128-bit data block is divided into 16 bytes and arranged into a matrix of 4×4 bytes so called the state matrix. All AES operations work on this state matrix. There are four basic operations in a round of AES encryption datapath including AddRoundKey, SubBytes, ShiftRows, and MixColumns. AddRoundkey step is the XOR of the state matrix with the 128-bit round

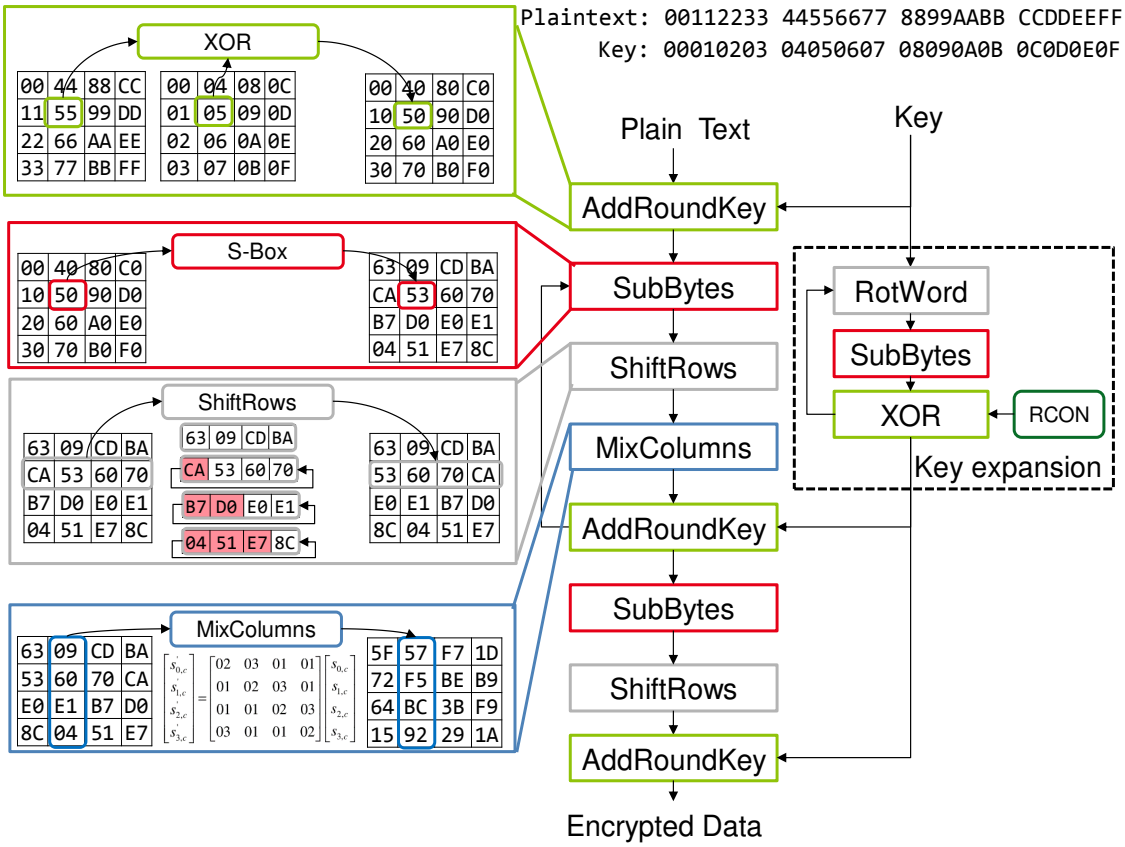


Figure 2.4: AES encryption algorithm in details.

key. SubBytes transform the state matrix byte-by-byte using a non-linear mapping function. This function can be used as a look-up table (LUT) or by using arithmetic in the finite field GF(2^8) and an affine transformation. ShiftRows transform the state matrix by rows. Each row is rotated by a different number of bytes while MixColumns transform the state matrix by its columns. In these steps, only SubBytes contain non-linear operations while the other steps are linear operations. Each round needs a different round key generated by the key expansion algorithm. The key expansion is composed of three operations: RotWords, SubWords, and XORs. SubBytes and SubWords are similar because they both implement S-box operations, while RotWord is similar to ShiftRow operation. RotWords and SubWords are only applied to the first word in the key matrix. For AES with 256-bit keys, an additional SubWords is added for the fifth word in the key matrix.

Recently, in the wake of information security issues, various works have been tapped into optimizing AES for various purposes. The most notable efforts into optimizing area and power/energy consumption of AES are summarized in Figure 2.5. The areas of these works are normalized using the number of 2-input NAND gate equivalence (GE) while the energy consumption utilizes the notation of en-

ergy per bit which is the required energy to processing a bit of data. The energy per bit is calculated by dividing the total energy required to process a data block by the block size. For high-speed applications such as optical links or high-speed networks, AES is implemented in hardware with the round-based implementation [Liu2011A2G] or pipeline architecture [Mathew20115GN] or unrolled-round architecture [Maene2015sio]. On the one hand, these kinds of architectures can provide gigabits of throughput, they, on the other hand, require high power consumption which is not suitable for embedded systems or constrained devices. For examples, in [Mathew20115GN], Mathew *et al.* present a two-stage pipeline architecture which can provide a throughput of $53Gbps$ with the power consumption of $125mW$. Nevertheless, these designs often consume more than 15,000 2-input NAND Gate Equivalences (GEs) to hundreds of thousand GEs. The biggest part of the parallel architectures is the S-box. In round-based architecture, there are 16 S-boxes for the encryption path and 4 S-Boxes for the key expansion. In this case, the S-boxes may occupy half of the total area. Round-based architectures often require 10 cycles/encryption; unrolled implementations take from 1 to 5 cycles/encryption while pipeline architectures can complete the encryption of a block in one cycle after the pipeline is fulfilled.

Most existing designs for low-cost and low-power AES focus on 8-bit datapaths. 8-bit datapath designs can reduce hardware implementation area significantly with the cost of reducing throughput as they use one [Mathew20153m1, Wamser2017ptl] or two S-boxes [Zhao2015nsa, Moradi2011ptl]. The theoretical limit of 8-bit datapath is 160 cycles/encryption. Extreme small designs such as in [Mathew20153m1] and [Moradi2011ptl] require more than 200 cycles/encryption. The activities in 8-bit datapath are reduced since there are only 8 bits processed in a clock cycle with the cost of additional registers for MixColumns and additional gates for control logic. The additional registers for MixColumns are required because MixColumns work on 4-byte column data which are available only when the whole column is processed. To achieve medium and high throughput, 8-bit datapath architectures have to run at a high frequency up to *GHz*.

Moreover, the authors in [Jean2017bag] put AES hardware implementation to another extreme by using the bit-sliding technique. By using a 1-bit datapath architecture, they reduce the hardware cost of AES down to 1500 GEs. However, this architecture takes an extremely long time to finish one encryption (up to 1700 cycles). They also explore other options for implementing AES algorithms using bit-sliding techniques. The 8-bit datapath presented in their work is close to the current state-of-the-art. Bit-sliding technique can be applied to other algorithms which use the Substitution-Permutation Network (SPN) as AES or PRESENT.

The further area reduction for 8-bit datapath is done by optimizing the S-box. In the AES standard, the straightforward implementation of the S-box is to use a look-up table (LUT). However, LUT-based implementations require large area footprint. Moreover, AES standard [FIPS-197] also describes AES S-box as an invertible

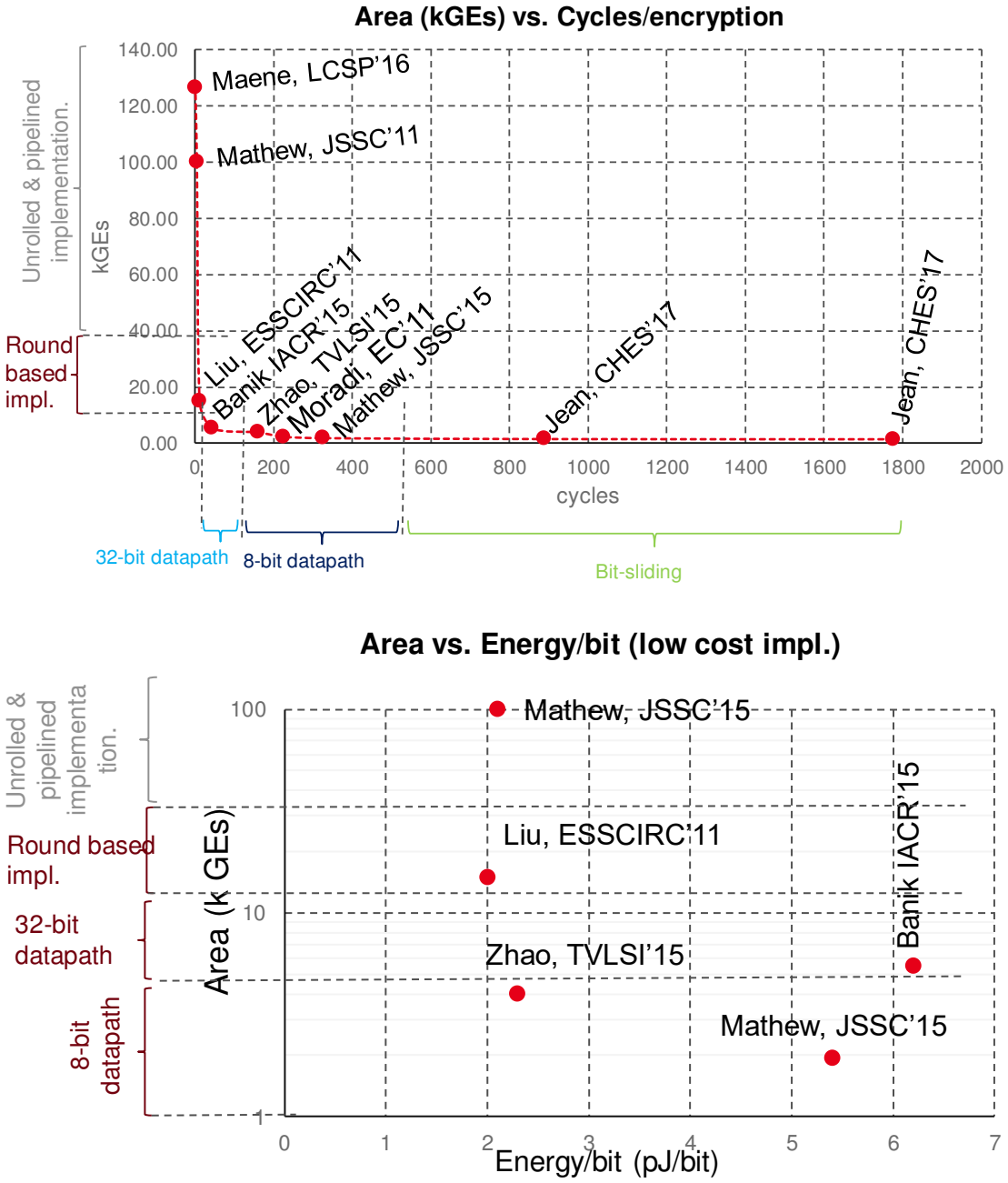


Figure 2.5: State-of-the-art of hardware implementations of AES.

non-linear function in the finite field. The first step is to find the multiplicative inverse of the input byte in GF(2^8). In the second step, an affine transformation in GF(2) is applied to the output in the first step. Many works tried to optimize S-box further by using equivalent representations of S-Box such as the tower field which is the decomposition of GF(2^8) into GF(((2^2)^2)^2) [Canright2005AVC, Satoh2001acr]. This can reduce the area of S-box to about 290GEs/S-box while the LUT-based implementations require at least 400GEs/S-box. Another decomposition of GF(2^8) is to use the normal basis of GF((2^4)^2) as in [Mathew20153m1]. These methods reduce the size of the S-box, but the unbalanced datapath of the S-boxes introduces more activities. In [Bertoni2004PAS], Bertoni *et al.* present a method to synthesize the S-box for low power by using the Decode-Switch-Encode method. It can achieve lower power consumption but requires more area than the previous methods.

Another option in optimization is to use 32-bit datapath. AES algorithm is designed for software implementation in modern computers with 32-bit instruction set architectures. Therefore, AES in 32-bit datapath has some advantages. The number of S-box is reduced, instead of 20 S-boxes in round-based architectures; 32-bit datapath uses only four S-boxes (in case of sharing the S-box between the encryption path and the key expansion) or 8 S-boxes (without sharing). The number of cycles required for one encryption is about 44 to 54 cycles [Satoh2001acr], at least 4 times higher than 8-bit architectures. 32-bit datapath architectures also use fewer registers than 8-bit datapath because the MixColumn step may have data of the whole column in one clock cycle. This opens an opportunity to optimize the architecture further for area, throughput, and power/energy efficiency.

2.3 Lightweight cryptography implementation

One of the applications with mandatory security requirement is RFID tag, although it is a small electronic card with power supply harvested from the reader. Therefore, it is considered as the first highly-constrained devices. The development of RFID tags also affects security trends. In particular, traditional security functions which require a large hardware area and high power consumption are no longer suitable for RFID Tag. To overcome this limitation, a new class of cryptography algorithms called lightweight cryptography was brought into play. It focuses on scaling down the hardware area and power consumption but with a reduced security level. To be more precise, many algorithms decrease the key size and the block size to reduce hardware area and power consumption, but this will reduce the security level. The block size is commonly seen ranging from 48 bits to 64 bits, for example, SIMON [Beaulieu2013tsa], TWINE [Suzaki2011tal], KATAN [Canniere2009kak], KLIEN [Gong2012kan] and so on. Some others come in 128-bit block size such as CLEFIA [Shirai2007t1b]. The key size is also reduced between 80 bits to 128 bits in order to maintain a decent security level. Smaller block size and smaller key size will help to reduce the hardware area, as a

consequence of the use of fewer registers to store the data and the key. Additionally, such reduction can help cut down on the power consumption to the respect that the dynamic power consumption is related to the activity of the registers and the static power consumption is related to its area. Another notable point is that lightweight block ciphers use hardware structure for permutation and smaller S-box in comparison with a traditional algorithm such as AES. However, to maintain the proper security level, lightweight cryptography requires a larger number of rounds which reduces the system throughput when running at the same frequency. Figure 2.6 reveals the area and the throughput of different cryptography algorithms. It is clear from Figure 2.6 that many lightweight cryptography algorithms result in smaller hardware area, but they generate very low throughput.

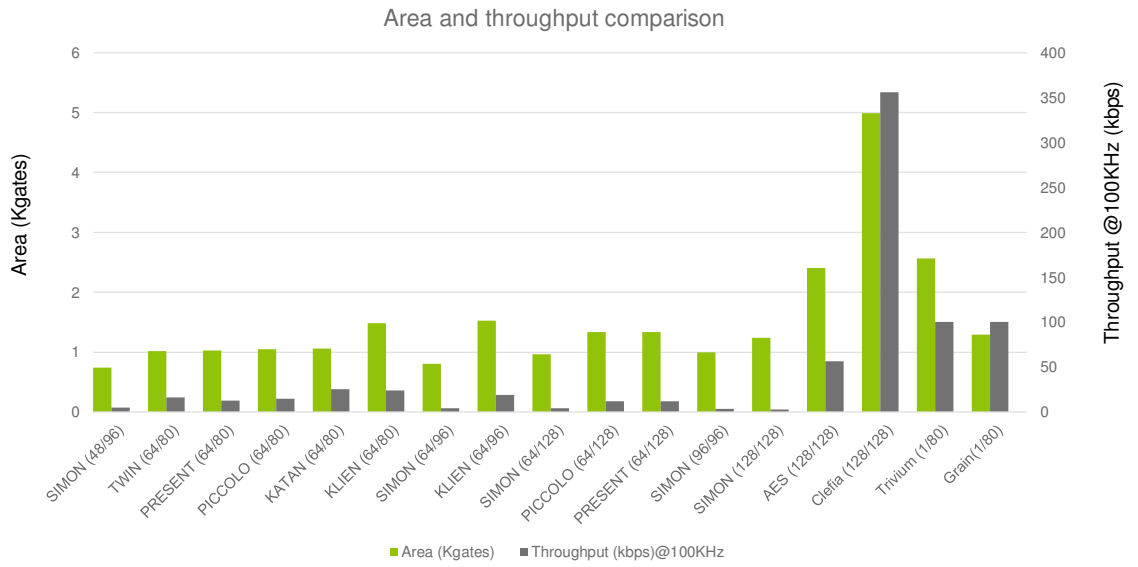


Figure 2.6: Area vs throughput of various cryptography algorithms.

Take the lightweight block cipher– PRESENT for example. PRESENT [Bogdanov2007pau] is a recently-standardized block cipher for constrained designs. In contrast to AES which uses byte level operations, PRESENT uses hardware friendly primitives to save hardware cost and power consumption. Figure 2.7 shows the PRESENT algorithm. To save hardware area, PRESENT uses an 80-bit key and 64-bit block size. Its extended version supports 128-bit key with the same block size. PRESENT uses 4-bit S-Boxes for the substitution layer instead of 8-bit S-Boxes as in AES. Furthermore, despite complex linear operations such as ShiftRows and MixColumns in AES, PRESENT uses only bit permutations for permutation layer. This can be done by wire permutation in hardware. However, because of its simple operations, PRESENT needs up to 32 rounds to keep it unbreakable on the algorithmic level within a reasonable time.

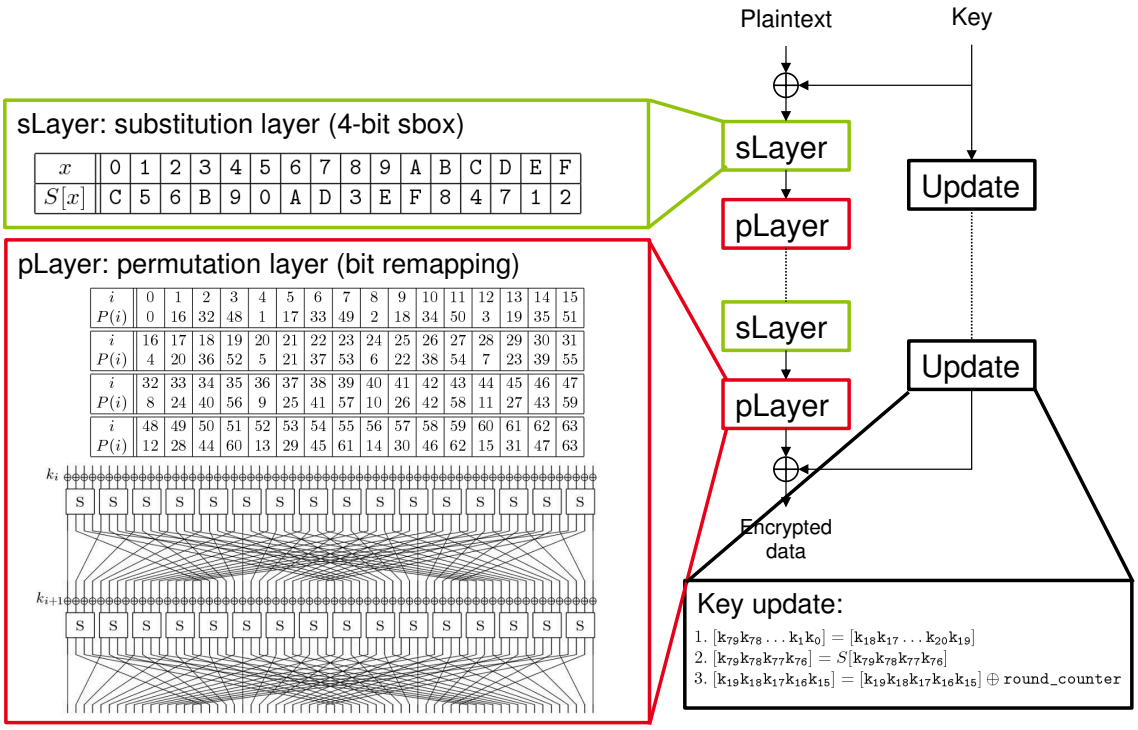


Figure 2.7: The PRESENT algorithm.

Among these algorithms, PRESENT and Clefia are standardized for lightweight block ciphers to be used in RFID application [ISO-29167-1] and lightweight block cipher [ISO-29192-2]. In addition, Grain and Trivium are selected as the standardized stream ciphers. Clefia supports multiple key sizes and block sizes similar to AES. It has high throughput but occupies a large area. Clefia, Grain and Trivium indicated smaller hardware area than the implementations of AES at the publication time of the ISO standard. However, running for the hardware implementation area, recent researches such as the work by Mathew *et al.* in [Mathew20153m1] show that AES encryption can also be implemented using about $2K$ GEs.

Along with the race for the area and power consumption, there are also algorithms which focus on optimizing the throughput. For example, PRINCE in [Borghoff2012pal] was designed to maximize its throughput with small hardware area and low power consumption. It was reported in [Borghoff2012pal] to occupied only $3.5K$ GEs, but has throughput up to $530kbps$ compared to AES (with only $8kbps$ of throughput with the same area). PRINCE has high throughput because it was designed for a fully-unrolled architecture to be implemented using a smaller area. Therefore, it can encrypt a block per clock cycle but with longer datapath delay as a trade-off. The unrolling technique can also be applied to other block ciphers to achieve high throughput but with larger area footprint and higher power consumption.

Lightweight cryptography might have low power consumption; however, it may

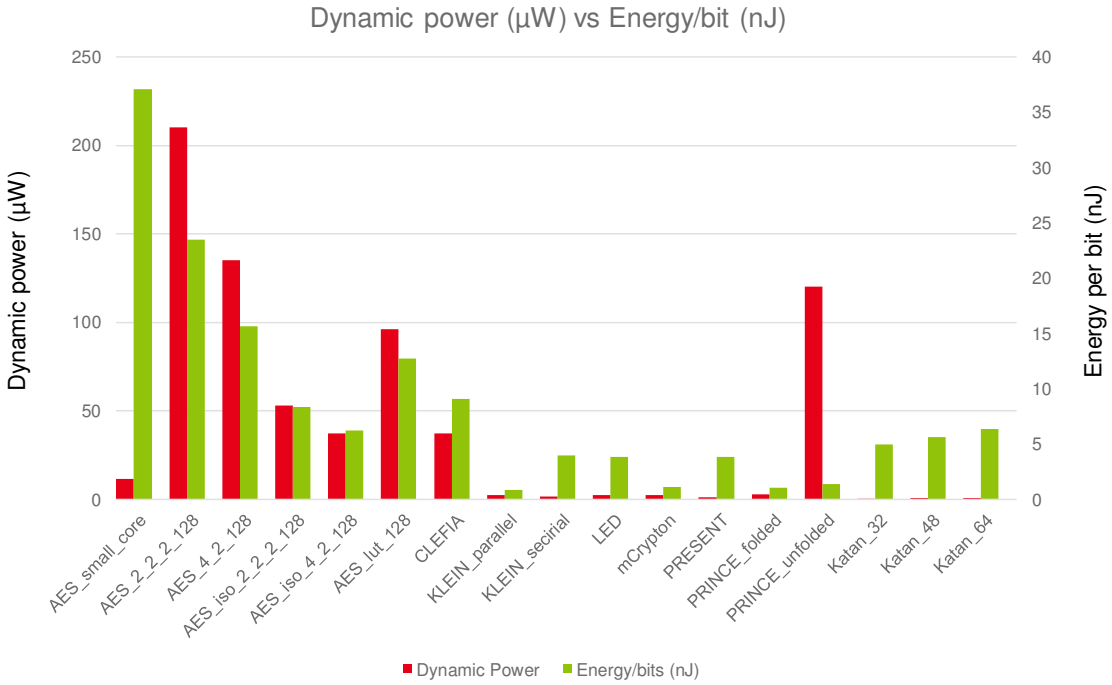


Figure 2.8: Dynamic power consumption vs Energy per bit of different cryptography algorithms.

consume a large amount of energy. Figure 2.8 shows the dynamic power consumption and the energy per bit of different hardware implementations of the traditional block cipher AES and some lightweight cryptography algorithms. Obviously, lightweight cryptography algorithms have low dynamic power consumption, but they are not efficient in terms of energy consumption. Theoretically, energy is related to the power consumption and the time that the chip consumes that power. However, lightweight block ciphers require a large number of rounds to keep its design security level which leads to low throughput and high energy consumption. In addition, energy consumption is becoming important for battery-based applications. Therefore, to optimize the energy consumption, both power consumption and the execution time must be optimized. For constrained devices, the power consumption has to meet the demands of the power budget while the throughput must be maximized. From Figure 2.8, AES can have good power and energy balance. Furthermore, some lightweight cryptography implementations can achieve lower power consumption, but they consume higher energy because of lower throughput.

Most of the early designed lightweight cryptography algorithms use hardware friendly structure to optimize the hardware area; however, they are not optimized for software implementation. Structures such as bit permutations or 4-bit S-boxes needs multiple instructions in software. Along with a large number of rounds, these lightweight cryptography algorithms have poor performance in software. This creates

the needs for lightweight algorithms which can run fast on both software and hardware. SPECK and SIMON [Beaulieu2013tsa] are designed with this idea. The authors claim the performance of 1.32 cycles/byte on Intel Processor [Beaulieu2015tsa].

With the development of new encryption methods such as homomorphic encryption [Brakerski2012fhe], calculations can be done in the encrypted domain. Two basic operations in the encrypted domain are the multiplication (logical AND) and the addition (logical XOR). Addition can be done easily, but multiplication in homomorphic encryption is complicated. Therefore, there are some algorithms focusing on reducing the multiplication depth so that it can be used more efficiently in homomorphic encryption [Canteaut2016sca, Albrecht2015cfm].

In summary, under tight requirements of constrained devices in terms of hardware area and power consumption, lightweight cryptography has been developed to reduce the hardware area and power consumption. However, they are observed not to be suitable for software implementation because of their hardware-specific structures. The area and power consumption are scaled down by reducing the security level with a smaller block size and a smaller key size, then applying the low-cost transform in hardware. Recently, other requirements have been taken into account, such as maximizing the throughput in both software and hardware or reducing the AND depth for homomorphic encryption. One of the areas which are not yet the focus of the current lightweight algorithms is to optimize for power/energy consumption. Different applications might have different requirements for throughput, power/energy budget and hardware area. Thus, finding the balance among these factors is one of the targets of this work.

2.4 Configurable hardware cryptography implementation

Security requirements vary among different applications which drive many works to focus on configurable hardware cryptography. Configurable cryptography modules implement multiple cryptography accelerators or multiple cryptography primitives so that they can be configured at run-time. This provides flexibility, but it increases the hardware cost and also the power consumption because the security primitives or accelerators must be included to be used when necessary. Even when they are not used, they might still consume power.

For example, Hutter *et al.* in [Hutter2011acp] proposed a configurable cryptography processor which supports both asymmetric algorithms and symmetric ones. This design uses an 8-bit microprocessor along with two accelerators. One accelerator is used for Elliptic Curve Cryptography (ECC) and the other is for AES. The architecture of this work is shown in Figure 2.9. It uses a microcontroller to manipulate a microcode ROM to control the two crypto-accelerator cores. The authors claim it as a low-cost solution with the area of 21 $KGEs$. Among them, AES acceleration

part occupies about 2.5 $KGEs$. In terms of throughput, this architecture takes more than 4500 cycles to finish one encryption.

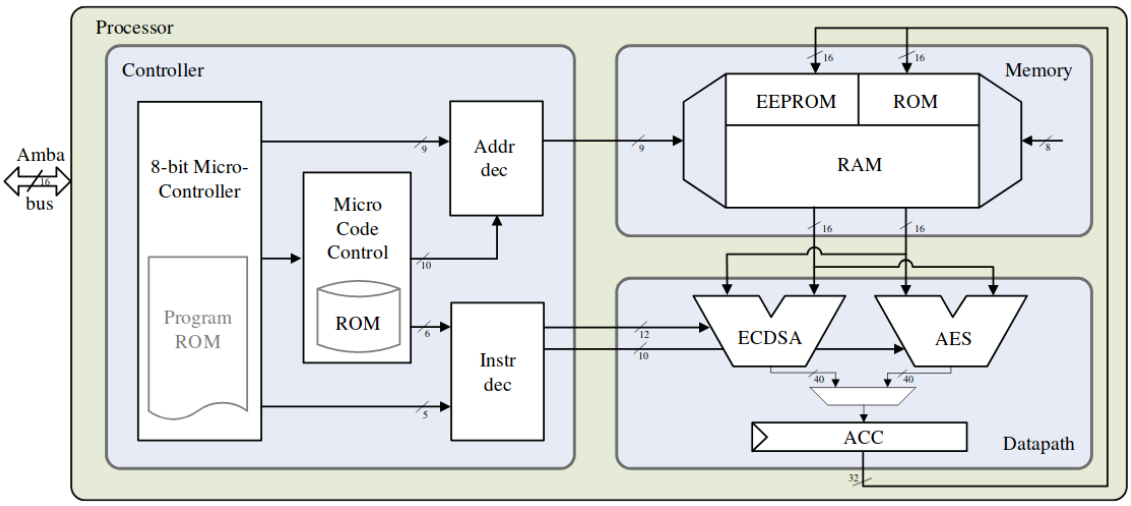


Figure 2.9: Reconfigurable crypto-processor using microcontroller and accelerators [Hutter2011acp].

A more generic way to design the configurable cryptography processor is to include all the individual primitives so that they can be composed into different algorithms. Sayilar and Chiou in [Sayilar2014cht] follow this direction. They design a processing element (PE) consisting of five configurable function units including Arithmetic Unit (AU), Logical Operation Unit (LOU), Table Lookup Unit (TLU), Shifter-Rotator Unit (SRU), and Permutation-Expansion Unit. The proposed system consists of an array of PEs connecting from one layer to another using a crossbar called connection row. By using this system, different cryptographic algorithms including AES, MD5, SHA-1, SHA-2, and stream ciphers such as RC4, RC5 and Phelix can be mapped into different PEs automatically or manually. AES implementation of this architecture takes 20 clock cycles to finish one encryption. The drawbacks of this architecture are high area cost of more than $6mm^2$ and high power consumption of nearly $1W/mm^2$.

Multiple algorithms with similar transformations can be grouped to create a reconfigurable module such as the work by Satpathy *et al.* [Satpathy2018grg]. This work implements three block ciphers including AES and SM4, and Camellia by using the common transformations of these algorithms in GF($(2^4)^2$). By using a hybrid S-boxes with shared inversion, affine scaling for MixcColumns of AES, Polynomial optimization, and key pre-computation, the authors can reduce the area by nearly 30 percent in comparison with the separate implementations. The design has been fabricated using $14nm$ technology with the power consumption of AES of $16\mu W$ at $0.24V$@2.1MHz.

In addition, a configurable crypto-accelerator using In-Memory-Computing and near-memory computing was proposed by Zhang *et al.* in [Zhang2018rar]. Figure 2.10 demonstrates their architecture with S-boxes, a bit rotators, a shifter along with a special memory which can perform logical operations such as XOR. This architecture supports AES, ECC and Keccak-f function (which is the SHA-3) [Bertoni2011tks]. For AES, the S-Box is designed in native GF($(2^4)^2$) with registers inserted before the inversion to reduce the glitches of the S-Box circuit. The authors claimed 80% runtime and energy savings when compared with their baseline processor architectures. In detail, AES needs 726 clock cycles to finish one encryption and 7.2nJ energy per bit.

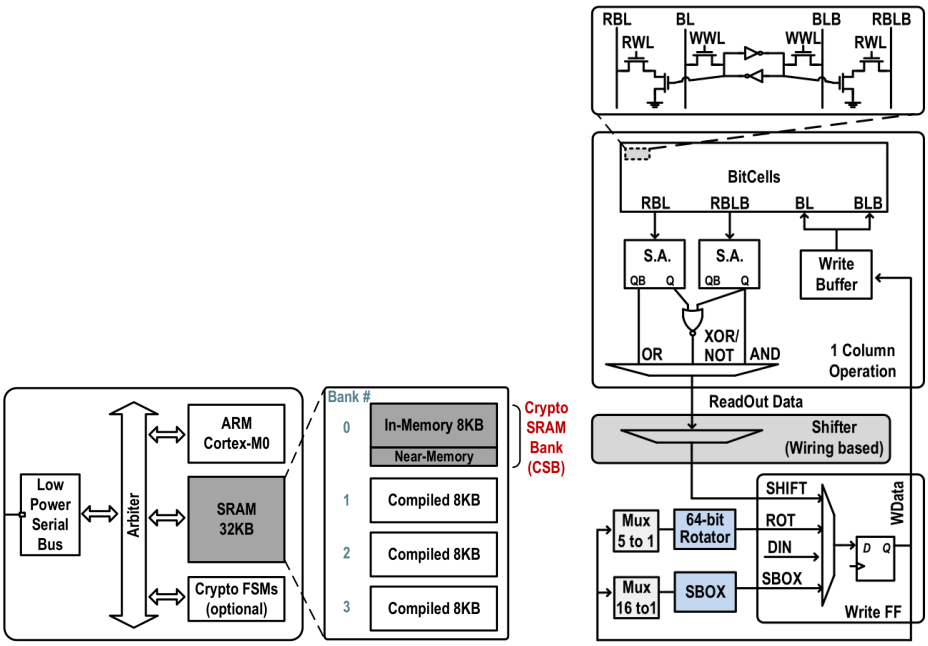


Figure 2.10: Configurable accelerator using In-Memory-Computing and Near-Memory computing by Zhang *et al.* in [Zhang2018rar].

In summary, configurable cryptography modules provide more flexible solutions than specific cryptography modules. Depending on the applications' requirements, different security mechanisms can be used with larger hardware area to exchange for flexibility. The drawback of these works is that they occupy a large area and consumes huge power consumption. These systems are typically in the class of nJ energy per bit, while the specific crypto-accelerators of AES often consume some of pJ. Due to high power consumption and cost, these systems are not yet suitable for constrained IoT system. In addition, In-Memory Computing and Near-Memory Computing show their potential to be applied to cryptography systems. It is because they can process the data near or in the memory which reduces the cost of transmitting data to the processor or the crypto-accelerators and concurrently reduces the risk of exposing the secret data through the system bus. Furthermore, the memory can also do parallel

operations not only with 32-bit or 64-bit as the processor, but also 512 bits. This opens a new research direction for future applications.

2.5 Encryption using memory elements

SRAM memory is widely used in electronics chips as a storage element. It has a high speed of operations and is one of the important parts of SoC architectures. SRAM even occupies most of the space of the chip as a cache memory to store temporary data and to prefetch instructions and data for faster access. SRAM can also be used as a look-up element. From early days, there have been ideas on implementing the S-Boxes using SRAM or a special type of SRAM such as Content Addressable Memory (CAM). For example, Labbe *et al.* [Labbe2004ehi] designed AES crypto-accelerator using SRAM elements to do fast S-Boxes look-up. Furthermore, Hua Li [Li2005anc] uses CAM as a way to implement S-Boxes and inverse S-Boxes. These works use SRAM in its pure form without any modification of the memory structure to do logic in memory.

New advances in semiconductors enable the discovery of new material for designing SRAM. For examples, Abid *et al.* in [Abid2009ecg] use new materials to design SRAM and use it to do cryptography. They used CMOL gate design to design an AES encryption core. After that, in 2016, Wang *et al.* [Wang2016dad] use Domain Wall Nanowire to design SRAM with the capability of doing shift operations and XOR operations. The authors use this new memory to design different operations of AES. Theoretically, the composed components of this type of memory into different AES sub-blocks can be used to build the whole AES encryption. However, this work is based on pure simulation.

2.6 Hardware Security

Another threat to constrained IoT applications lies in hardware attacks. Even though the algorithm is proved to be secure, its implementation in hardware or software potentially contains flaws which lead to different types of attacks such as fault attacks, timing attacks, side-channel attacks and so forth. The exploration of Differential Power Attack (DPA) [Kocher1999dpa] by Kocher *et al.* has shown that the microchips unintentionally leak information when they process it. This opens a new research area to protect the system from these types of attacks. Integrating the countermeasures of these types of attacks will raise not only the cost, but also the power consumption.

The hardware security can be divided into different categories including hardware trojans [Dupuis2018pah], reverse engineering of hardware components and information which it stores, and side-channel attacks. Hardware trojans have become the new issues of highly-integrated System-on-Chip because IP cores are now designed by different vendors that can intentionally insert trojans which are set to be activated

when having enough conditions. For cryptographic devices, hardware trojans might be used to take out the secret keys. In addition, reverse engineering of hardware components can also be used to extract helpful information. Key stored in the memory or non-volatile elements might be extracted using this method. More importantly, side-channel attacks employ other types of information from the chip such as voltage, current or power consumption to attack the cryptography algorithm. This work focuses on side-channel attacks which can be used to mount key extraction attacks.

Side-channel information could be any forms of physical information emitted by the microchip. This physical information is one part of the physical characteristics of the chip. Two popular side-channels include power consumption of the chip [Kocher1999dpa] and electromagnetic emission (EM) [Agrawal2003tes]. Power consumption and EM captured from a device are related to the chips activities. When applied to cryptographic chips, this instantaneous power consumption or EM are directly related to the chip operations which contains the operations with the secret key. Therefore, power or EM traces can be extracted using the current state-of-the-art attacks.

The current semiconductor technologies use CMOS transistors to build the functional circuit. These transistors have different levels of power leakage during their operations. The leakage is related to the current state or the function that it performs and it is called the state dependent leakage which is directly related to the secret key of cryptographic algorithms. By measuring power consumption of the circuit, the secret key can be extracted if enough data is collected.

Power analysis attacks are divided into different categories including Simple Power Analysis (SPA) [Kocher1999dpa], Differential Power Analysis (DPA) [Kocher1999dpa], Correlation Power Analysis (CPA) [Brier2004cpa], and Template Attack [Chari2003ta]. SPA is a simple method which guesses the secret key directly from the power traces. SPA is often used for asymmetric cryptography when the serial operation is used. The secret key can be guessed directly from the power trace. However, in real hardware, the computations are often performed in parallel and there are many noises which leads to a low success rate of SPA in practice. In contrast, DPA and CPA use differential function and correlation estimation function respectively to evaluate a set of guesses. The guesses with the highest score might be the correct key. CPA is one of the most effective methods at the moment. For template attacks, the attacker must have full control of a sample device to create a template for the attacks. Numerous power traces must be collected, and the template attack will characterize the system. After characterization, the attack can be mounted with a few power traces. Template attacks need more preparation steps than DPA or CPA, but it is also a powerful method because the victim's secret key can be found with fewer power traces after characterization.

Figure 2.11 demonstrates the attack flow of Power Analysis Attacks. Instead of brute forcing the key, the attack can focus on a certain byte at a time. This reduces

the total number of key guesses. At first, an intermediate value in the algorithm is selected for attack. The intermediate value can be the output of any steps in the algorithm. After that, a hypothesis consisting of all the guesses can be calculated using a power model. After that, the hypothesis can be correlated and compared with the real power traces. The one which has the highest correlation can be the correct guess. This reduces the search space and can reduce the time to find the correct key. For a software implementation on 8-bit micro-controller, a key can be revealed using some hundreds of traces. Power model which is often used in the power analysis attack is Hamming Distance which is related to the activity of the key-dependent operation. Pearson Correlation function is often used as the statistical tool for this analysis.

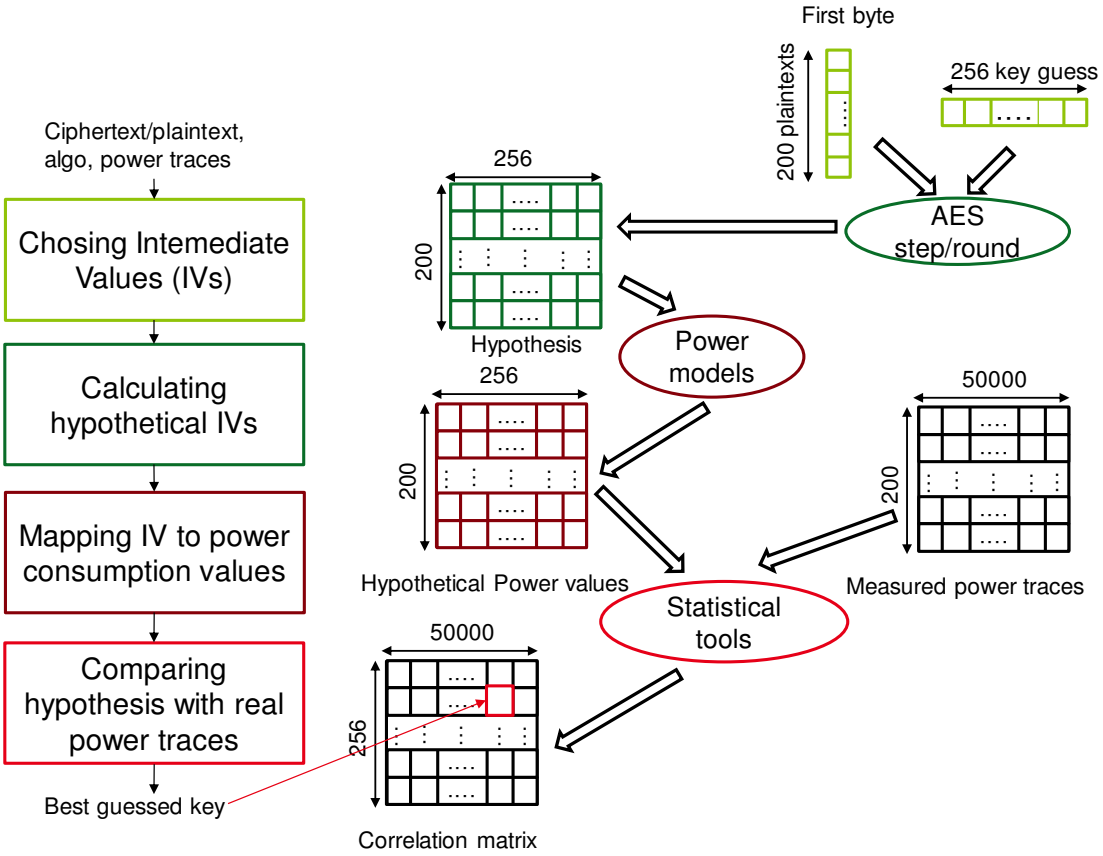


Figure 2.11: Power analysis attacks.

Furthermore, key-dependent leakage can be addressed faster using statistical tests such as Student T-Test [Student1908tpe] or a more general version – Welch T-Test [Welch1947tgo]. These tests perform the statistical analysis of the power traces to find the distinguishable data points in the power traces. The calculation of T-Test is faster than DPA and CPA. Therefore, the leakage can be found earlier [Schneider2015lam]. However, the leakage found by T-Test might not lead to key

recovery attacks [Schneider2015lam].

To generalize the testing methods for cryptographic devices, Test Vector Leakage Assessment (TVLA) [Goodwill2011atm] has been proposed as a generalized method to help designers identify the leakage. This method proposes procedures to evaluate the designs including the key generation for the test, the plaintext generation and the trace processing. The statistical test used is Welch T-Test. There are two kinds of TVLA tests including non-specific tests and specific tests. In non-specific tests, the encryption of the same inputs is performed repeatedly. Then, the captured power traces are divided into two groups. If the T-Test scores of the two groups are larger than 4.5 or smaller than -4.5 at the same point in time, the test fails and the design might leak important information which can be used to reveal the secret key using power analysis attacks. In specific tests, TVLA also defines a set of tests on different steps of the algorithms such as the statistical test of the S-Box output (SOUT), the inputs and outputs of each round (RIRO), the specific bytes of each round (ROUT_BYTE0) and so on. There is a total of 896 tests per round. The power traces are captured with a predefined key and data with the output of the current test being used as the input of the next test. The power traces are also divided into two groups as in the non-specific test. The test fails when the T-Test scores of the two groups are larger than 4.5 or smaller than -4.5 at the same point in time. The design is not qualified when at least one of the tests fails. T-Test can be performed much faster than mounting DPA and CPA attacks. It can also be used to evaluate the high order leakages such as the work in [Schneider2015lam]. In addition, TVLA can also be linked to the attack success rate [Roy2016lmi].

This work performed CPA and TVLA on the power estimated traces of the crypto-accelerator designs to ensure that the power optimization techniques applied to the crypto-accelerator do not introduce new leakages. There are other statistical methods, which are alternatives to TVLA, such as Analysis of Variant (ANOVA) or Normalized Inter-Class Variant (NICV) [Bhasin2014nicv, Moradi2014dhl]. However, these methods are not considered in this work.

2.7 Conclusion

Security concerns for IoT systems are rising rapidly along with their vast range of applications. Therefore, implementing security mechanisms for IoT devices, especially highly-constrained devices, need careful consideration. For these devices, a new class of cryptography algorithms called lightweight cryptography has been developed to save the hardware cost and reduce power consumption. However, they might have barely acceptable security levels and a margin of lower energy consumption. Ultra-low-power operations are crucial to many applications which do not have a stable power supply such as the power harvesting devices. Lightweight cryptography algorithms standardized by ISO/IEC for different applications are available, but they

have not been used yet for IoT applications.

Moreover, new IoT proposals have been adopting AES as the main security mechanism. Therefore, optimization for AES in terms of area, throughput and especially power/energy consumption is becoming increasingly important. Current implementations of AES for low power consumption is focusing on 8-bit datapath designs with one S-box which reduces the system throughput and increase the energy consumption. 8-bit datapath with one S-box has small hardware area because of the small area footprint of one S-box, but they need extra registers to store the intermediate results for the MixColumns calculations. In addition, different implementations of S-boxes have different hardware area and different power consumption. This opens the need for optimizing the S-boxes for the trade-off among hardware area, power consumption, and energy consumption.

Furthermore, different applications are subject to various security requirements and power/energy budgets. Therefore, a configurable solution is needed to adapt to various applications. In particular, lightweight block ciphers and conventional block ciphers can be combined to create a flexible solution with low power consumption. In addition, the trade-offs among area, throughput and power/energy consumption are challenging to achieve at the same times. It even becomes more challenging when flexibility and configurability are added.

Despite the low power consumption of these implementations, their fixed structures cannot adapt to new standards and mitigate new threats. New capabilities are added, while new attacks and threats are discovered. Therefore, flexibility and configurability of security primitives need attention. In-Memory Computing and Near-Memory Computing are innovative methods to implement flexible security functions. Because of the serial operations of the memory, flexibility and configurability are traded off with throughput and power consumption.

In addition to that, hardware security which uses side-channel information as an attack vector is bringing more and more threats to new IoT devices. Side-channel attacks can be used to reveal the secret key of the system. Countermeasures to these attacks are costly and often not included in the design for ultra-low-power devices.

Based on the current state-of-the-art of hardware implementations of various block ciphers, this work proposes a low-power implementation of two standardized algorithms which can be used for ultra-low-power IoT devices. The first one, AES, is a conventional algorithm widely used to secure Internet applications with high levels of security. The other is PRESENT, a new lightweight algorithm which uses hardware constructs to reduce the hardware area and the power consumption. In the next step, this work combines the two modules, AES and PRESENT, into a configurable crypto-accelerator with various key sizes, block sizes, and security levels. Not only the power consumption is optimized, but the security evaluation using the current state-of-the-art methods are also applied to the proposed design to ensure that low-power optimization does not introduce new information leakages. Finally, this work

explores the configurability, the flexibility and the feasibility of different block cipher algorithms using In-Memory Computing. The next chapter will present the work on the power optimization of block ciphers including AES and PRESENT for IoT and our configurable crypto-accelerator in SNACk testchip.

Chapter 3

Proposed crypto-accelerator for ultra-low-power IoT

New IoT applications such as implant and wearable appliance, healthcare monitoring, and environment monitoring need to be implemented in ultra-low-power devices with small hardware area. Furthermore, secret data might be collected, processed and transmitted in the IoT network. Therefore, it is crucial to equip IoT applications with security functions to protect confidentiality and authentication. However, security functions may reduce the system throughput and increase the power consumption, because extra functions are required for their execution. This brings into play the requirements to optimize the security functions for throughput, cost, and power consumption.

In addition, different applications are characterized with diverse security requirements, depending on the type of secret data they manipulate. Applications which require long-term security should use strong cryptography algorithms such as AES with 256-bit keys, while lightweight applications can use lightweight cryptography to reduce the power consumption. Security primitives with different configurations, which can adapt to various applications' needs, can provide the flexibility for application development.

Ultra-low-power consumption solutions can be provided by the hardware implementation of security functions using block ciphers and lightweight block ciphers with serial architectures. Serial processing helps improve the power consumption. However, it increases the overall system latency and energy consumption. Therefore, trade-offs among security levels, throughput, hardware costs and power/energy consumption should be studied thoroughly.

The recent IoT proposals have been using Advanced Encryption Standard (AES) as the main security mechanism to protect the data confidentiality because AES has been proved to provide long-term security. Therefore, optimizations for AES are important not only for IoT applications but also for a wide range of other applications

which use AES. However, AES has been considered to be a conventional cryptography algorithm with high power consumption even when it is implemented on hardware. On the other hands, for ultra-low-power applications, lightweight cryptography such as PRESENT might also be applied to further reduce the power consumption by using hardware friendly constructs, but this reduction has to be traded off with a low security level and throughput.

In addition, optimizations, for instance, power optimization might cause hardware security issues. Attacks using side-channel information especially power analysis attacks can be mounted easily. Therefore, security evaluation for the optimized security module needs careful consideration. Methods such as Correlational Power Analysis and Test Vector Leakage Assessment might be used to evaluate the possibility to mount the key recovery attacks and to address the different leakages of the hardware modules.

In this chapter, a multi-standard hardware crypto-accelerator with multiple security levels covering from lightweight security up to very long-term security with optimization for power/energy consumption is proposed. It contains two standardized encryption algorithms: AES [FIPS-197] and PRESENT [Bogdanov2007pau]. AES, which is widely used in current IoT proposals, provides high levels of security with high throughput; while PRESENT, a lightweight block cipher, encrypts data at extremely low power consumption. This crypto-accelerator has a 32-bit data interface with configurable parameters. It is up to the application to decide which algorithm, with which key size and number of rounds to use. The AES module was designed with 32-bit datapath to save the implementation area while the PRESENT module has 64-bit datapath to maximize its throughput. Among other means, the hardware area of the two designs is saved by doing key expansion on-the-fly. AES power consumption is further optimized by a low power implementation of the substitution boxes (S-boxes), and by gating the inputs of S-boxes in the key expansion when they are not used to reduce the switching activities. This crypto-accelerator was implemented in the $28nm$ FD-SOI technology of STMicroelectronics with $11K$ gates equivalent (GEs) in total and throughput up to $170Mbps$. At $10MHz@0.6V@25^oC$, our design provides a throughput between $17Mbps$ and $28Mbps$ while consuming less than $20\mu W$ on average, and its energy per bit is less than $1pJ/bit$. Furthermore, this proposal does not focus on countermeasures to side-channel attacks but the security evaluation, using Correlation Power Analysis (CPA) and Test Vector Leakage Assessment (TVLA), has been performed on the proposed design. The key recovery attacks using CPA and the test score in TVLA show that the proposal has equivalent information leakage when compared with an unprotected reference design on OpenCores. Thus, this accelerator is suitable for a wide range of ultra-low-power and ultra-low-energy applications in IoT sensor nodes.

This chapter is outlined as follows. Section 3.1 provides a short introduction to AES and PRESENT in the proposal. Section 3.2 presents the architecture of the

proposed AES including all the optimizations performed on AES. PRESENT architecture is shown in Section 3.3. After that, the two block ciphers are combined into a block cipher module with 32-bit data interfaces and implemented in SNACk testchip. The estimation and measurement results of SNACk testchip are demonstrated in Section 3.4. Then, the security evaluation based on the power trace generated from the post-signoff netlist is presented in Section 3.5. This section also includes the detail explanations of the trace processing framework to prepare the power traces for the evaluation. Finally, some conclusions and perspectives are raised in Section 3.6.

3.1 Introduction

IoT applications might require various security profiles with different power/energy budgets. A configurable crypto-accelerators with different power/energy profiles can provide a flexible solution for a wide range of IoT applications. This chapter presents a crypto-accelerator with a widely used cryptography standard core – AES and a lightweight cryptography core – PRESENT. Two IP cores are implemented into the cryptography kernel in SNACk testchip for evaluation. AES is selected because it provides long-term security and has broadly been used as the main security mechanism for new IoT proposals, even though it occupies a large hardware area and has high power consumption than lightweight algorithms. Meanwhile, PRESENT is one of the new lightweight cryptography algorithms which use the hardware construct to reduce the hardware cost and minimize the power consumption.

The proposed AES encryption core uses 32-bit datapath supporting multiple security levels from 128-bit key up to 256-bit key. In addition, power optimization is applied at the hardware architecture level to minimize the power consumption. Hardware area is also optimized to reduce the hardware cost. The datapath of our proposed AES was designed to minimize the switching activities, consequently to save power consumption. In the AES algorithm, in each round, the certain order of transformations can be changed without the impact on the output of the algorithm. For example, the order of ShiftRows and SubBytes can be changed without changing the output of the algorithm. For the design of AES, we rearranged the order of these steps in order to achieve the best efficiency in terms of power, energy and throughput. Our AES encryption architecture supports all the key sizes specified in AES standard [FIPS-197].

At the same time, our proposed PRESENT encryption core uses a full parallel architecture with 64-bit datapath to increase throughput since our PRESENT core is a round-based encryption core which requires 32 rounds to do an encryption. Even in this configuration, PRESENT still shows some advantages over AES with small hardware size and low power consumption. However, PRESENT in software has some drawbacks because of its hardware-friendly constructs in bit-based operations.

In the following sections, the full hardware architectures of the proposed AES and

PRESENT will be presented. Two algorithms are integrated into a crypto-accelerator with a 32-bit data interface. The crypto-accelerator is implemented using FD-SOI $28nm$ technology in SNACk testchip. The implementation results and the power consumption measurement will be discussed in the following sections.

3.2 Proposed AES architecture

AES supports multiple key sizes to allow multiple levels of security ranging from midterm security with 128-bit keys to long-term one with 256-bit keys. Furthermore, in a non-standard way, the reduced rounds of AES can also be used to increase the throughput and reduce energy consumption. For examples, the best theoretical attack on a 5-round AES requires $2^{22.25}$ chosen plaintexts, a memory of 2^{20} blocks of 128 bits, and computation time of $2^{22.5}$ encryption [Achiya2018ikr]. A lightweight application, which does not use up to this bound, can use the reduced-round AES to minimize the energy consumption. Consequently, this work implements a configurable AES encryption module with different configurations of key sizes and also support reduced-round AES. The proposed architecture is presented in Figure 3.1. The encryption path includes four parts: a state register, 4 S-boxes, a MixColumn, and an output register which also acts as a temporary register to store intermediate results. The key expansion consists of two key registers and a key transformation module to support all key sizes specified in AES. This design is a 32-bit datapath architecture which means the input data and the input key are divided into 32-bit chunks. Each pair of 32-bit data and 32-bit key is loaded together. This takes 4 cycles to load the 128-bit key and 128-bit data and XOR them into the state register. For 192-bit keys and 256-bit keys, after the first 128 bits are loaded, the encryption is started while the other bits of the key are continuously loaded to maximize the throughput. There are two feedback paths, one in the key expansion and the other in the encryption path. The state register needs to be updated every four cycles with new 128-bit data while the previously expanded word is sent back to the key registers to generate the new expanded key. The details of the optimizations in our proposed architecture are presented in the next subsections.

3.2.1 32-bit datapath optimizations

To reduce area and power consumption in the datapath, this work minimizes the number of flip-flops and control logic in the datapath by using shift registers with a special organization. Shift registers help to simplify loading data and loading key steps. 32 bits of both plaintext and key are loaded at the same time into the state register and the key register by using shift operations. By minimizing the number of flip-flops, this work also reduces the number of clock buffers and power consumption of the clock tree because clock buffers in the clock tree consume a large amount of

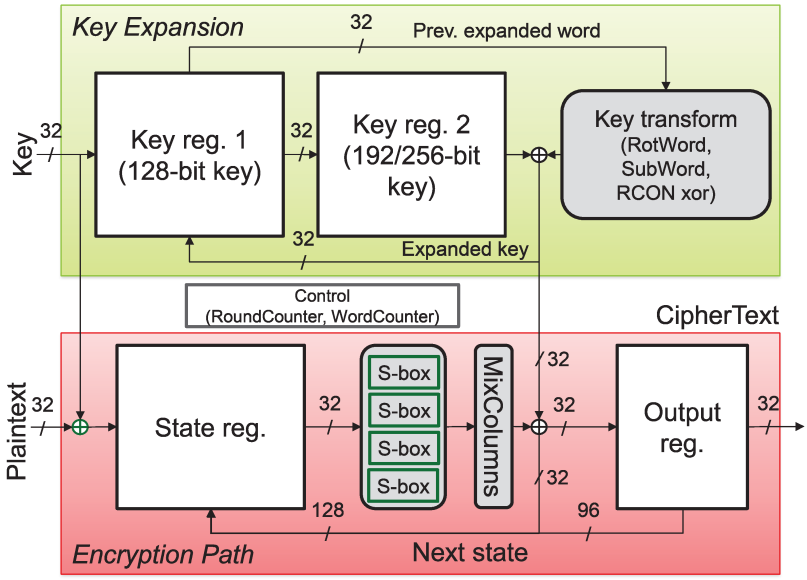


Figure 3.1: The proposed AES architecture.

power, up to 40% of the total chip's power [Lu2005rpf]. A further optimization is to select S-boxes with minimal power dissipation.

Figure 3.2 illustrates the organization of the proposed state register. The state register is organized in order for the encryption to be done by shifting the data 32 bits in each clock cycle after loading the input data and the input key. The state register consists of sixteen 8-bit registers (forming a "state matrix") which are further divided into four 4-stage shift registers. AES standard specifies that ShiftRow is a permutation operation on the rows of the state matrix while MixColum is an operation on the columns. However, in this design, based on ShiftRow specification, this work completely eliminates ShiftRows by selecting the diagonal of the state matrix (from lower-left corner to upper-right corner). The output of the state register after each shift operation is one column of the state matrix after ShiftRow. This reduces the control logic for the state register and completely removes the logic for ShiftRow steps. In the proposed datapath, in contrast with 8-bit architectures, MixColum is designed as pure combinational logic to reduce the number of flip-flops. Thanks to this structure, the state register's contents will be updated by next state data which are the contents of the output register concatenated with 4 last bytes of the round operation every 4 cycles (or after each round finishes) as described in Figure 3.3. Consequently, this work saves a 32-bit register because this architecture needs to store only 3×4-byte temporary data from the encryption path in the output register, while the last 32-bit data are directly written back into the state register. The output register is a simple 4×3-stage shift register to save area and power.

In between the state register and the output register, there are four S-boxes followed by the MixColums to enable processing four bytes in each clock cycle. The temporary results are stored in the output register. When the encryption finishes, the

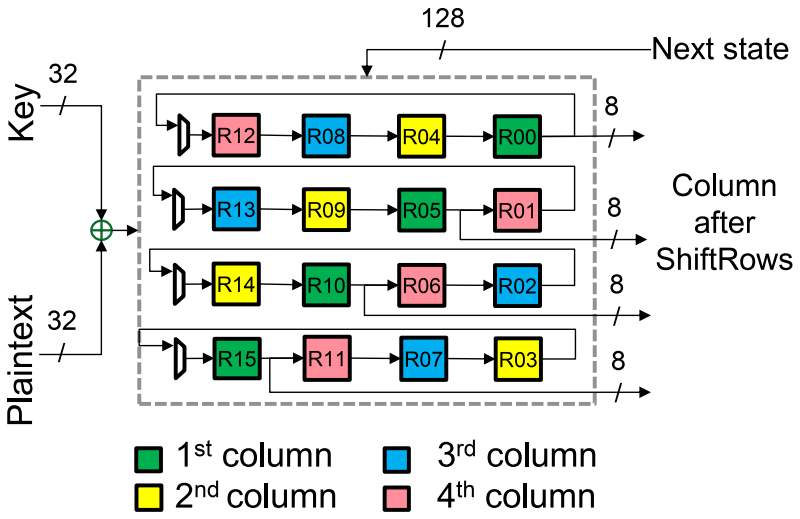


Figure 3.2: Our proposed state register.

results are written out from the output register. In the 128-bit key configuration, AES encryption module needs 10 rounds, which leads to 40 cycles to finish the encryption for a 128-bit block of data. The total number of cycles to encrypt a block in this architecture is 44 cycles. For other key configurations, our architecture needs 52 cycles and 60 cycles to encrypt a data block for 192-bit key mode and 256-bit key mode, respectively.

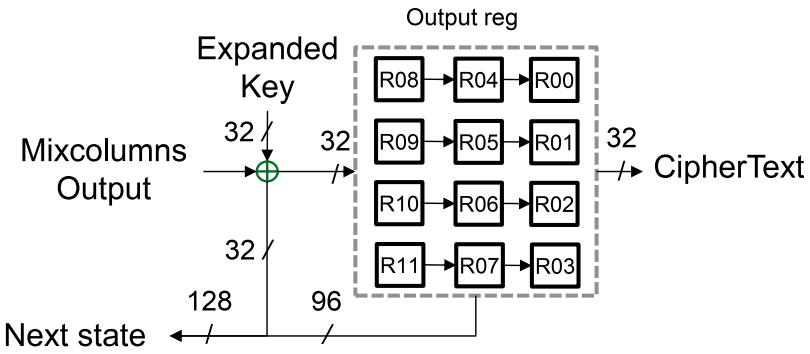


Figure 3.3: Our proposed output register.

Clock gating technique is applied on the stage register and the output register separately to save the dynamic power consumption. For examples, in data loading state, the clock to the output register is disabled to save power because there are no valid data to the output register. Furthermore, when in the inactive state, the output of these registers is not changed, which means that there is no activity in the encryption path. The power estimation results using the post-placement-and-routing netlist show that even in the highest throughput mode (44 cycles/encryption for 128-bit key mode) the applied clock gating technique can save more than 13% of power.

Certainly, with smaller throughput, the clock gating technique can even save much more power consumption.

3.2.2 Substitution box (S-box)

The substitution box (S-box) has a significant impact on area and power consumption of the AES design. This architecture chooses S-box implementation for achieving the lowest power consumption. S-boxes may occupy up to 60% of the total cell area, and consume about 10%- 30% of the total power consumption [Hamalainen2006dai]. The smallest implementation of S-boxes has up to present been attained by Canright [Canright2005AVC]. Canright S-box demonstrates an optimized area (292 gates/S-box) but needs more power/energy consumption because it creates more activities, especially in architectures with 8 S-boxes. The most popular and straight-forward S-box implementation is the LUT-based S-box. LUT-based S-box is bigger in terms of area (434 gates/S-box) but consumes less power/energy than Canright S-box. The most efficient S-box in terms of power consumption is Decode-Switch-Encode (DSE) S-box; however, it occupies a larger area. DSE S-box can be further optimized for power consumption using the structure proposed by Bertoni *et al.* in [Bertoni2004PAS] and described in Figure 3.4. The idea is to use a onehot decoder to convert S-box inputs into onehot representation. In this scheme, the input is represented using a group of bits where there is only one bit '1' and all the others '0'. The non-linear operations are performed using wire permutation as in lightweight cryptography algorithms. After that, the S-box output in onehot encoding is converted back into the original field.

Decode-Switch-Encode S-Box can reduce the power consumption because it minimizes the activity inside the S-box circuit. After decoding state, only one signal changes its value to go to the encoding state. Most of the area lost is because of the size of encoder and decoder circuits. This optimization leads to 10% power reduction to the whole design. The synthesized DSE S-Box in this work has the size of 466 GEs/S-box which is 7% increase in size in comparison with LUT-based S-Box or 1.6 times the size of the smallest S-boxes. The S-Boxes in our design consume only 10% of the total power consumption.

3.2.3 Key expansion optimizations

The key expansion applys the same mechanism as in the encryption path with further optimizations for S-boxes and loading data into the key registers for different key sizes. Also, the power consumption is saved by masking the S-Box inputs with constant values when not being used to save the dynamic power consumption. To save the hardware area and to improve the system throughput, the expanded key is calculated on-the-fly and fed back directly to the key registers and the encryption path. To support three sizes of keys in AES, the key expansion module consists of

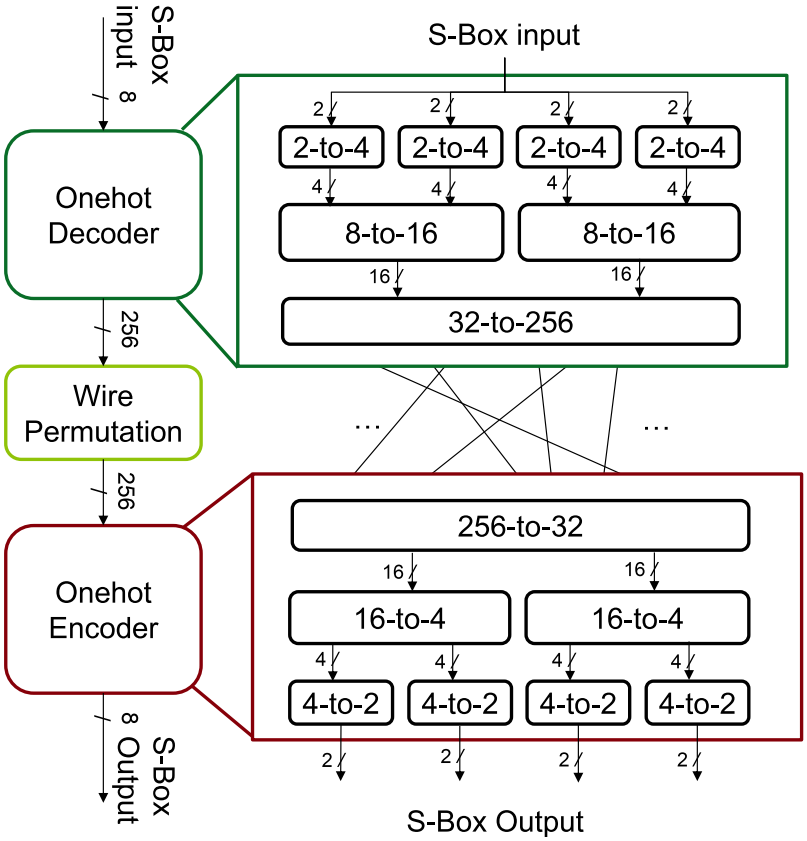


Figure 3.4: Our Decode-Switch-Encode (DSE) S-Box.

two 4×4-stage shift registers which support storing and expanding of 128-bit keys, 192-bit keys and 256-bit keys. The key transform module includes four S-boxes, and an XOR to do key expansions for all key modes.

The structure of the two registers is presented in Figure 3.5. For 128-bit key mode, only the first shift register is used, at the same time, the clock signal to the second shift register is disabled to save power. For 192-bit key mode, the first shift registers and a half of the second shift register are used, while for 256-bit key mode, both shift registers are used. The last expanded word of the key expansion output is sent back into the first key register to continue generating the round key. Depending on the key size, the last word may need to be transformed using RotWord, SubWord and XOR with RCON, a round constant, before being added with other key words. In 128-bit key mode, these three operations are applied to the last word every 4 clock cycles while in 192-bit key mode and 256-bit key mode, they are applied every 6 clock cycles and 8 clock cycles, respectively. 256-bit key mode needs one additional SubWord in the middle of 8 clock cycles. The second key register generates two different key modes which are 192-bit key mode and 256-bit key mode. In 128-bit key mode, the second key register is disabled while the output of the first key register is selected to XOR with the output from the key transform module. In 192-bit key

mode, the second stage in the second key register is chosen. Finally, in 256-bit key mode, it is the output of the last stage in the second key register.

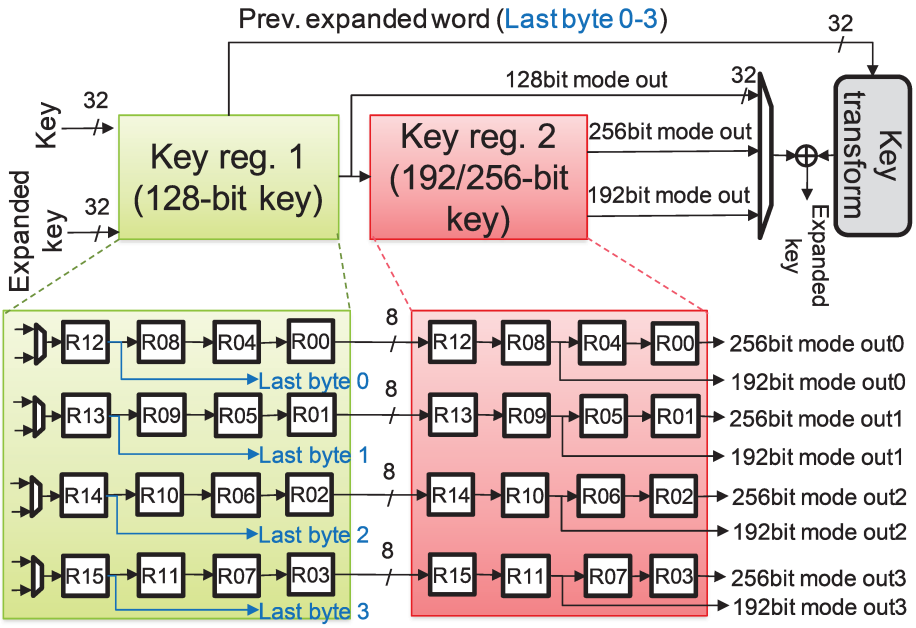


Figure 3.5: Key registers.

The key transform module is shown in Figure 3.6. The input of the key transform module is the last word from key registers. Based on the key expansion specification, 4 S-boxes are used one cycle among four cycles of a round in 128-bit key mode and 256-bit key mode while in 192-bit key mode; the 4 S-boxes are used every six cycles. Furthermore, during the loading of the key into the key register, these S-boxes are not used. During the idle time, the inputs to these S-boxes are gated by a mask to save the dynamic power. These S-boxes are enabled for the first cycle of a round for 128-bit key and 256-bit key and every 6 cycles for 192-bit key. After that, they remain inactive. This leads to 30% reduction in power consumption of the S-boxes in the key expansion. RotWord step can be removed because it exchanges the position of bytes in a 32-bit signal. The RCON constant can be modeled as a shift register as in [Mathew20153m1] but in our architecture, we designed it as a lookup table of 10 constant values to minimize the number of registers in order to reduce the power consumption and minimize the clock network. The XOR of the RCON with the output of the S-boxes after RotWord is minimized by XORing only the necessary bits.

The 32-bit output of the key expansion is sent directly to the encryption path to be XORed in the AddRoundKey step. The clock gating technique is also applied in the key expansion to save power consumption. During the idle state, the key register and the S-boxes will not create any activities.

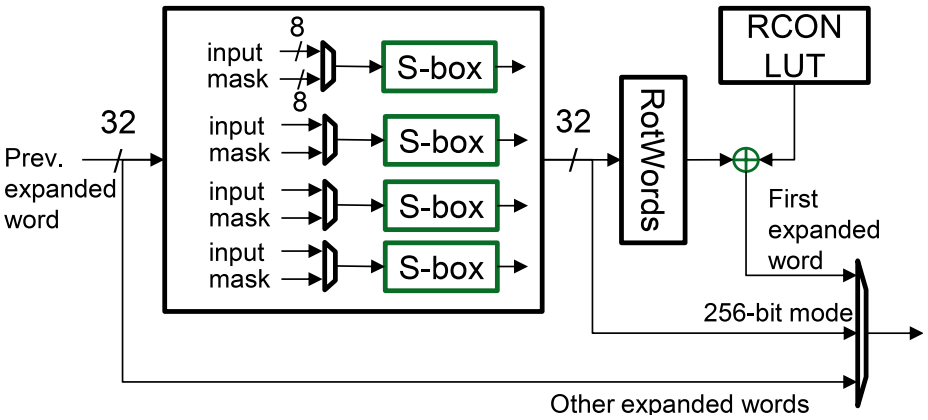


Figure 3.6: Key transform.

3.3 Proposed PRESENT architecture

Similar to AES, PRESENT has been designed with multiple security levels supporting from decent security levels with 80-bit keys to the midterm security of 128-bit keys. The reduced-round version of PRESENT [Cho2009lco] can be used to improve throughput and minimize power consumption. Therefore, the proposed PRESENT module is designed with different modes of keys and various number of rounds. The proposed PRESENT architecture is presented in Figure 3.7. This architecture is a straight-forward implementation from PRESENT specification which supports both 80-bit key and 128-bit key. Therefore, in this work, the two key modes are combined in a configurable module with various number of rounds to allow multiple power/energy consumption profiles. The input plaintext and key are 32-bit data interface. Therefore, loading 64-bit blocks will take two clock cycles, whereas loading keys needs extra clock cycles: 3-clock cycles for 80-bit keys and 4 clock cycles for 128-bit keys. Similar to AES architecture, keys and input data blocks are loaded together so that the addition of key and data can be done immediately. To reduce the hardware area, the intermediate results are stored directly back to the state registers. The key extension also uses shift registers to store the expanded key. PRESENT use only 4-bit S-Boxes, therefore, they are designed as combinational logic. The permutation is the wire permutation. As a result, they cost fewer logic gates. All the control logics are based on a single round counter and the key configuration.

To optimize the power consumption, the shift registers are clock gated using an enable signal. When the shift registers in the idle state, the clock is disconnected from the clock network to save power consumption. The S-boxes are optimized using combinational logic. The S-Boxes are used in all steps; therefore, they are not masked as in the case of AES. With this configuration, the proposed PRESENT core takes 37 clock cycles and 38 clock cycles to finish encrypting one 64-bit data block for 80-bit keys and 128-bit key, respectively. In comparison with the proposed AES architecture, PRESENT core uses 2.5 times fewer registers than the ones in AES

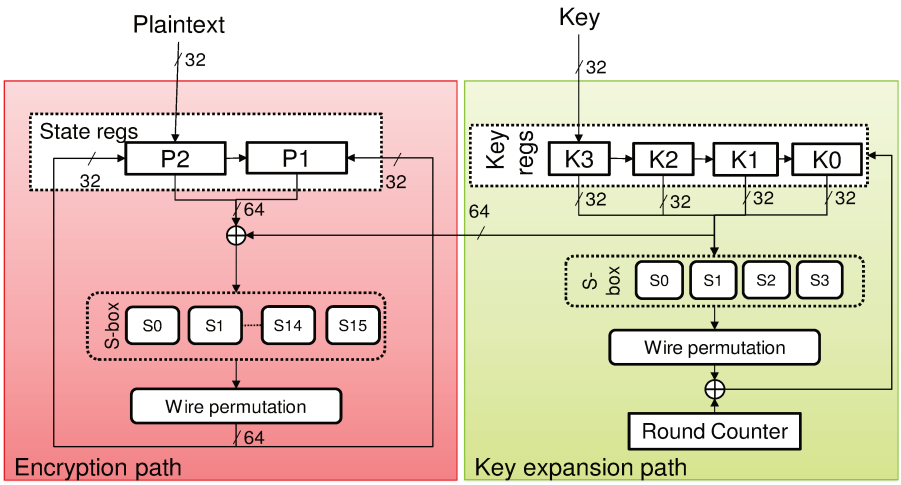


Figure 3.7: Proposed PRESENT architecture.

cores.

3.4 Estimation and measurement results of SNACk testchip

The proposed AES architecture and a lightweight cryptography algorithm PRESENT are modeled in VHDL, synthesized using Synopsys DC Compiler and fully implemented using Cadence Innovus into the testchip SNACk using ST FD-SOI 28nm technology. The maximum target frequency is set to 60MHz which provides the maximum throughput of 170$Mbps$ and 106$Mbps$ for AES encryption core and PRESENT encryption core, respectively. This throughput meets the demand for medium and high throughput IoT applications. AES encryption module and PRESENT encryption module are combined into the block cipher module in SNACk testchip for comparison purpose. The power consumption at different corner cases is estimated using the post-signoff extraction. The following subsections present our power estimation results and measurements on SNACk chip and the security evaluation that we implemented using Synopsys PrimeTime Power.

3.4.1 Configuration and test environment of SNACk

Figure 3.8 shows the interface of the encryption module in the SNACk testchip. It contains the test environment for the proposed AES encryption architecture and also a lightweight cryptography algorithm PRESENT. It has a 32-bit data interface with the possibility of selecting different key sizes and the cipher type between AES encryption core and PRESENT encryption core. AES encryption core supports all

the encryption modes specified in AES standard including 128-bit key, 192-bit key and 256-bit key. PRESENT encryption core with the same interface contains two modes: 80-bit keys and 128-bit keys. The two designs were implemented using the same technology.

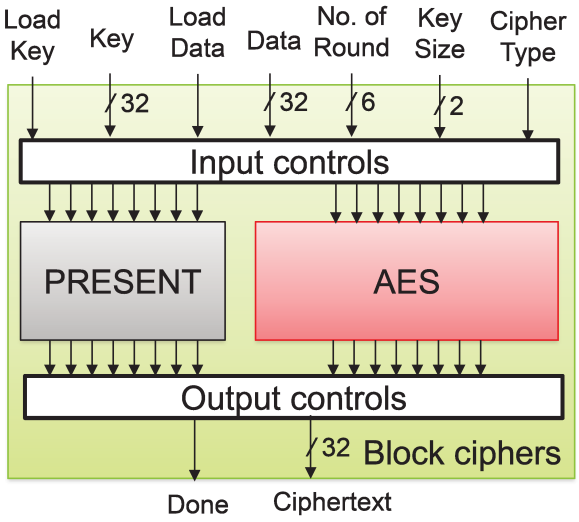


Figure 3.8: The block cipher modules in SNACk testchip.

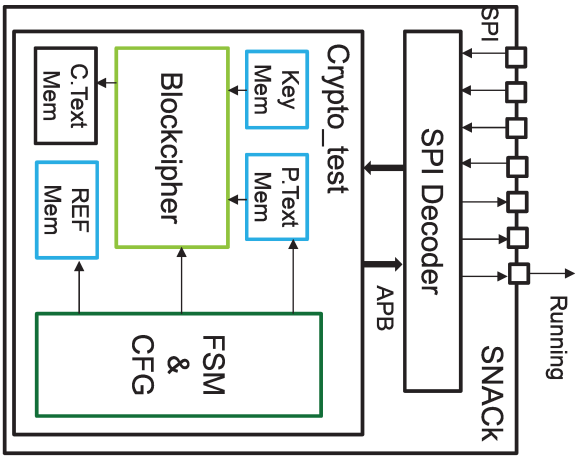


Figure 3.9: Blockcipher module in SNACk testchip.

The test environment for block cipher modules in SNACk chip is presented in Figure 3.9. The plaintext and the key are loaded from the host through SPI interface. Inside SNACk chip, there is an SPI decoder with the APB-like interface to write the test data into the correct memories including the configuration registers, the key memory, the plaintext memory, and the reference memory. After loading all necessary data, the encryption test is run by activating the control finite state machine (FSM). If the encryption is done correctly, the running signal will toggle. The encryption process continues running repeatedly until the control finite state machine receives

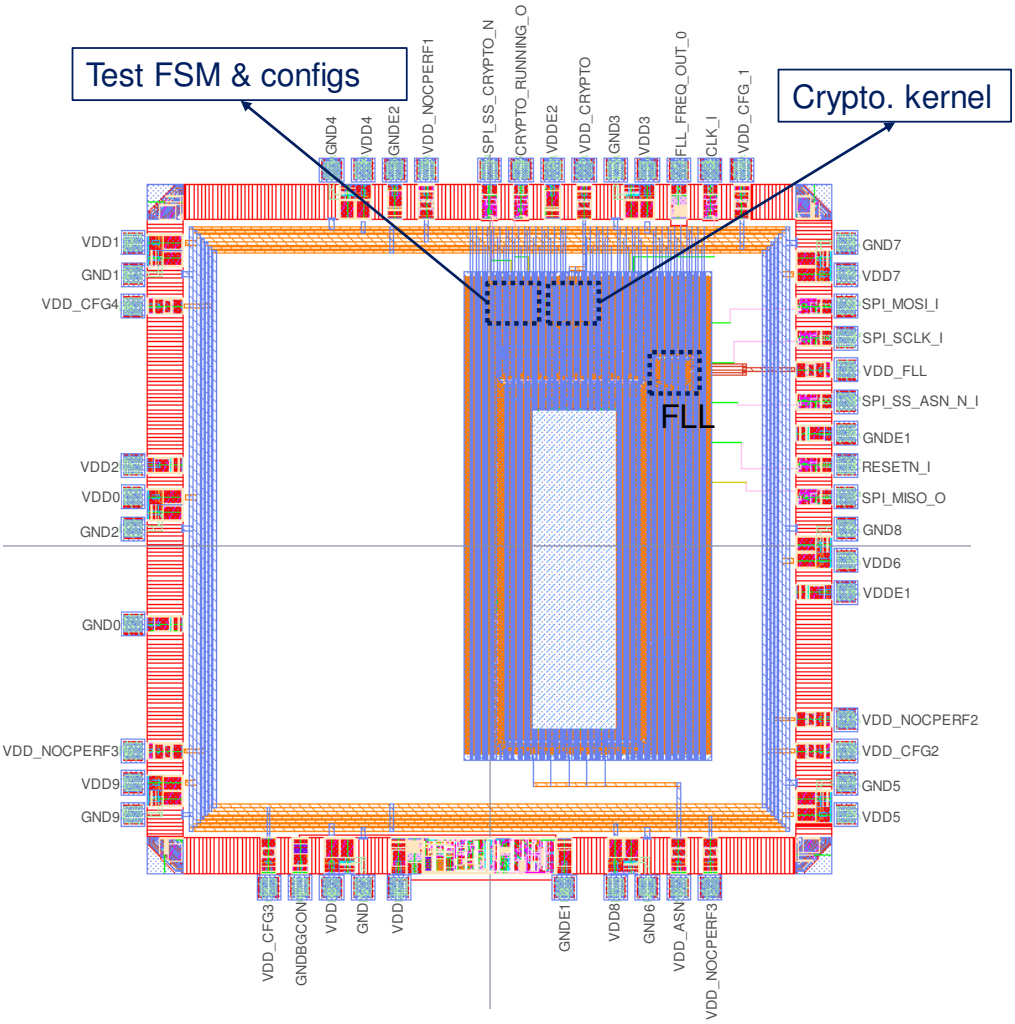


Figure 3.10: SNACk testchip's layout.

the stop signal through the SPI interface. The final layout of SNACk testchip is shown in Figure 3.10. An on-chip Frequency-Locked Loop (FLL) is used to generate different frequencies for the test. The FLL is also programmed through the SPI interface. All the power estimation results in the next subsection are obtained using this test configuration using the post-signoff extraction of the layout in Figure 3.10.

3.4.2 Power estimation results

To evaluate the performance and the power consumption of the proposed architectures using the test environment of SNACk chip, two encryption cores are tested with different key lengths at different supply voltages and different operating frequencies. The same key and the plaintexts are sent to each encryption module. Concurrently, the activity of the post-signoff timing simulation for each encryption module is cap-

tured for the whole encryption period. Then, the activity data are used to do power estimation in PrimeTime with FD-SOI $28nm$ technology libraries provided by STMicroelectronics. The technology libraries are characterized for the supply voltage from $0.6V$ to $1.3V$ for different working conditions. Figure 3.11 and Figure 3.12 illustrate the leakage power and the dynamic power of different encryption modes at $10MHz$ with the supply voltage ranging from $0.6V$ up to $1.3V$ at different corners at $125°C$.

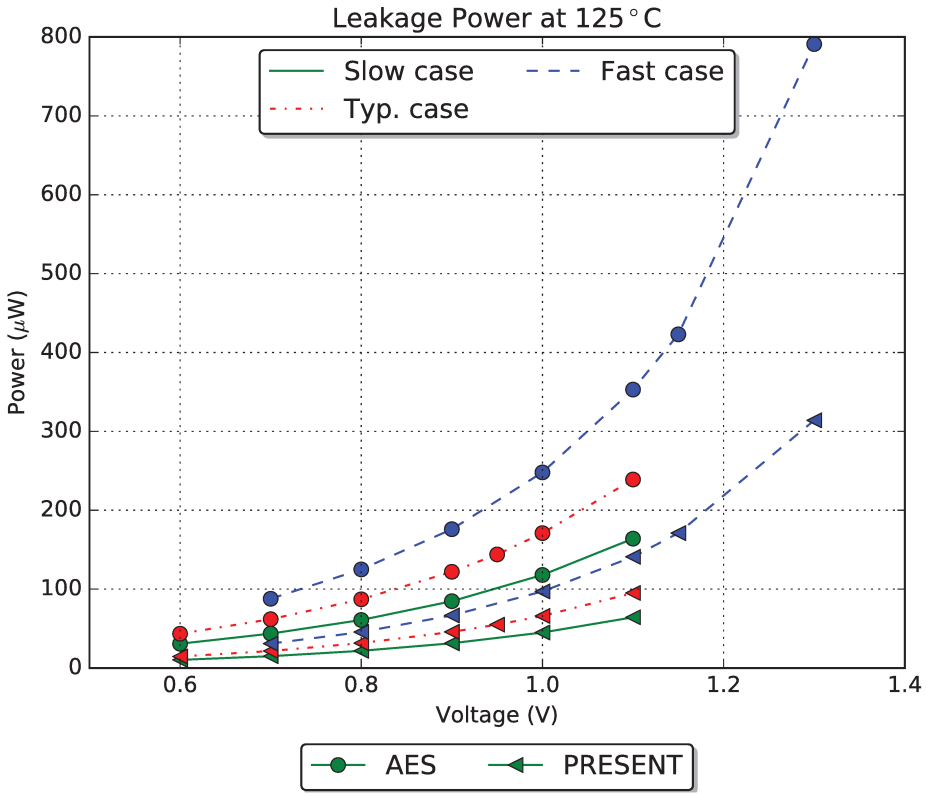


Figure 3.11: Estimated leakage power at $10MHz$ at different supply voltages at different corners.

Obviously, the worst case in terms of power consumption is the fast corner. It is also clear that there are different leakage powers at different corners while dynamic powers stay unchanged across different corners. The leakage powers increase significantly when the supply voltage is raised, especially in the fast corner. Within the same algorithm, the leakage power has minor differences for different key sizes; however, the leakage power of AES module is from 2.5 to 3 times higher than the one of PRESENT module. This is corresponding to the difference in the areas of two modules. AES module occupies 3.6 times more area than PRESENT module.

In terms of dynamic power, because of our optimization for different configuration by using separated clock gating for different keys, AES module with 128-bit key has 20% less dynamic power than AES module with 192-bit key and 256-bit key while the difference between AES 192-bit key and AES 256-bit key is a small margin. The

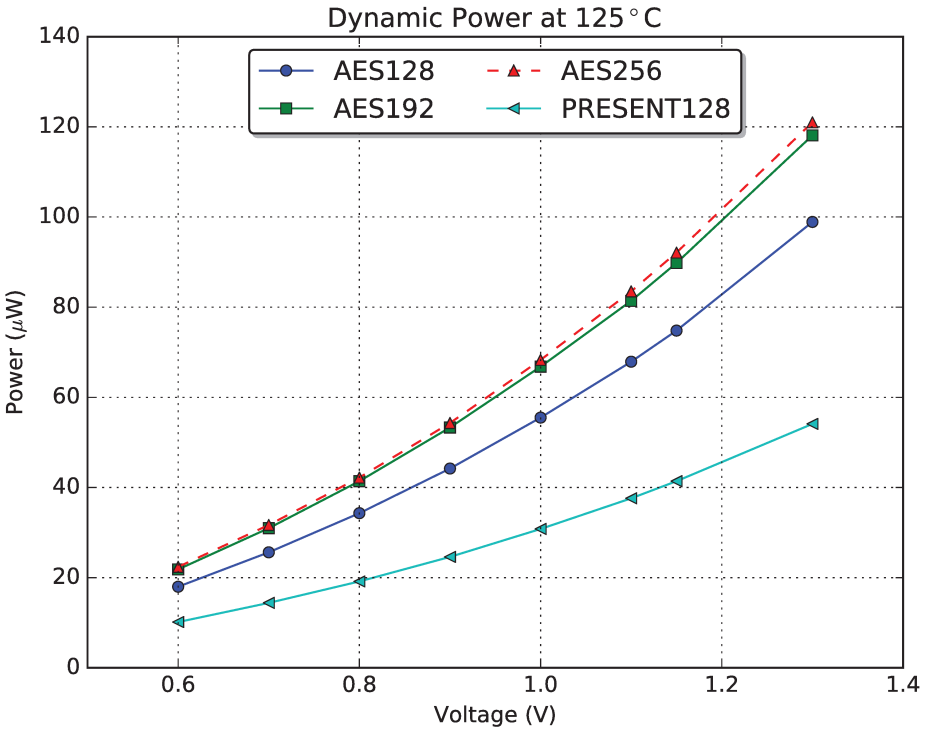


Figure 3.12: Estimated dynamic power at $10MHz$ at different supply voltages.

difference among three corners tested is small. The power consumption decreases gradually when we decrease the supply voltage. The best case in our power estimation results is at $0.6V$ where the leakage in the different key configurations for the two algorithms is close to each other. At the supply voltage of $0.6V$ at typical corner in the worst case of power consumption (at $125°C$), AES module consumes the power from $61.5\mu W$ to $65.6\mu W$ in total and PRESENT module consumes the power of about $24\mu W$; while in the typical case at $25°C$, our AES module and our PRESENT module consume only less than $20\mu W$ and $12\mu W$, respectively.

Furthermore, Figure 3.13 shows our estimation of energy per bit for the two designs. It is clear that our design can achieve extremely low energy per bit at $0.6V$. In typical case at $25°C$, our AES module achieves $0.65pJ/bit$, $0.78pJ/bit$ and $0.81pJ/bit$ for 128-bit key, 192-bit key, and 256-bit key, respectively. The energy per bit of different key configurations in AES module varies because different key modes require a different number of cycles to finish the encryption. In SNACk testchip, 128-bit key AES needs 45 cycles to finish one encryption; 192-bit key AES needs 53 cycles while 256-bit key AES needs 61 cycles. The lightweight algorithm PRESENT consumes nearly the same amount of energy per bit as AES because PRESENT needs 74 cycles to finish the encryption of 128-bit data in 128-bit key mode. In the worst case at $125°C$, our AES consumes less than $3pJ/bit$ while PRESENT needs less than $2pJ/bit$. The difference in energy per bit in the two working conditions is caused by the different leakage power contributed to the total power consumption.

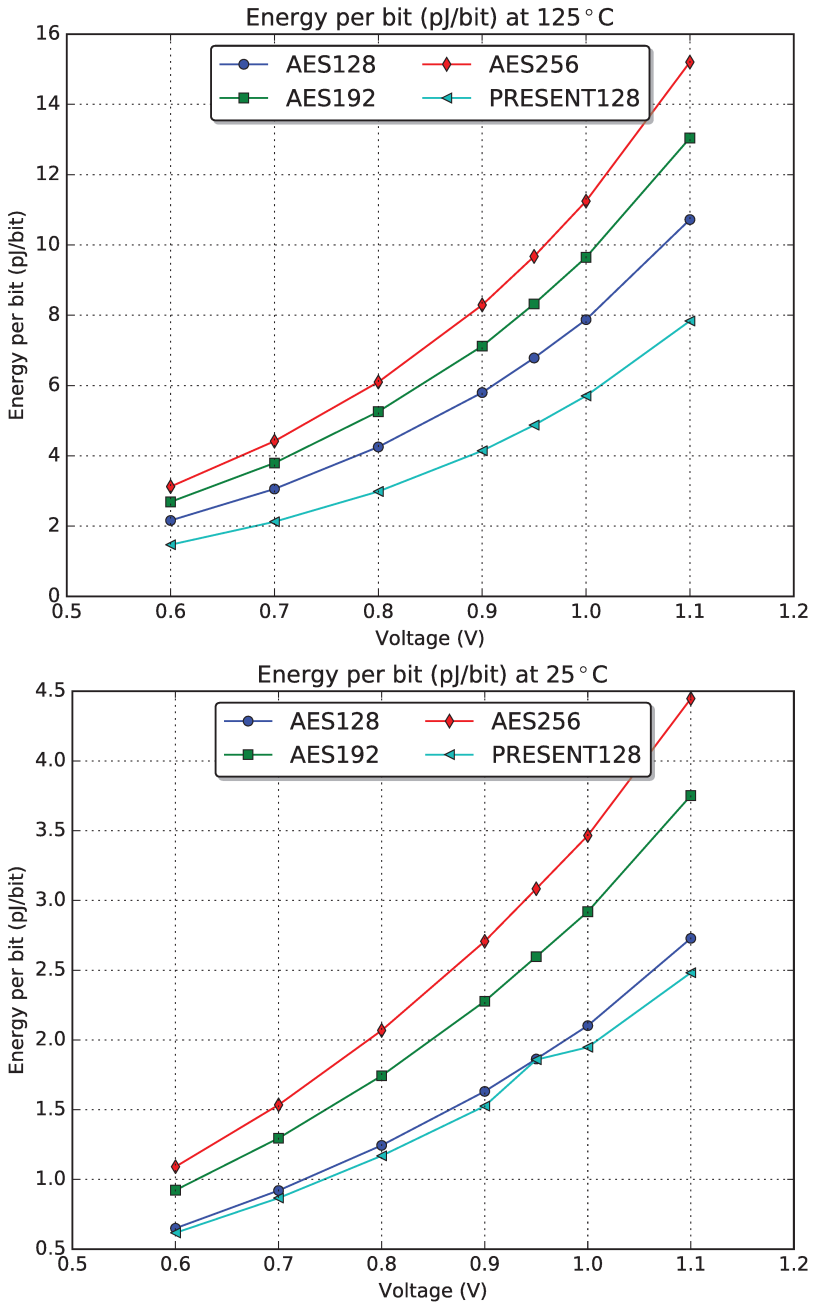


Figure 3.13: Energy per bit of our AES implementation at the typical corner at different working temperatures.

3.4.3 Measured results of Cryptographic Kernel in SNACk testchip

The measurements of the proposed crypto-accelerator have been performed after the testchip was fabricated with the test card. Figure 3.14 shows the test card connected with the FPGA development kit, Spartan-3E. Figure 3.15 shows the SNACk test card and the measurement setup with an oscilloscope. The testchip is controlled through the SPI interface. A UART-to-SPI converter is implemented on FPGA using Spartan-3E kit to allow controlling the testchip using a personal computer. The test programs are written in Python with the communication to the UART-to-SPI converter using PySerial module. The test setups are transmitted from the computer through the FPGA kit to the testchip using UART and SPI interfaces. The power consumption is calculated by measuring the voltage and the current of the cryto-accelerator during its operations.

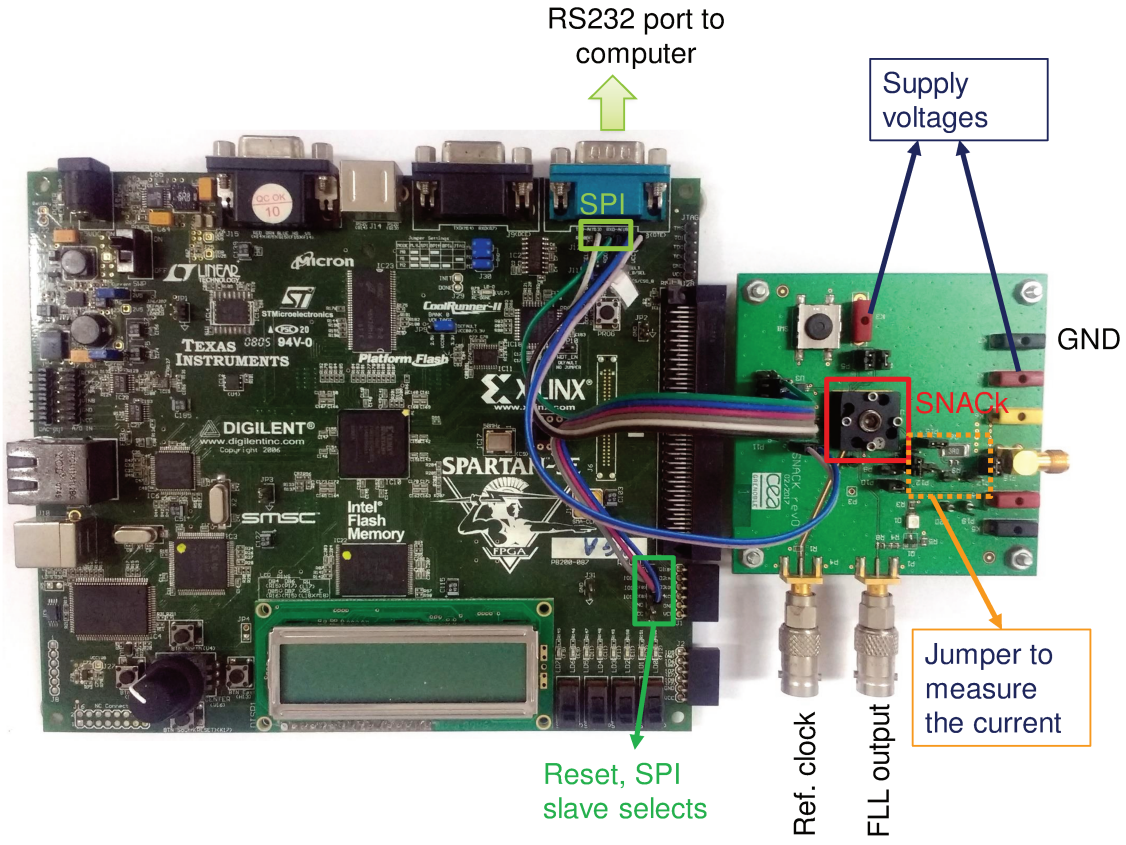


Figure 3.14: SNACk test card and the UART-to-SPI converter implemented in Spartan-3E development kit.

A dedicated power supply pin (*VDD_CRYPTO* in Figure 3.10) is created to measure the power consumption of the block cipher module in SNACk testchip. This pin powers both AES module and PRESENT module. The power consumption is mea-

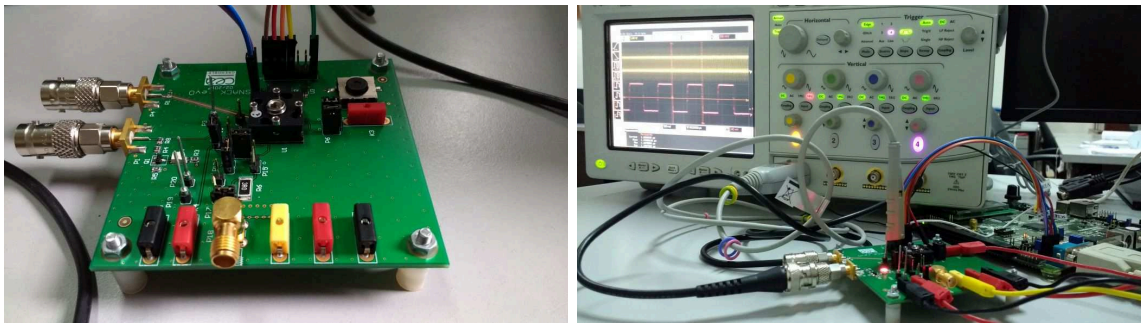


Figure 3.15: SNACk test setup with the oscilloscope.

sured by measuring the supply voltage and the current through this pin. Figure 3.16 shows the power consumption of AES and PRESENT measured at $10MHz$ at the room temperature. The supply voltage varies from $0.4V$ up to $1.3V$ which is a larger range than the one in the power estimation setup. This is because the standard-cell libraries used in the power estimation are only characterized for the working condition down to $0.6V$. At $0.4V$, the testchip can still work at 10MHz and consumes a power of less than $20\mu W$. In the worst case at $1.3V$, AES has a power consumption of $180\mu W$, while PRESENT only spends more than $100\mu W$. There is a small margin in terms of power consumption between AES with 192-bit keys and AES with 256-bit keys. Figure 3.17 shows the leakage power at various supply voltages. This leakage power consumption is measured by turning off the clock generator and the circuit is in the idle state without any activities of the block cipher modules. The two modules are not separated by different power domain during the implementation. Therefore, the result is the sum of the leakage of both AES and PRESENT. It is clear from this figure that the worse case of the static leakage is at $1.3V$ with $40\mu W$. In addition, Figure 3.18 shows that the power consumption increases linearly with the operating frequency.

The measured energy per bit of AES with 128-bit keys is illustrated in Figure 3.19. Obviously, the AES module with 128-bit keys consumes a small energy of $0.4pJ/bit$ at the supply voltage of $0.4V$ and with a throughput of $28Mbps$. With the same condition, PRESENT consumes an energy of 0.3pJ/bit but with a throughput of $17Mbps$. At the highest throughput of $2Gbps$, the proposed AES consume $2.2pJ/bit$. Our proposed crypto-accelerator can achieve very low power consumption with high throughput at the subthreshold supply voltage of $0.4V$.

The proposed architecture shows dominant performance compared to the state of the art which is illustrated in Figure 3.20 and Table 3.1. From the area point of view, the proposed AES architecture with only 128-bit key is 1.5 times bigger than the design in [Zhao2015nsa] in the same technology node, and four times bigger than the design in [Mathew20153m1]. However, our design has four times more throughput than the design in [Zhao2015nsa] and about eight times more throughput than the design in [Mathew20153m1] at the same operating frequency. In compari-

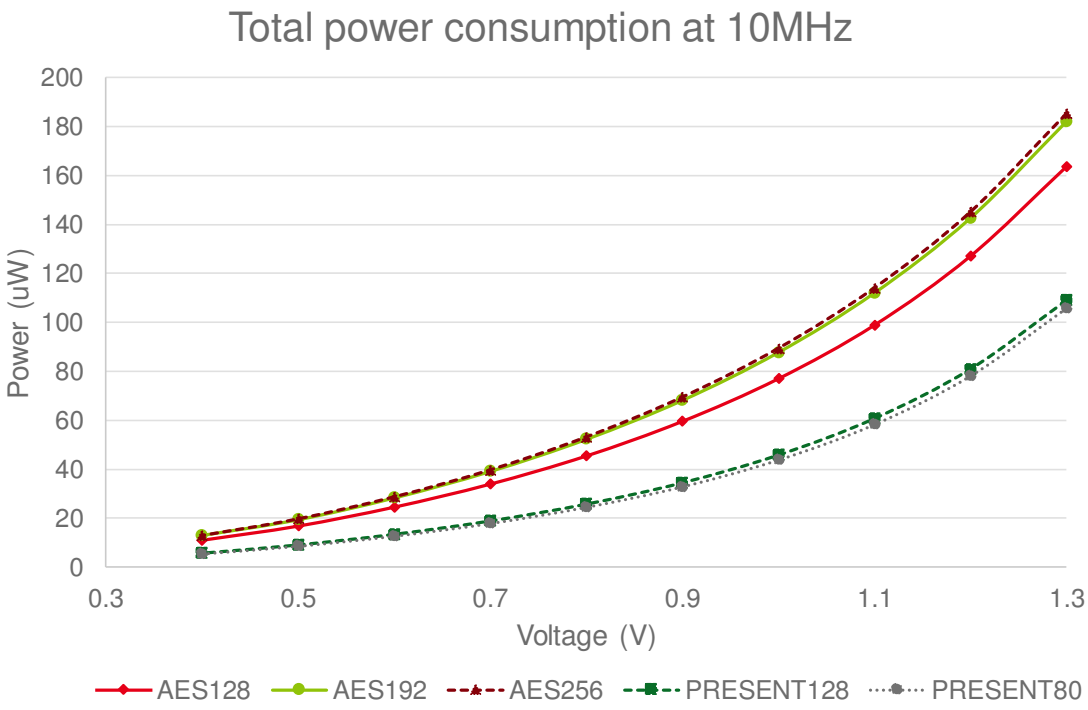


Figure 3.16: Measured power consumption of AES and PRESENT in SNACk with different operating voltages at $10MHz$.

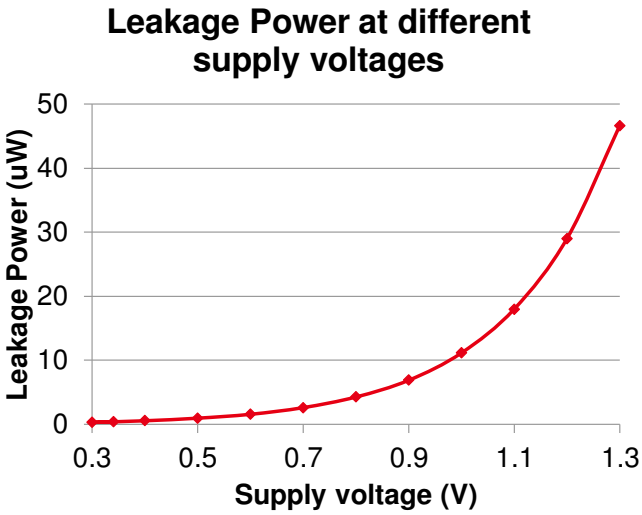


Figure 3.17: Measured leakage power of the blockcipher module in SNACk testchip at different supply voltages at room temperature.

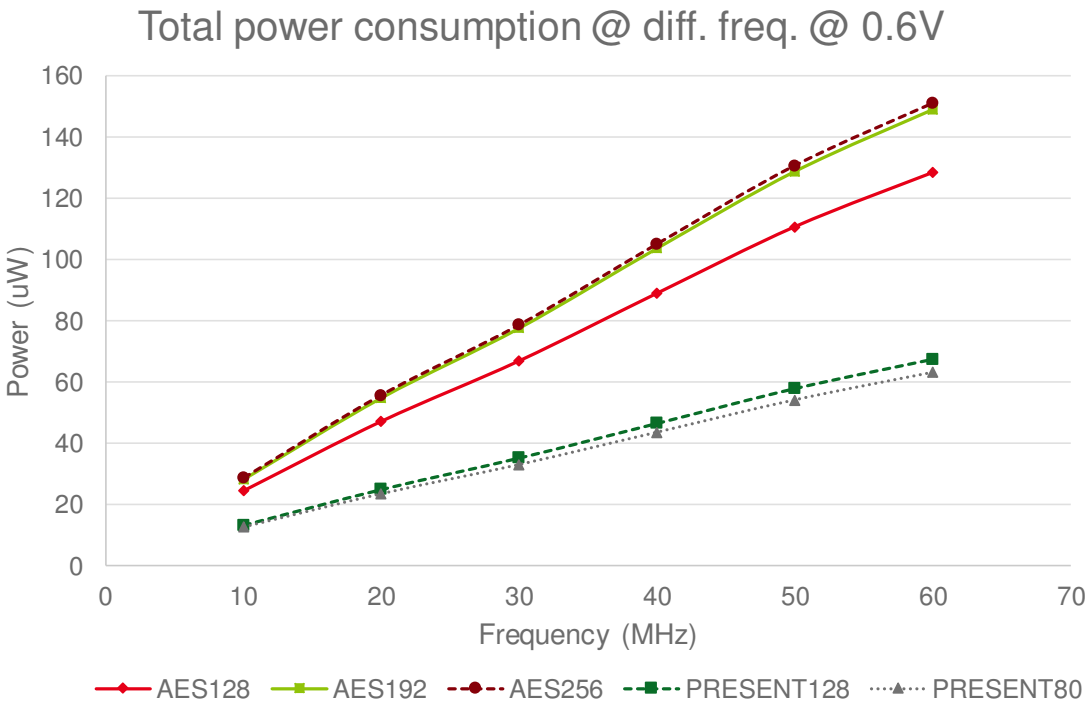


Figure 3.18: Measured total power consumption of SNACk testchip at different operating frequencies.

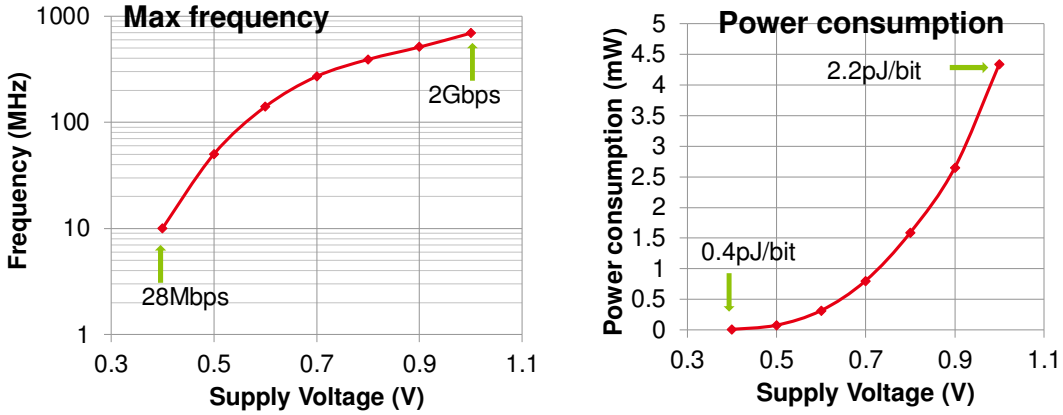


Figure 3.19: Measured energy per bit of AES with 128-bit keys.

Table 3.1: Comparison with other AES implementations

Design	Block size (bit)	Key size (bit)	Arch. (data-path)	Tech. (nm)	#cycles per encryption	Area ($kGEs$)	Working freq. (MHz)	Power (μW)	Throughput ($Mbps$)	Energy/bit (pJ/bit)
The proposed AES	128	128, 192, 256	32-bit	28	44, 52, 60	8.6	10	**24** ($0.6V$), **11** ($0.4V$)	28	**0.65** ($0.6V$), **0.4** ($0.4V$)
Banik, SAC'15 [Banik2015EEE]	128	128	32-bit	90	44	5.5	10	-	28	6.2
Zhang, VLSI-C'16 [Zhang2016AC4]	128	128	8-bit	40	337	2.2	122	100 ($0.45V$)	46.2	2.2 ($0.45V$)
Zhao, TVLSI'15 [Zhao2015nsa]	128	128	8-bit	65	160	4	32	61.7 ($0.6V$)	25.6	2.3 ($0.6V$)
Mathew, JSSC'15 [Mathew20153m1]	128	128	8-bit	22	336	**1.947**	76	170 (0.34V)	29	5.6 ($0.34V$)
Liu, ESSCIRC'11 [Liu2011A2G]	128	128	128-bit	90	10	15	255	5,990	**2.99***Gbps*	2.0
Mathew, JSSC'11 [Mathew20115GN]	128	128, 192, 256	2-stage pipeline	45	5,6,7	100	31	409 ($0.34V$)	**800**	**0.511** ($0.34V$)
Satpathy, VLSI'18 [Satpathy2018grg]	128	128	8 S-boxes (recon-fig.)	14	25	54	620 ($0.75V$), 2.4 ($0.24V$)	26.1mW ($0.75V$), 15.6 ($0.24V$)	3.1*Gbps* ($0.75V$), 2.4($0.24V$)	8 ($0.75V$), 1.3 ($0.24V$)

son with the same 32-bit datapath, according to our optimization, our architecture achieves 20% improvement in power consumption in TSMC 65nm compared with the work in [Banik2015EEE] with a small increase in terms of gate counts. However, the work in [Banik2015EEE] does not provide the information about how they got these results. If their result comes from a post-synthesis estimation, it is less accurate than the proposed results because the power estimation result is based on parasitic values which are sometimes not available or not accurate at the post-synthesis stage. At the same throughput of about 28$Mbps$, our architecture consumes the least power (24μW@0.6V) when compared with the 8-bit datapath designs such as in [Zhao2015nsa, Mathew20153m1, Zhang2016AC4]. At this throughput, our proposed architecture has about three times less power consumption than the best 8-bit datapath design for low-power and low-energy in [Zhao2015nsa]. In terms of energy efficiency, our design consumes the least energy per bit among the low-cost designs [Liu2011A2G, Mathew20153m1, Zhao2015nsa, Zhang2016AC4, Banik2015EEE] with only 0.65pJ/bit (@0.6V@25^oC); and approaches the energy per bit of the high-performance design in [Mathew20115GN] (0.511pJ/bit@409μW@0.34V). Comparing with the work in [Satpathy2018grg] using 14nm technology and a configurable architecture for three symmetric ciphers, the block cipher modules in AES still have advantages in term of power and energy consumption. At the supply voltage of 0.4V, the configurable cryptography kernel in SNACk testchip still consumes a power consumption of 11μW less than the one in [Satpathy2018grg].

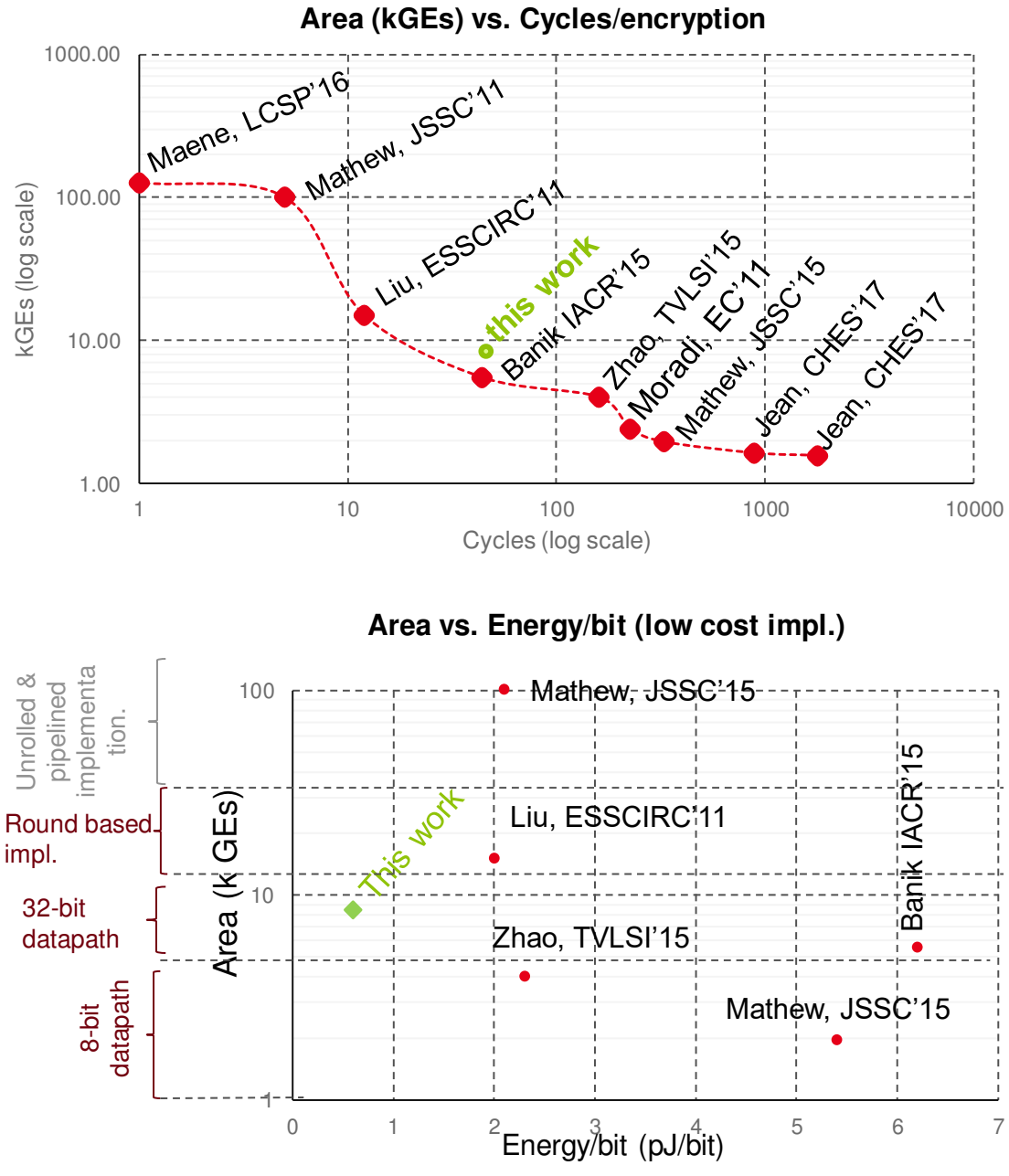


Figure 3.20: Comparison with other low-cost AES implementations.

3.5 Security Evaluation

Hardware security has become a critical issue over the last few years. However, adding countermeasure to the chip not only takes time but also increase power consumption and cost. The main purpose of this work is to investigate the low power features for IoT. For this reason, no countermeasure was integrated into its design. However, the security evaluation was completed by using the current state-of-the-art methods including Correlation Power Analysis (CPA) attacks and Test Vector Leakage Assessment (TVLA) method based on the post-signoff power-trace extraction. Power traces are obtained by performing power estimation using the technology libraries with the netlist after implementation. The simulation waveform of the netlist provides the activities of the design similar to the real condition. This simulates the real situation when the chip is working. The advantage of this method is that it can be performed at different stages of the design such as the post-synthesis netlist or post-place-and-route netlist. The power traces have lower noises when compared with the real power traces. Optimization techniques can be applied to fasten the calculation and reduce the storage size such as the compression of the power traces or selecting only the region of interest. This evaluation can help to identify the flaws early in the design flows. In addition, it also reduces the design cost because the designers do not have to wait until the chip is fabricated. The power-trace generation and processing using power estimation tools will be presented in the next section.

3.5.1 Power trace generation using PrimeTime and Post-signoff netlist

Hardware security is emerging as a major threat to the IoT systems. Therefore, an early method to evaluate the security feature at the design stage is implemented. Accordingly, detailed power traces are extracted from power estimation results using Primetime with the post-signoff netlist and delay information in Standard Delay Format (SDF). Figure 3.21 illustrates the design flows to generate the power traces for evaluation. To begin with, the post-signoff netlist is exported after the placement and route with its SDF. After that, the post-signoff timing simulation is done using QuestaSim. The timing simulation generates the waveform with real timing information. This waveform is then used as an input to power estimation.

PrimeTime is employed to do power estimation with full timing information of each encryption. PrimeTime records the changes at the time that power consumption is changing. Therefore, interpolation is needed to generate the power traces similar to the real cases. The alignment of the power traces is also performed so that each power trace has the same reference point in time. This can be done using the start of each encryption. Figure 3.22 shows the trace processing steps. Because PrimeTime only records the power when there is a change in the event, therefore, a zero-order interpolation filter is applied to get the power traces at each time step. After that,

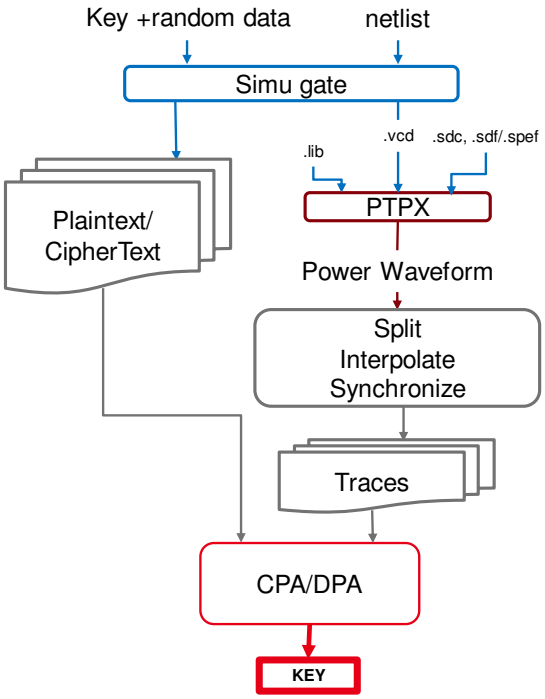


Figure 3.21: Design flow to generate the post-signoff power traces for evaluation.

the power trace for each encryption is extracted based on the encryption time. Furthermore, in order to reduce the processing time and reduces the size of the traces to simulate different sampling speed of the trace measurement device, adjacent points can be added and reduced into only one point in the final power traces. This compression technique is used in our trace processing techniques to reduce the size of the power traces and fasten the computation. For instance, the power traces of 20,000 encryptions exported by PrimeTime have the size of $22GB$ in text format, while its compression using Gzip only occupies $4.7GB$. After the Gziped file processed by this trace processing framework using the interpolation, alignment and trace compression technique, the trace file, which contains the region of interest, takes $4.6GB$ of disk space. The file size can be made smaller by reducing the number of points in the trace file which is similar to reducing the number of sampling file in a real measurement. Figure 3.23 illustrates the whole trace processing framework including extracting and splitting the power traces, interpolating and compressing them and mounting CPA attacks. This trace processing framework has been validated using the reference design [Aes128Opencore] which was successfully attacked by this framework to recover the secret key.

In this work, the physical side-channel attacks are evaluated based on the simulated power traces from a post-signoff design flows. PrimeTime is used to generate the power traces from the timing simulation and the physical parasitic of the designs. After that, the power traces are interpolated and compressed to simulate

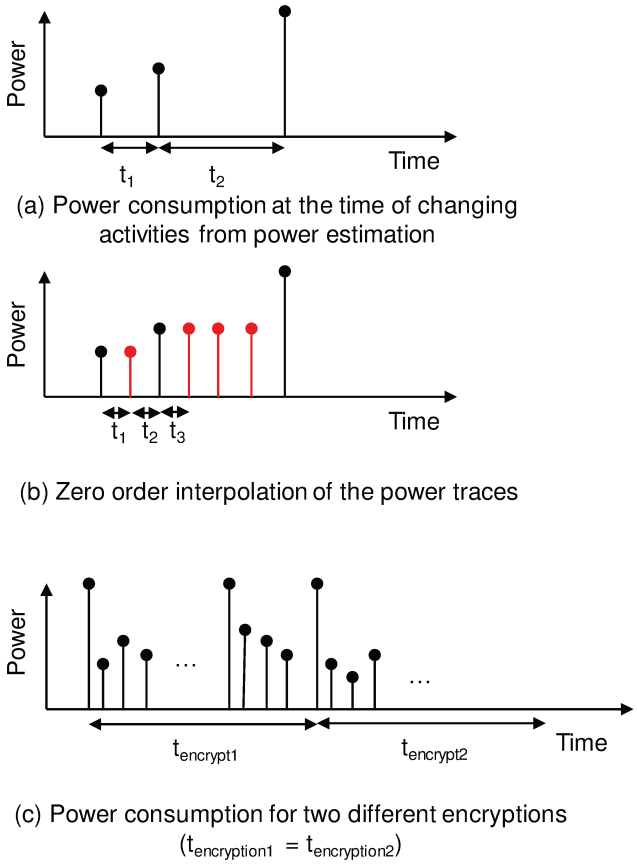


Figure 3.22: Trace processing of the power curve from PrimeTime.

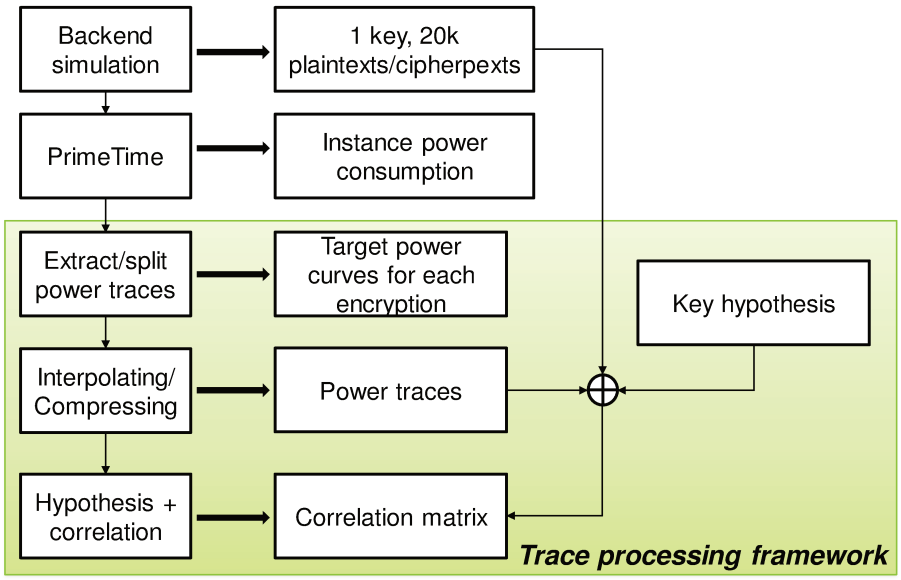


Figure 3.23: Trace processing framework.

the real trace acquisition. This framework has been used in successful attacks on the implemented designs, therefore, it will be used to generate the power-traces to evaluate the security features in this work. The security evaluation based on power analysis using Correlation Power Analysis and Test Vector Leakage Assessment using this trace-processing framework will be presented in Section 3.5.2 and Section 3.5.3, respectively.

3.5.2 Test Vector Leakage Assessment evaluation

This work firstly uses Test Vector Leakage Assessment (TVLA) [Goodwill2011atm] method to identify the possible leakages of the proposed cryptographic designs. TVLA includes two types of tests to evaluate the information leakage using Welch's T-Test function. The first one is the non-specific test – a fixed-versus-random test which compares the T-test score between the power traces of a fixed key and fixed data encryption and the one of a fixed key and random data encryption. The other is the specific test which can be used to identify the information leakage at different steps in a round such as the round output, S-Box output, round input versus round output, and the leakage at a specific byte. The total number of specific tests is 896 tests per round.

The proposed designs and the reference design on Opencores [Aes128Opencore] are implemented in the same technology up to signoff. The signoff netlist and delay information are used to generate 20,000 encryption traces. These power traces are compressed to extract only the interesting part to do the TVLA evaluation. The TVLA evaluation is performed on all ten rounds of AES 128-bit key case. This leads to a total of 8.960 tests of specific tests. Figure 3.24 presents the number of test failures for each category of the proposed design and the design on OpenCores [Aes128Opencore]. Obviously, the two designs have different leakages. The proposed design has the weakness in ROUT and SOUT test while the design on OpenCores shows strong leakages in RIRO, ROUT and SOUT. In addition, The proposed design does not have any test fails in ROUT_BYTE0 and ROUTE_BYTE1 tests. However, the reference design on OpenCores shows a few test failures. In general, the proposed design has 21 test failures fewer than the reference design.

In addition, the non-specific tests are also performed on the two designs. 10.000 power traces of fixed data and fixed key encryption are generated for the proposed design and the reference design. After that, they are divided into two groups to calculate the T-test scores. Figure 3.25 and Figure 3.26 shows the absolute T-test scores of this test for the proposed designs and the reference design, respectively. It is clear from these figures that the two designs do not pass the non-specific test because there is no countermeasure implemented in the two designs. In terms of T-test scores, the reference design has five times bigger in T-test scores than the proposed design. The power traces used in this test is generated using the post-signoff power estimation. Hence, some real conditions are not considered. For example, there is no

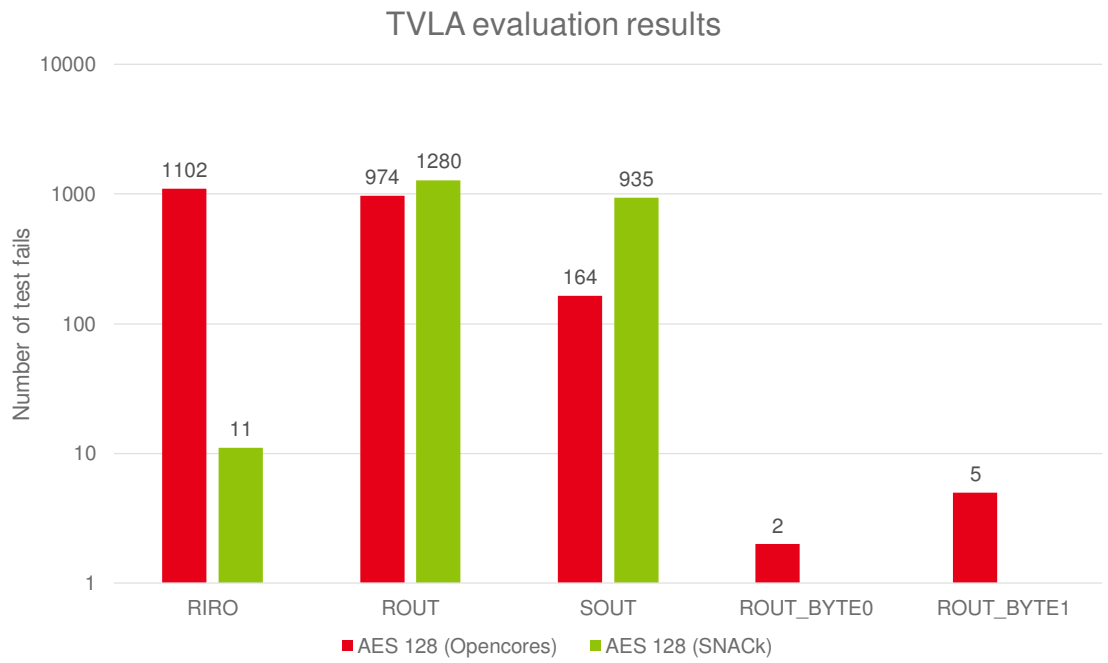


Figure 3.24: TVLA evaluation results of the specific test of the proposed design versus the design on Opencores [Aes128Opencore].

clock jitter in this case, so, the power traces are perfectly aligned. In a real system, a noise created by the running components or from the clock source, the noise from the power supply and so forth might make the T-Test score much lower than this theoretical case.

All things considered, the security evaluation using TVLA method was performed on the proposed design and a reference design from OpenCores. The evaluation results show that our design was exposed to some of the attacks especially in the Round-Out (ROUT) tests and the S-box-Out (SOUT) tests. While the reference design has the weakness in Round-In Round-Out (RIRO) tests, SOUT tests, and ROUT tests. For non-specific test, our design shows five times lower than the reference design in terms of the T-test scores. Regarding TVLA results, the security levels of the optimized design are equivalent to the reference design. It means that the optimizations do not introduce new leakages.

3.5.3 Correlation Power Analysis attacks on estimated traces

TVLA results provide a level of confidence to conclude that the design has exploitable leakages. However, it might not lead to a successful key-recovery attack [Schneider2015lam]. Therefore, this work also performs Correlation Power Analysis (CPA) attack, a key-recovery attack, to verify that the proposed optimizations do not introduce any new leakage. The attack is based on the power traces extracted

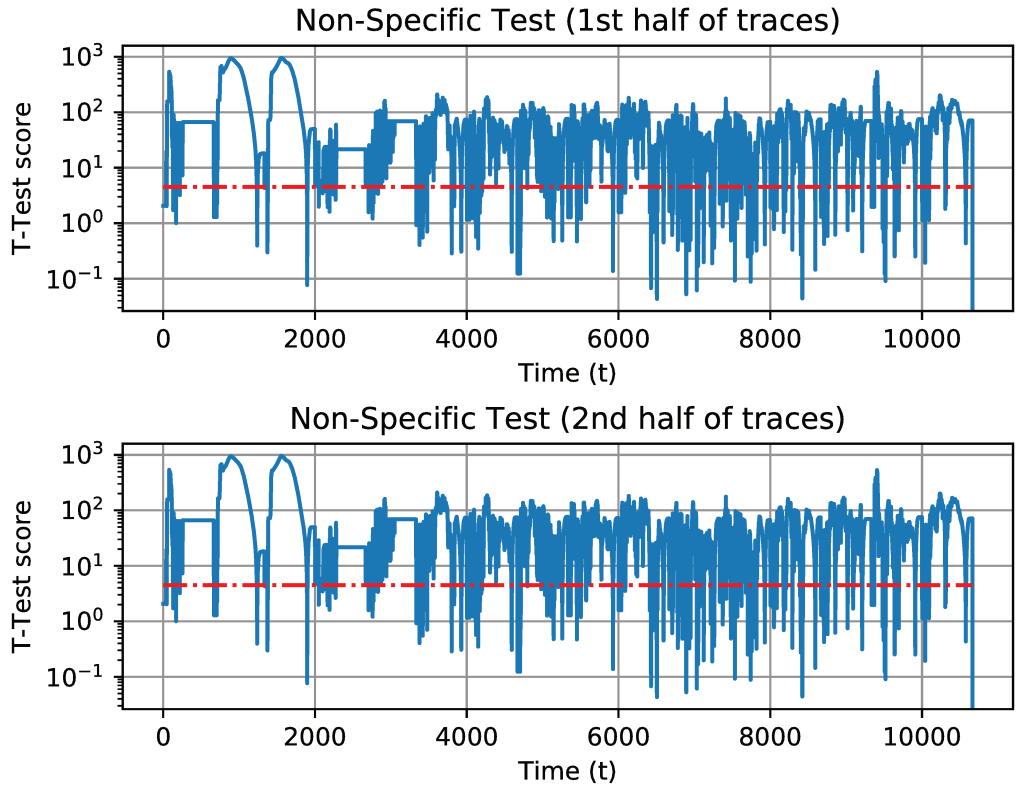


Figure 3.25: TVLA evaluation results of the non-specific test of the design on Opencores.

through the post-signoff power estimation. A simulation of 20 thousand encryptions of our design in 128-bit key encryption mode is executed to capture the ciphertext and the power traces. For comparison, we execute the same hardware implementation process with a full parallel design from OpenCores [Aes128Opencore]. In general, the more parallel level of the datapath, the harder it is to attack the design since parallelism is one way of hiding countermeasures. 8-bit datapath without protection is more exposed to this type of attacks because the number of traces required to perform the attack is very small. According to DPA contest [dpacontestV3], even a round-based datapath with full 128-bit parallel computation on FPGA, with good measurement equipment, only 800 traces are required to reveal the key of the cryptographic devices. Figure 3.27 presents the results of our experiment on post-signoff power traces. The AES 128-bit datapath requires about 4,000 traces to reveal 16 bytes of the secret key while with our architecture, even with 20 thousand traces, only 12 bytes are revealed. These bytes are hidden because at the end of each round, the data registers are overridden with new data. This hides the correlation of the activity of the last 4 bytes of the key which increases the resistance of our design to the last round CPA.

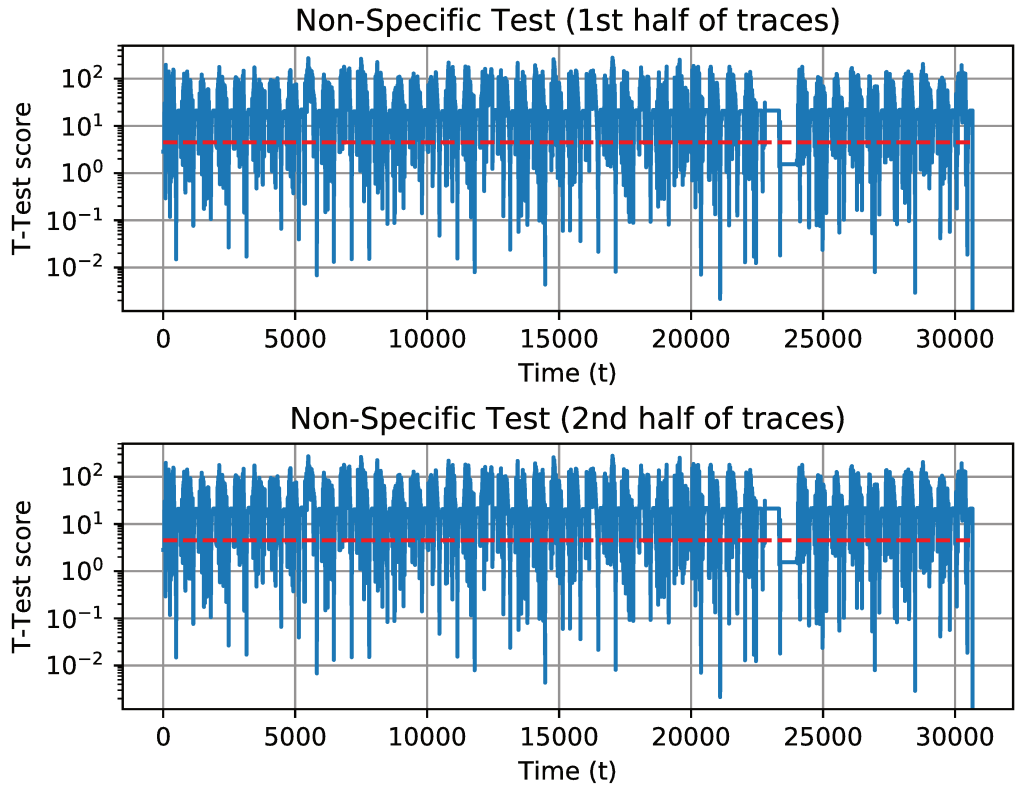


Figure 3.26: TVLA evaluation results of the non-specific test of the proposed design

.

Nevertheless, in terms of security, based on the proposed architecture, this design might be attacked using Differential Power Analysis attack, Correlation Power Analysis attack, fault attacks if the attackers have the full control of the chip. For timing fault attacks, it is possible to use an internal clock source so that the attacker cannot control its operating frequency. The design cannot resist against fault attacks which use the external sources such as laser or EM.

For power analysis attack, at the first round, the key is intermediately XOR with the plaintext, therefore, if the plaintext is predictable, the proposed architecture will be exposed to first-round DPA and CPA attacks. For the last round attack, the proposed design has certain bytes which are not exposed to CPA. The experiments presented in Section 3.5.3 shows that only 12 bytes are exposed in the last round attack. Regarding system integration, power analysis attacks can be avoided by using external protection methods such as using a voltage regulator to balance the power consumption of the chip. However, this will increase the power consumption.

Additionally, the proposed architecture is an IP core in a system-on-chip. Therefore, it will have to read the data from memory through the system bus to do the encryption. This can add another security concern. Other IP cores probably come

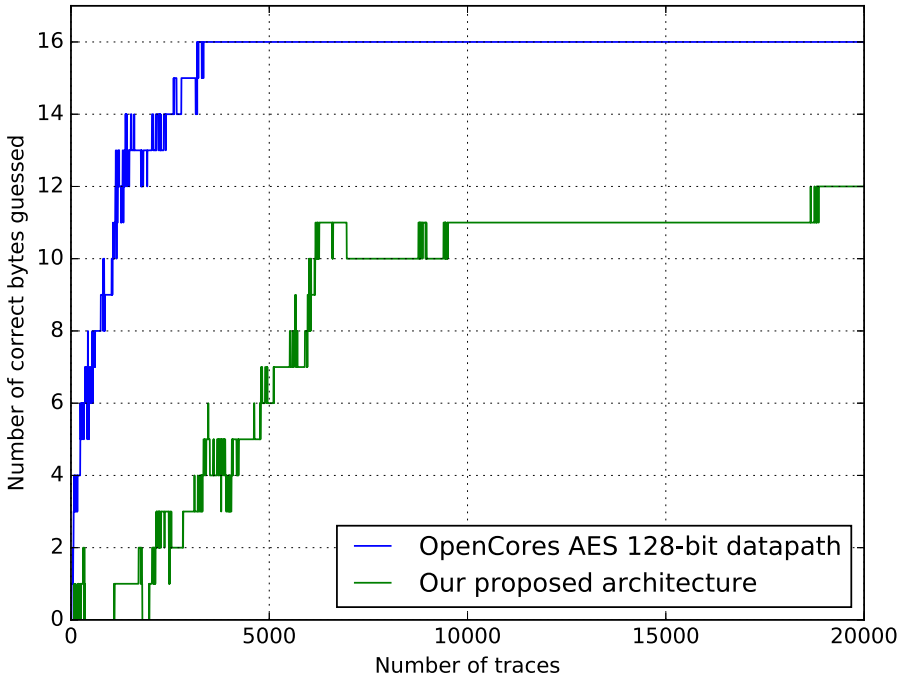


Figure 3.27: Number of correct guessed key bytes (in 128-bit key mode) by last-round CPA attack.

from different vendors, therefore, they may monitor the bus, and expose the data to accelerate the attacks. IP cores containing hardware trojans are becoming new threats in IoT system because IoT might contain many IP cores from different vendors. Moreover, hardware trojans are hard to be detected.

In summary, the proposed design firstly focuses on optimizing area, throughput and power/energy consumption of the data encryption module, i.e., the block cipher module in SNACk testchip. As the design of countermeasures is a different area of research, this work does not consider the countermeasures for different attacks such as fault attacks or side-channel attacks. Regarding the TVLA and CPA results, the proposed architecture has certain resistance to correlation power analysis attacks. Our experiment shows that only 12 bytes are revealed when mounting correlation power analysis attacks on the designs. At the system points of view, there exists a threat of data movements along the system bus, which can be monitored by hardware trojans from third-party IP cores. These problems can be solved by using a new mechanism to implement cryptography. In-Memory Computing is a new promising solution in this case. Countermeasures such as masking and hiding techniques can be easily implemented using In-Memory Computing by executing multiple operations at the same time. Because of the regularity of the memory, these countermeasures might increase the security features of the IoT system at no extra cost.

3.6 Conclusion

Current ultra-low-power and ultra-low-cost security solutions for IoT security usually come along with using block ciphers as the main security primitives. Block cipher is one of the deterministic components in a secure system and can be configured to perform different modes of operations. However, block cipher consumes a large amount of power. To minimize the power consumption as well as the hardware cost, lightweight cryptography has been chosen with a reduction in the security level by using small block size, small key size and a large number of rounds. This leads to a reduction in the throughput of the system.

On the other hand, recent IoT proposals have been continuing using AES as the main security primitives because of its well-studied security features and performance in software. Lightweight cryptography such as PRESENT with lower security level but providing low cost and ultra-low power consumption has not been adopted yet. Furthermore, depending on the applications' needs, lightweight security functions can be selected when ultra-low-power mode is needed to extend their operations. Strong security can be provided by conventional security functions, while configurable security solutions can provide flexibility in terms of applications' points of view.

Various previous works in AES focused on the serial implementation by using 8-bit datapath and one-S-box architecture. 8-bit datapath architectures can reduce the hardware cost and power consumption, but it also reduces the system throughput and increases the energy consumption. 8-bit datapath architecture also requires additional hardware registers to store the intermediate values. Trade-offs among security levels, hardware cost, throughput and power/energy consumption must be considered carefully.

This work proposes a configurable block cipher module with a traditional algorithm AES and a lightweight cryptography algorithm – PRESENT. IoT applications can select among various options such as key sizes, block sizes and the number of rounds which corresponds to different security levels and different power/energy consumption profiles. Accordingly, to reduce the hardware area, AES was implemented with 32-bit datapath architecture and supported with power reduction techniques in the datapath. Also, a round base architecture is used to maximize the throughput of PRESENT. The proposed PRESENT architecture has about half of AES dynamic power consumption.

Furthermore, multiple optimization strategies for AES 32-bit datapath to achieve a low-cost, high-throughput, ultra-low-power, ultra-low-energy design with multiple levels of security have been proposed. The area of the proposed AES architecture is saved by a reorganization of both datapath and key expansion to minimize the number of registers and control logics. The power consumption is reduced by choosing the S-boxes for low-power, by minimizing the activity in the key expansion and the datapath, and by applying a clock gating strategy to data storage registers. The throughput is maximized by using 8 S-boxes and doing key expansion in par-

allel with the encryption path. Multiple key sizes of the encryption module provide different security levels which help IoT applications to adapt to a wider range of security protocols and mechanisms. In terms of power and energy consumption, at $0.6V@25°C$, the proposed AES design can achieve power consumption of less than $20\mu W$ for all key configurations with the energy consumption of less than $1pJ/bit$ with the throughput of $28Mbps$ at $10MHz$. In this condition, the proposed AES implementation has achieved nearly the same energy consumption in comparison with the lightweight cryptography algorithm PRESENT on the same technology node – ST FD-SOI $28nm$ technology. Nevertheless, this work does not look into the countermeasure for power analysis attack, but the security evaluation is performed on the AES design to identify the possible weakness in terms of hardware security. The evaluation results using Correlation Power Analysis (CPA) and Test Vector Leakage Assessment (TVLA) show that the optimizations are somewhat beneficial to security features. In the key recovery attack using CPA methods, only 12 bytes out of 16 bytes of the key are revealed using the last-round key hypothesis. TVLA test results show that the proposed design has equivalent information leakage when being compared with the reference design on Opencores. The proposed block cipher module obviously can be used for different applications with different security requirements for future IoT systems.

On the other hand, the crypto-accelerator approach to design the encryption modules still embrace a number of weaknesses. For example, countermeasures must be integrated separately, while countermeasures for one attacks might not be applied to the others. In addition, the hardware accelerator cannot be changed after production, therefore, in case of any flaws in the design of the hardware module, the only solution is to make a replacement with a more secure one. This will increase the cost of maintaining security and bring about the interruption of the service. This urges the demand for investigating a new method for an efficient implementation of the cryptography algorithms. In the next chapter, a mapping of the same algorithms will be explored using a new concept of computing so-called In-Memory Computing. As earlier said, In-Memory Computing can help improve the flexibility and configurability of the design.

Chapter 4

Using memory as acceleration for data encryption

The proposed crypto-accelerator can provide ultra-low-power consumption with configurable capability. With different optimizations as presented in Chapter 3, the block cipher modules in SNACk testchip have small hardware footprint with ultra-low-energy consumption. However, these advantages are from the fixed hardware structure with specific optimizations which cannot be altered after fabrication. Therefore, specific solutions also reveal certain drawbacks.

To begin with, the crypto-accelerator often has a fixed structure with fixed algorithms to optimize hardware cost and power consumption. However, it is crucial for IoT standards to evolve to adapt to new security threats and to mitigate new attacks. New cryptanalysis techniques along with newly discovered weaknesses in cryptography algorithms might lead to the exclusion of algorithms out of the standards. Therefore, flexibility and configurability which are adaptable to new IoT standards and capable of mitigating new threats are required for future IoT applications.

In addition, systems on chip are threatened by hardware trojans. IoT devices use highly integrated systems on chip with various IP cores procured from different vendors. It is possible for a third party IP vendor to insert a hardware trojan into their IP cores to monitor the system bus for secret information. Moreover, crypto-accelerators typically read out the data from memory through the system bus before performing their tasks and writing the data back into the memory. Consequently, this not only raises a security threat but also creates communication overheads.

Possible solutions to overcome this overhead could be Processing In Memory (PIM), In-Memory Computing and Near-Memory Computing which can be employed to implement the security primitives directly in the memory. Processing In Memory and Near-Memory Computing use the accelerator embedded in the memory controller, hence the proposed crypto-accelerator can also be applicable to them. Meanwhile, In-Memory Computing enables the execution of logical or arithmetic operations

using the memory array itself. Thus, In-Memory Computing shows big potentials to implement flexible and configurable security solutions which can map various security primitives using the memory itself. However, because of the serial operations of the memory, flexibility and configurability have to be traded off with the system throughput and power consumption. Further study is required to prove its effectiveness.

This chapter will look into techniques to perform data encryption using In-Memory Computing to overcome the drawbacks of hardware crypto-accelerators. Two case studies using the same algorithms as in Chapter 3 are selected to evaluate the flexibility, configurability and efficiency of the proposal. All operations of AES and PRESENT can be accomplished in a new type of memory called SmartMem. SmartMem supports not only the normal memory operations such as reading and writing but also the logic operations such as AND, OR, NOT, XOR, XNOR and SHIFT. Using this new kind of memory, encryption can be conducted directly at the place where the data are stored. Therefore, the data transfer is minimized while the throughput of the whole system can be improved and the power consumption can also be reduced.

This chapter is organized as follows. Section 4.1 gives an introduction to various benefits of data encryption using memory elements. After that, Section 4.2 describes the mechanism to do In-Memory Computing which is used to create SmartMem array. The implementations of AES – a conventional cryptography algorithm, and PRESENT – a lightweight one are presented in 4.3. Finally, Section 4.4 concludes this chapter along with its perspectives.

4.1 Introduction

In a traditional computer-based system, most of the area of the chip is actually occupied by cache memories or SRAMs which are used to accelerate the computation by reducing the data access time and power consumption. However, the processor has to read the data stored in these memories, then do the calculation and write back the results into them. Consequently, it creates a huge communication overhead and wastes power/energy consumption. Recently, with the advances in the design of memories, SRAM and DRAM memories can be designed so as to execute simple logical and arithmetic operations on the data directly in the memories or through the memory controller. This results in a shift in system design mindset as well as brings benefits to the design of cryptographic modules.

Figure 4.1 summaries different computation architectures for cryptographic modules. Cryptography in the first box is conducted by software running by a CPU or a microcontroller. In this scheme, the processing unit needs to read the data from the memory, do the cryptographic algorithms, then write the data back into the memory. This simultaneously reduces the system throughput and increases power consumption to the extents that the processing unit has to wait for the data trans-

ferred into the memory, and this data transfer occupies the system bus. The next improvement is to design the cryptographic algorithm as an IP core and integrate it into the processing system as a co-processor in the trend of System-on-Chip (SoC) design paradigms. This releases the processing unit to do the other tasks but still occupies the system bus. For the next advance, In-Memory encryption can be done by using in-memory computation. Logical and arithmetic operations can be designed as a part of the memory (in the cache for example). Instead of reading the data and doing the calculations on them, the CPU now only sends controlling information for doing in-memory encryption. This reduces the communication overhead, frees the CPU for the other tasks and frees the system bus for other IPs.

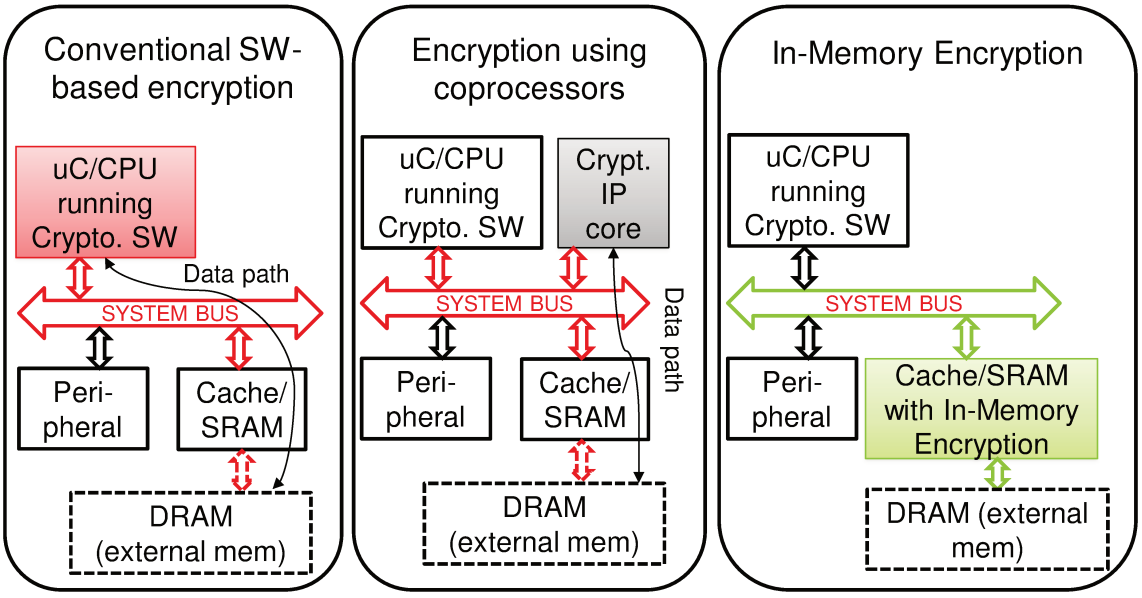


Figure 4.1: Comparison of traditional software-based encryption, cryptography co-processor, and in-memory encryption.

In terms of security, in-memory computation in general and in-memory encryption in particular comes with many advantages. Firstly, it reduces the risk of exposing secret data when the data is transmitted from the memory to the processing unit. In addition, in-memory computing can perform operations in parallel using multiple memory banks. Each memory bank can execute its own operations in parallel with each other. This parallel execution could be a method to protect the security algorithm from Power Analysis Attack. In the current state of the art, this technique is performed by generating some random values and doing the same operation in parallel with the random values. If the in-memory computing is used, each memory bank or multiple data lines will be executed. If the attacker does not know all the information (with the same assumption that the attacker does not know the random values) the same effect as a natural masking can be achieved. However, in this case, useful calculations are used instead of running some random operations. Other coun-

termeasures can also be applied when using in-memory computing. For example, the independent operations can be randomized so that it will be harder to guess at which time a certain operation is executed. Even fault tolerant mechanisms can also be applied. The memory can be designed with error correcting code to detect errors and replace it by a redundancy cell. This all benefits the in-memory computing especially in-memory encryption.

Processing in Memory (PIM) has been developed before by embedding the ALU into the DRAM controller. In this mechanism, the dedicated processing element is included in the memory controller for calculation. The calculation is completed by adding dedicated hardware closer to the DRAM where the main data is stored. The DRAM controller will read the data, do the calculation on them, and write the results back into DRAM. SmartMem is differentiated from PIM by using In-Memory Computation which uses SRAM memory as an accelerator for calculation. With a special design of the memory cell and the IO circuits, the area overhead is minimized. SmartMem structures can also be used as acceleration for PIM. In the next section, mechanisms for in-memory computation will be addressed. After that, these mechanisms will be used to create the Encryption-In-Memory modules.

4.2 Computation In-Memory mechanism and SmartMem

SRAM memory is one of the most common components in a computing system, however, adding computation into memory is a challenge. The main purpose of SRAM is to write data into the memory bit cells and read data from them. Memory is often a full custom design, therefore, adding computation to them might increase the complexity of the design. However, adding computation to memory has many advantages. Firstly, it reduces the data bandwidth for the memory IO. The data can be processed directly in the memory without reading it out of the memory. This will minimize the power consumption of the system. In addition, it also minimizes the risks of exposing the secret data when the data are read through the system bus. In the security view, SRAM memory is more regular than ASIC design. Therefore, the delays and glitches are less abnormal when compared to ASIC implementation. This makes it more secure when using the memory elements as the acceleration for IoT. In-memory computing has more resistance to DPA attacks as claimed by Zhang et al. in [Zhang2018rar]. A software implementation might need only 20 power traces to reveal the first byte of the key while the near-memory architecture of Zhang et al. needs 300 traces to obtain the same key. Nevertheless, memory design can also be hardened to reduce information leakage, for example, the work presented in [Rozic2012dsf].

Recently, new ideas have been raised on adding some logical and arithmetic operations to the memory without changing the memory structure very much. For

example, Jeloka *et al.* in [Jeloka2016anc] create logic computation using TCAM cells by modifying the sense amplifiers of the original TCAM memory. In the logic-in-memory mode, this type of memory is able to do simple logic operations such as AND or NOR. However, for implementing block cipher in memory we might need other operations such as XOR and SHIFT. Akyel *et al.* in [Akyel2016ddr] show a memory structure which supports multiple operations on each row. In this work, we use this memory structure as the accelerator for in-memory encryption. This memory structure is called SmartMem.

A SmartMem bit cell is constructed using traditional 6T SRAM bit cells with two additional read ports which lead to a 10T SRAM cells. The detailed structure of the SmartMem bit cells is presented in Figure 4.2. It contains one Write Word-Line (WWL) and two read word lines including Read Word-line Left (RWL) and Read Word-line Right (RWR). Reading and writing data are separated using two different pairs of bit lines. Write Bitline Left (WBL) and Write Bitline Right (WBR) are used for writing operation of bit cells with the same mechanisms as 6T SRAM; while Read Bitline Left (RBL) and Read Bitline Right (RBR) are used to read the value stored in the bit cells. For read operations, RBL and RBR must be precharged first. After that, depending on the value stored in the 6T cell, RBL and RBR will be pulled down to '0' or kept at '1'. Because of this configuration, RBR reads the value of the bit cell while RBL reads its inverse.

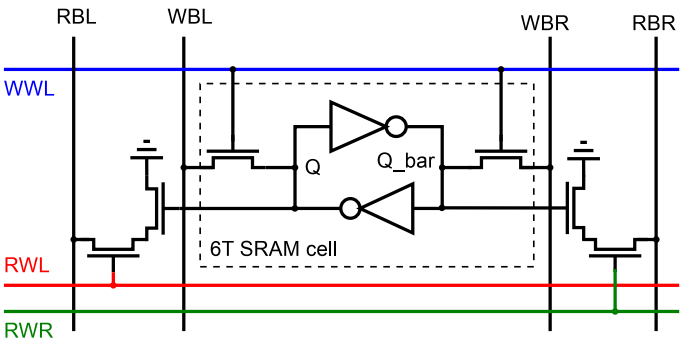


Figure 4.2: 10T SRAM cell for In-Memory Operation proposed by Akayel *et al.* in [Akyel2016ddr].

Writing operation is accomplished by precharging WBL and WBR with the correct voltage value. After that, the word line is activated. This is the same as writing operation of 6T SRAM cell. Similarly, reading operation is carried out by precharging either RBL or RBR then activating the corresponding read wordline RWL or RWR. However, RBL will read the inverse of the value stored in the bit cells. For in-memory calculation, we will need to control both read ports.

Figure 4.3 describes the in-memory operation using this type of bit cells. Logic operations can be performed on the data in the same column of memory. For example, by performing read operation on both bit cell A and bit cell B using two read ports

of each cell, different logic results could be obtained. In this example, RBL has the logic function A NOR B because either A or B stores the logic value '1' will result in the logic value '0' at the output. In contrast, RBR has the logic function A AND B because both A and B must store a logical value of '1' to make the RBR stay in '1'. Further logic operations can be constructed based on this basic operation. For example, XOR can be performed by adding a NOR gate between two read bitline in the peripheral circuits. In [Akyel2016ddr], Akyel *et al.* use this memory cells and some additional logics in the peripheral of the memory to perform simple logic operations and arithmetic operations such as AND, OR, NAND, NOR, XOR, XNOR, SHIFT, addition, subtraction and so on. The logic functions can also be performed on multiple rows. However, in this work, we only focus on mapping of cryptographic function into the memory, therefore, we only focus on XOR and shift operations. The XOR operation will be performed on two selected rows of the memory.

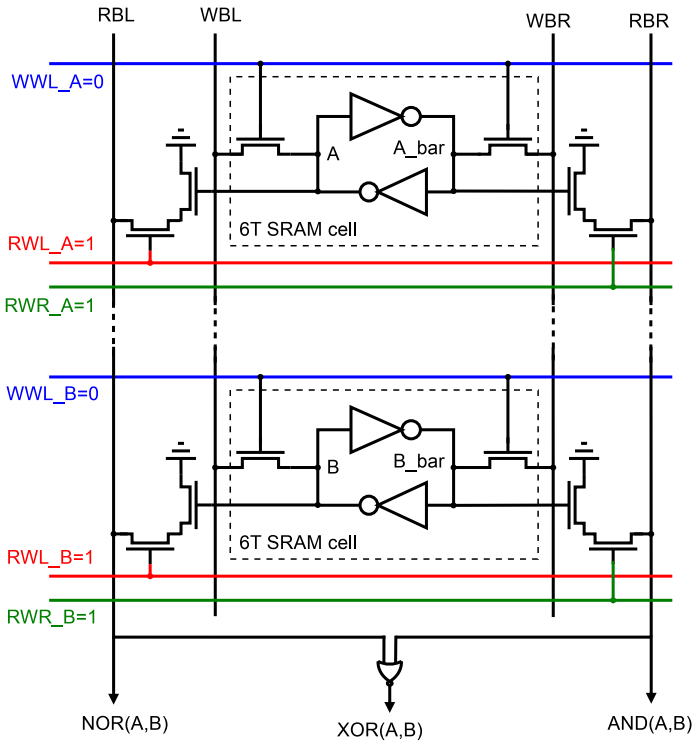


Figure 4.3: In-Memory logical computation using 10T SRAM cells.

To use the additional operations of SmartMem, the interface of the traditional SRAM must be modified to add more control information. Figure 4.4 illustrates the differences between the traditional SRAM and the SmartMem. The signals in red are the additional signals to support the smart operations. As can be seen from the figure, Write Enable (WE) signal is extended into two bits to support four modes: normal-read, normal-write, smart-read and smart-write. In the smart-read operation, the contents of multiple memory rows will be read, and the results of the smart operations on these rows will be obtained at the peripheral circuits of the memory.

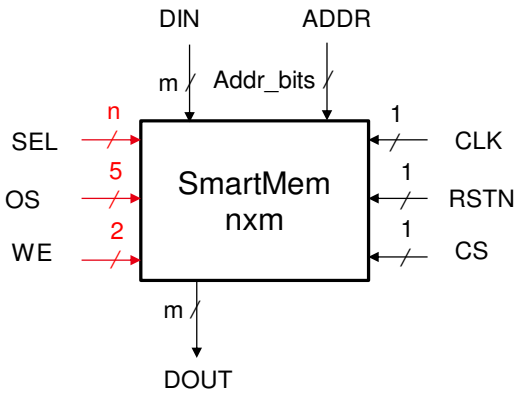


Figure 4.4: SmartMem's block diagram with its inputs and outputs.

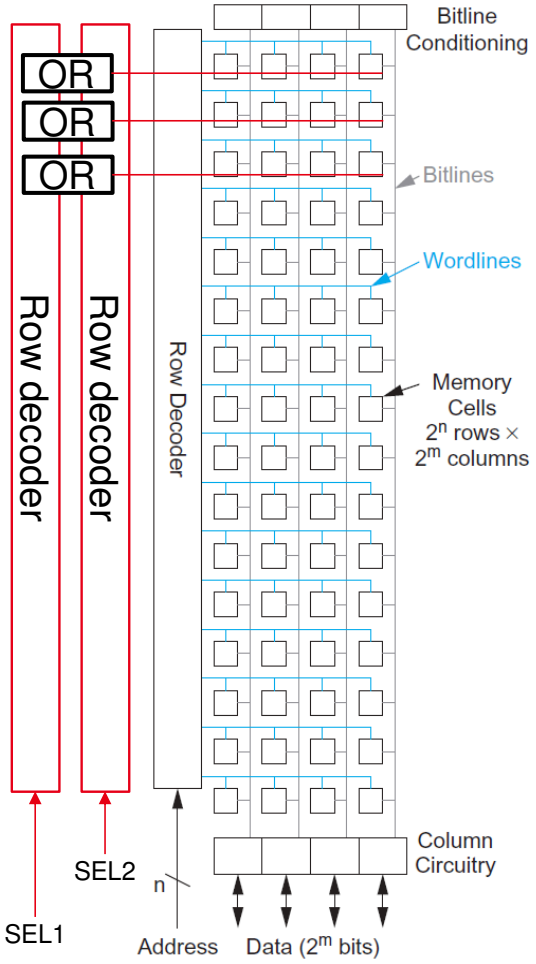


Figure 4.5: SmartMem structure with two selection lines.

Additionally, in the smart-write operation, the contents of multiple memory rows will be read at the same time to perform the smart-read operation. After that, the result of the smart operation will be obtained in the peripheral circuits, and then will be written back into the memory. The smart operation is specified using the operation-selection signal (OS). The selection signal (SEL) is used to select different rows of the memory for the smart operations. The bit width of SEL signal is equal to the number of rows in the memory. A large number of rows makes the interface of SEL signal impractical. Therefore, to reduce the number of bits of the interface, the SEL signal is generated by using the addresses of rows, which will be accessed, and onehot decoders. The results from the onehot decoders will be ORed together to generate the SEL signal. Consequently, the inputs to the memory will be the addresses of the accessed rows passed through onehot decoders and OR together as in Figure 4.5. The more rows are selected, the more decoders must be added to decode the row address. This view of the memory is similar to multi-read-port SRAM.

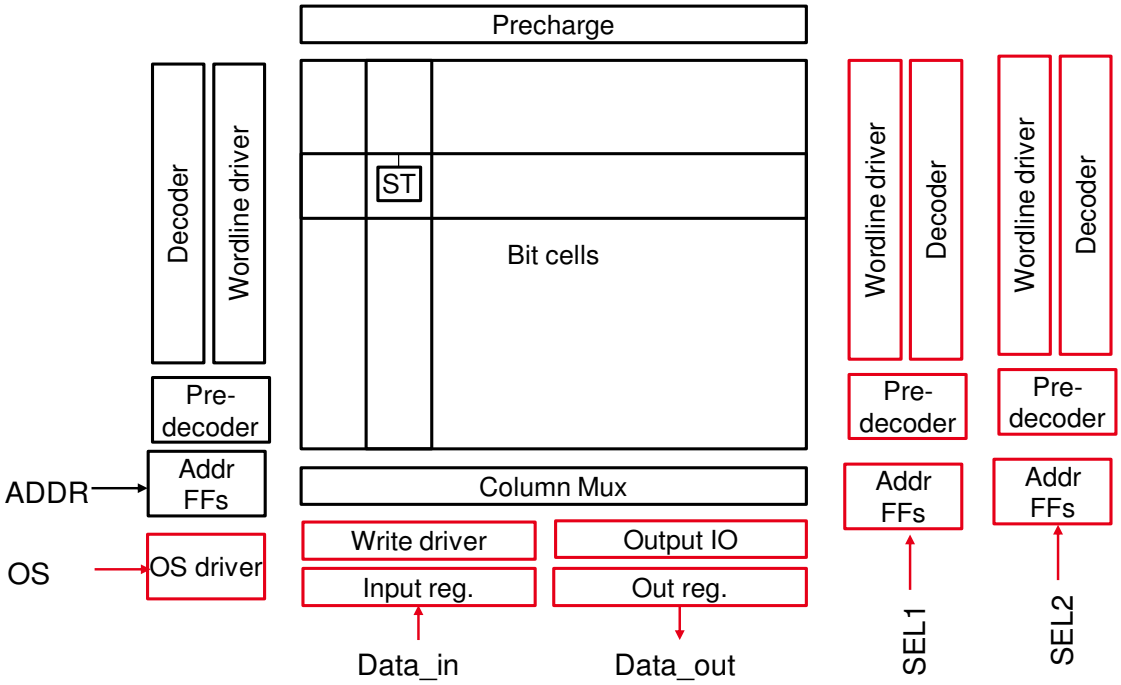


Figure 4.6: SmartMem structure with detailed blocks.

The detailed block diagram of the SmartMem organization is shown in Figure 4.6. It contains the similar modules as normal SRAM including bit cell array, address decoders, wordline drivers, column multiplexers, precharge circuits, write driver and output IO. SmartMem adds modified circuits for the write driver and the output IO to support smart operations. The bit cell array uses a 10T SRAM cell with two read ports and one write port as presented in Figure 4.2. To support for selecting two rows for smart operations, two additional address lines named SEL1 and SEL2 are

added. After decoded into the word line, two addresses are ORed with each other to form the SEL signal for the smart access. To control the smart operation, a new OS driver module is added. This module will be in charge of selecting the correct smart operation.

In summary, different logical and arithmetic operations can be performed with the special design of the bitcell and the IO circuit to facilitate the In-Memory-Computation capability. The data movement is reduced on account of the shortened datapath. Moreover, In-Memory Computing has fewer area overheads than the one in Processing in Memory because it uses the facilities of the memory to perform the operations instead of using dedicated hardware as in Processing in Memory. The next section presents the implementation of AES and PRESENT using the memory design.

4.3 Implementation of Advanced Encryption Standard and PRESENT using Encryption in memory

Specific hardware crypto-accelerators can provide ultra-low-power and ultra-low-cost security solutions for IoT applications. However, they are not adaptable to new standards and mitigate new threats. Flexibility and configurability can be provided by a generic solution using In-Memory computing. To improve the weakness of the proposal in Chapter 3, this section presents the implementation of AES and PRESENT based on the In-Memory Computing mechanisms which were discussed earlier in Section 4.2. In spite of implementing two algorithms using a specialized hardware structure, this work utilizes the logic operation performed in the memory to create a flexible and configurable implementation of the cryptography algorithms. The memory is different from the ASIC implementation that it has regular structures with well-defined components. On the other hand, memory is fast at looking-up operations while it is limited in computation. Therefore, this section will manage to use an alternative organization of the memory and the logic operations to implement AES and PRESENT. Accordingly, two algorithms will be transformed to utilize the advantages of the SmartMem architecture.

4.3.1 Advanced Encryption Standard

Advanced Encryption Standard (AES) is an algorithm optimized for modern computers with the 32-bit and larger registers. Therefore, it contains operations which can be performed efficiently using software. Consequently, mapping AES into the SmartMem requires careful consideration especially with the nonlinear operation of S-boxes. In addition, SmartMem is similar to normal memory, but it has the capability to calculate the logical operations efficiently. As a result, AES operations must be reorganized and transformed to use the advantage of the SmartMem. Figure 4.7 shows

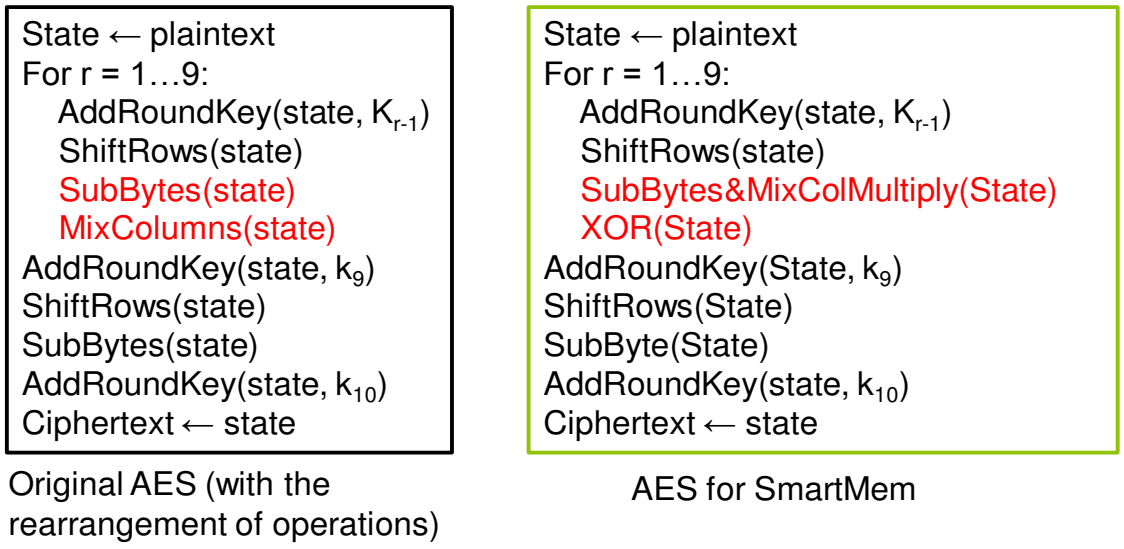


Figure 4.7: AES original algorithm and the one for SmartMem.

the original AES algorithm and its modification for being implemented on SmartMem. SubBytes and MixColumns are composed into SubBytes&MixColMultiply and followed by an XOR because SubBytes and MixColumns multiplications will be merged together and done by using Look-Up Table (LUT). After that, the results are XORed to get the equivalent results as the original AES. The LUT can be done easily using the memory while XOR is supported by SmartMem. The other steps are kept the same because these operations are supported by the SmartMem. For example, AddRoudKey is actually XOR operation while ShiftRows can be implemented by controlling the read and write address.

Furthermore, to generate the Look-up Table for SubBytes and MixColumns, the MixColumns is decomposed into two steps based on the method proposed by James A. Muir [Muir2013ato] in Figure 4.8. The output from SubBytes is multiplied with a metric with different coefficients as in Figure 4.8(a). A new byte is created by multiplying the output of SubBytes with different coefficients and then XOR them together. The LUT is generated by multiplying the SubBytes LUT with different multiplying coefficients. In the XOR step, the results will be combined as in Figure 4.8(b). The LUT values will be stored in the memory to be used as LUT tables.

The proposed organization of the SmartMem for implementing AES encryption is illustrated in Figure 4.9. The memory is composed of four small memories with 8-bit width. Each memory will store a copy of the LUT generated in the previous steps. The encryption controller will manage the address of each memory. Each memory contains the LUTs for Subbytes&MulcolMultiply. Four LUTs for Subytes&MulcolMultiply occupied 1024 words of each memory bank. The expanded keys could be reused by being generated in advance and being stored in the memory. This occupies 160 bytes in total and 40 bytes for each memory bank. Therefore,

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = x_0 \begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} \oplus x_1 \begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \oplus x_2 \begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} \oplus x_3 \begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix}$$

(a) Original MixColumns in AES

$$Ty_0(x) = x \cdot [02 \quad 01 \quad 01 \quad 03]^T$$
$$Ty_1(x) = x \cdot [03 \quad 02 \quad 01 \quad 01]^T$$
$$Ty_2(x) = x \cdot [01 \quad 03 \quad 02 \quad 01]^T$$
$$Ty_3(x) = x \cdot [01 \quad 01 \quad 03 \quad 02]^T$$

$$\text{results} = \text{Ty}_0(\text{x}_0) \oplus \text{Ty}_1(\text{x}_1) \oplus \text{Ty}_2(\text{x}_2) \oplus \text{Ty}_3(\text{x}_3)$$

(b) MixColumns Table-Lookup construction for In-Memory Operation

Figure 4.8: MixColumns for AES using SmartMem based on the method described in [Muir2013ato].

each memory bank has the size of 2048 words to store the LUTs, expanded keys, plaintexts, ciphertexts and temporary data. Table 4.3 summarizes the number of memories needed to store the LUTs, the expanded key, the S-boxes and the state. More than $2KB$ can be used to store plaintexts and ciphertexts.

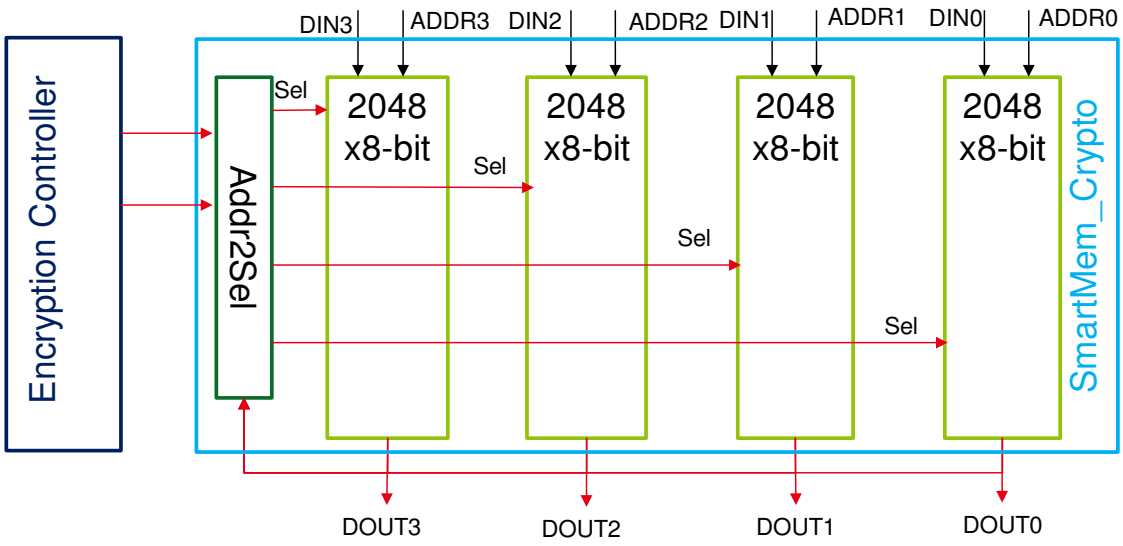


Figure 4.9: SmartMem organization for AES 32-bit encryption.

Table 4.1: Memory size to store the expanded key, look-up table for SubBytes and Mixcolumns of 32-bit datapath architecture

Name	Quantity	Size (byte)	Total size (byte)
Key storage (expanded key)	11	16	176
S-Box&MixcolMultiply table	4×4	256	4096
S-Box table	4	256	1024
Total			5296
Remaining for users' data			2896

At the startup phase, the LUTs and the expanded key will be written into each memory. After that, the plaintexts can be written into the memory and the encryption can start right after that. The address of the plaintexts, the expanded keys and the LUTS offsets can be programmed into the encryption controller using software. When the encryption process is activated, the encryption controller will take control of the memory interface to do the encryption.

After the memory is configured with the LUTs and the expanded key and the plaintexts are written into the memory, the encryption can be started by the encryption controller. Table 4.2 summarizes the mapping of AES operations into SmartMem operations. The first operation in the AES algorithm is AddRoundKey. This operation is the XOR of the plaintexts and the key, therefore, it is done by using the smart-write operation. The wordline containing plaintexts and the key will be selected and the read data will be applied with XOR. The output data is written back into the memory directly. The second operation is ShiftRows. ShiftRows is done by a normal read operation with the addresses of 4 memory banks form a diagonal of the state matrix. To read out the diagonal of the 4×4 matrix, the four addresses of the memory are added with 0, 1, 2, and 3, respectively at first. After that, these addresses are shifted to read the remaining ShiftRows values. Because ShiftRows are simply normal reads, they can also be integrated into Subbyte&MulcolMultiply. The last operation is Subbyte&MulcolMultiply which is carried out by using the output data from ShiftRows as the address to look up the corresponding values in the LUTs. The output data has to be added with the LUTs' offsets to select the correct LUTs for the operation. To further optimize the mapping process, ShiftRows, SubBytes and MixColumns are combined to use one normal read, one look-up operation and three smart-write operations for each 32-bit data.

Figure 4.10 shows the performance evaluation for AES implementation using a 32-bit datapath on SmartMem. It shows the number of clock cycles which each step needs. AddRoundkeys takes one cycle for 32-bit data and four cycles for 128-bit data. ShiftRows needs one clock cycle to perform the normal read for 32-bit data. SubBytes and MixcColumns need 3 XOR and one look-up, therefore, it needs four cycles for 32-bit datapath. In total, the AES mapping on SmartMem using 32-bit datapath requires 232 clock cycles to finish an encryption.

It is also possible to implement the AES encryption algorithm on a single mem-

Table 4.2: Mapping of AES operations into SmartMem operations

AES' operations	**SmartMem's operations**
AddRoundKey	Smart-Write: Selecting wordlines of plaintexts and corresponding key to apply XOR and write back.
ShiftRows	Normal read operation: Read each memory using diagonal addresses of a 4×4 matrix
Subbyte&MulcolMultiply	Three smart-write operations (smart-write and write back) and one normal read
SubBytes for the last round	Look-up operations

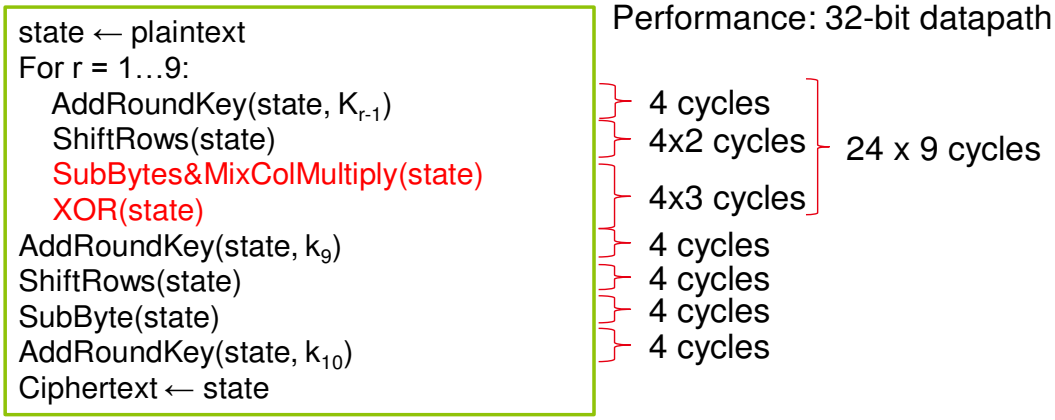


Figure 4.10: AES 32-bit datapath using SmartMem.

ory bank to reduce hardware area and power consumption. In addition, with this configuration, multiple banks can be used to do parallel encryption to improve the throughput. In this case, instead of using multiple memories, a single memory bank with a size of 2048 words is used. Figure 4.11 demonstrates the performance evaluation of AES with 8-bit datapath on SmartMem. In 8-bit datapath, 1024 bytes are used to store 4 LUTs of the SubByte&MixColMultiply. 160 bytes are used to store the expanded key. As a consequence, only 864 bytes are used to store plaintexts, temporary data and ciphertexts. Each step is performed on 1 byte. As a result, each step needs to be repeated 16 times. In summary, the 8-bit datapath architecture needs 874 clock cycles to finish one encryption. Table 4.3 presents the memory required to store the LUTs and data for encryption. In 8-bit datapath, a single 2048-word memory bank is used and 592 bytes are available for user data while 1456 bytes are used to store the LUTs and the expanded key.

In summary, it is possible to map AES algorithm using only SmartMem's operations. AES works on a byte data. Therefore, it can be implemented using a 4-byte

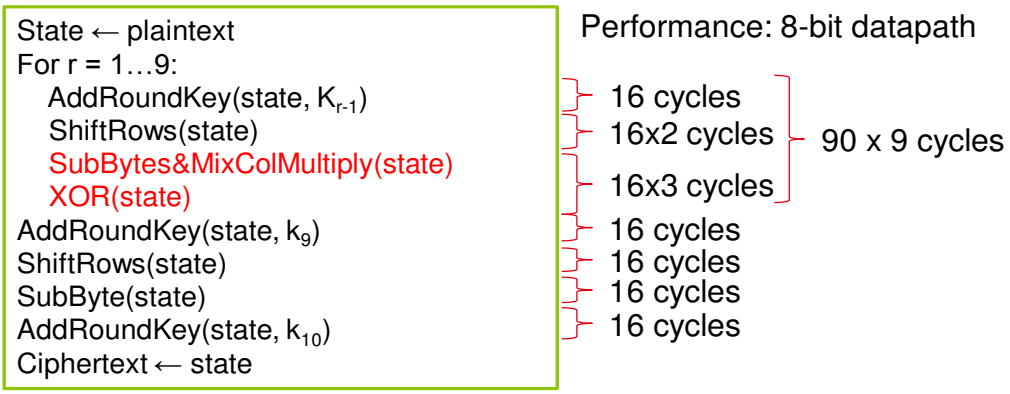


Figure 4.11: AES 8-bit datapath using SmartMem.

Table 4.3: Memory size to store the expanded key, look-up table for SubBytes and Mixcolumns of 8-bit datapath architecture

Name	Quantity	Size (byte)	Total size (byte)
Key storage (expanded key)	11	16	176
S-Box&MixcolMultiply table	4	256	1024
S-Box table	1	256	256
Total			1456
Remaining for users' data			592

datapath (32-bit datapath) or an 8-bit datapath. This section presents the detailed mapping of the AES operations into the SmartMem's operations. The encryption controller will control the read/write operations and smart-read/smart-write operations. The AES operations are decomposed into multiple operations of SmartMem, whereas shiftRows, SubBytes and MixColumns are combined into a normal read followed by a look-up and XORs. The mapping has been demonstrated using VHDL and the SmartMem model. The 32-bit datapath architecture of AES using four memory banks of 8-bit width takes 232 clock cycles for one encryption. The 8-bit datapath architecture using a single memory bank needs 874 clock cycle per encryption.

4.3.2 PRESENT

Lightweight cryptography algorithms can also be implemented using In-Memory Computing. Specifically, when being implemented in specific hardware crypto-accelerators using bit-based operations, lightweight cryptography utilizes the hardware constructs to reduce the hardware cost and power consumption. This is in contrast to conventional cryptography algorithms which are optimized for software implementation with operations on bytes. However, it is the bit-based operations that make lightweight cryptography less efficient in comparison with software implementation. Thus, this section is set to examine the possibility to implement

bit-based operations of cryptography algorithms using SmartMem through the implementation of PRESENT.

PRESENT is an algorithm optimized for hardware implementation. Implementing it using SmartMem operations is more complicated because PRESENT uses bit-based operations. PRESENT contains only two operations: S-box and wire permutation. Since, S-Box of PRESENT uses 4-bit S-Box, there will be a redundant to store them in the 8-bit memory. In addition, the wire permutation in PRESENT makes it less efficient in both software implementation and in SmartMem. Figure 4.12 demonstrates the permutation layer in PRESENT. The permutation is done at the bit level making it hard to implement PRESENT on SmartMem. Furthermore, the output of S-Boxes will be used as inputs to permutation. This leads to the idea of combining S-Boxes and permutation to use Look-Up Table [Benadjila2013ilb] for SmartMem.

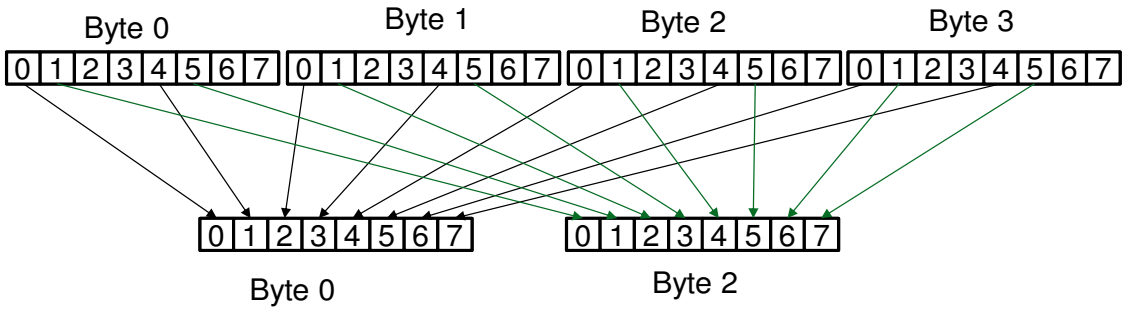


Figure 4.12: PRESENT bit permutation.

It can be seen from Figure 4.12 that the first byte after permutation is constructed by bit 0 and bit 4 of 4 input bytes, while the second byte after permutation is from bit 1 and bit 5 and so on. We can calculate the LUTs for PRESENT by using the following equation:

$$sbox\_permuted = sbox_0||sbox_4||sbox_1||sbox_5||sbox_2||sbox_6||sbox_3||sbox_7 \quad (4.1)$$

In equation 4.1, || is the bit concatenation operator. This means that each individual bit of the S-box is permuted using this order: 0, 4, 1, 5, 2, 6, 3, 7. Then, the look-up table can be generated using these equations:

$$LUT_1 = sbox\_permuted \quad (4.2)$$
$$LUT_2 = sbox\_permuted << 2 \quad (4.3)$$
$$LUT_3 = sbox\_permuted << 4 \quad (4.4)$$
$$LUT_4 = sbox\_permuted << 6 \quad (4.5)$$

In the above equation, << is the circular shift operator. The look-up tables are created by circularly shifting the *sbox_permuted* with a different number of bits.

After the look-up tables are generated, the organization of these look-up tables using the memory structure as described in section 4.3.1 is demonstrated in Figure 4.13. Each memory bank contains 4 LUTs of size 256 bytes and 4 bytes of bit masks. The purpose of the bit masks is to select only 2-bit at a time. These two bits, which are the outputs of the S-box after permutation, will be XORed with the expanded key, therefore, only selected bit are selected using these bit masks.

The encryption is carried out by selecting a word and then selectively applying the bit mask and applying the XOR. They are a combination of smart-read and smart-write operations. For example, to create the first byte of the output of a round, the input data was used as an address to look-up through the LUTs to select the first 2-bit of the first 4 bytes. Also, a mask is used to apply to each look-up, and the outputs after masking are ORed together to form the output byte. This is done repeatedly to get the output of a round.

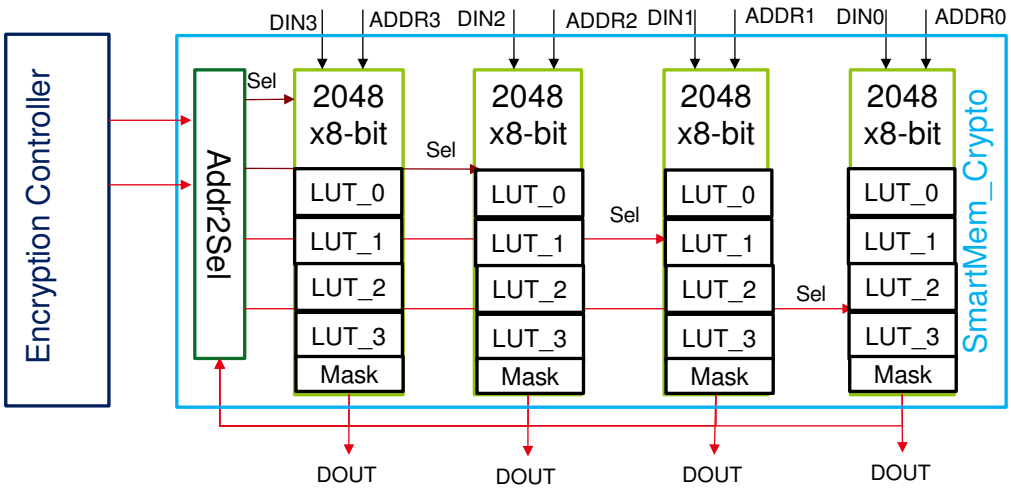


Figure 4.13: 32-bit architecture for PRESENT on SmartMem.

Table 4.4 summarizes the memory requirements to store the look-up tables and the expanded key for this implementation. 31 expanded keys must be stored in the memory along with 4×4 of 8-bit S-boxes-permuted tables. 4 Masks are also needed for each memory bank. This results in the total requirements of 4422 bytes of memory for 32-bit datapath architecture. With this memory configuration, 3770 bytes are remained for storing temporary data, plaintexts and ciphertexts. The total number of clock cycles to finish one encryption is 873 clock cycles. PRESENT on SmartMem employs a large number of clock cycles for one encryption because of its bit based permutation which leads to the use of bit masks for each encryption.

In summary, PRESENT was designed with the hardware construct to reduce the area and power consumption of the hardware implementation. However, when it comes to software implementation, bit masks have to be applied which brings about the inefficiency of the software implementation. For its implementation on SmartMem

Table 4.4: Memory size to store the expanded key, look-up table for substitution, permutation and masks of a PRESENT 32-bit datapath architecture

Name	Quantity	Size (byte)	Total size (byte)
Key storage (expanded key)	31	10	310
S-box 8-bit and permutation table	4×4	256	4096
Mask table	4	4	16
Total			4422
Remaining for users' data			3770

using 8-bit memory bank like the one in AES implementation, the S-boxes and the permutation can be combined into a look-up table. After that, bit masks can be applied to get the expected output. However, because of the use of a large number of bit masks operation, PRESENT with 32-bit architecture requires 873 clock cycles to finish one encryption of 64-bit data block. To encryption 128 bits of data, PRESENT will need 1746 clock cycles. Compared to AES implementation using Smartmem with only 232 clock cycles per 128 bits of data, PRESENT is nearly eight times slower than AES. In term of memory overhead, PRESENT uses about 800 bytes less than AES when implemented using SmartMem.

4.4 Conclusion

Crypto-accelerators can help achieve ultra-low-power consumption with low hardware cost, but it is also enclosed with some drawbacks in terms of security points of view. Crypto-accelerators implement a specific algorithm and have specific optimizations to get low power and small hardware footprint. However, the IoT standards continually evolve to cope with critical vulnerabilities and to mitigate security threats. Meanwhile, crypto-accelerators with fixed structures and algorithms cannot keep up with this constant change. In addition to that, today's widely used integrated circuits are critically vulnerable to hardware trojans which can be embedded by a third-party IP hardware vendor to monitor the system bus or to leak secret data.

In-Memory Computing is a new promising technology which can enable the in-place processing of information. In-Memory Computing uses the memory itself to do logical operations such as AND, OR, XOR and so on, and some basic arithmetic operations with the help of some extra circuits in the peripherals. This type of memory is not only applicable to accelerate different types of calculations but also employable to implement various security primitives. Security primitives implemented using this technology can reduce the risks of moving secret data through the system bus and the overhead of data transfer.

In-Memory Computing can be designed using various technologies such as Domain Wall Nano Wires, Non-Volatile Memory (NVM) or Complement Metal Oxide Semiconductor (CMOS). SmartMem was proposed by Akyel *et al.* and implemented

based on the concepts of In-Memory Computing using the current CMOS technologies. This has been seen as an innovative idea to move the computation close to the place where the data are stored. Some benefits could be claimed from mapping data encryption algorithm into the SmartMem, for instance, saving the memory bandwidth, saving the power consumption, and saving time to move the data from the memory to the processor or the crypto core and vice versa. However, under current conditions, it is challenging to map the algorithm using pure memory operation.

The SmartMem proposed by Akyel *et al.* uses 10T SRAM bitcells with one write port and two read ports to enable the smart operations. Specifically, two read ports can be used to select two different wordlines for smart operations. Also, logical operations are performed using the memory structure itself. Some other operations including XOR and arithmetic operations are performed by adding small extra circuits to the peripheral circuits of the memory.

In this chapter, the implementations of AES, a traditional cryptography algorithm, and PRESENT, a lightweight block cipher, into SmartMem are proposed using the SmartMem model. It is possible to implement them using the memory as a look-up table in combination with smart in-memory operations. The mapping involves using many sub-steps, hence, requires more clock cycles to finish. For AES with 32-bit datapath using four memory banks of 8-bit width, it needs 232 clock cycles for one encryption. PRESENT even needs 873 clock cycles to finish one encryption because of its bit-based permutation. In terms of throughput, PRESENT has nearly eight times slower than AES when being implemented in SmartMem using 32-bit datapath. In comparison with the crypto-accelerator solutions proposed in Chapter 3, the implementations in Memory are five times slower than the hardware crypto-accelerator for AES. Also, the obtained results are 27 times faster than the software implementation using ARM processor Cortex M0 as presented in [Zhang2018rar]. This indicates the potential of In-Memory Computing which needs further studies on the overall SmartMem architectures and system level designs, and more precisely the usage of them for security. In addition, the implementations in SmartMem can be updated to accommodate new requirements. This demonstrates the trade-offs of flexibility and configurability along with the conventional parameters including the cost, throughput and power consumption.

In addition, because of the serial operations of the memory, various existing countermeasures for software implementation are applicable to the In-Memory Computing. Multiple memory operations can be executed in parallel using different memory banks which support the implementation of masking countermeasure. In addition, code polymorphism can be applied to independent operations. However, this study is considered a preliminary work on using SmartMem to accelerate cryptography algorithms, living a promising direction for further investigations. Further researches could offer an in-deep exploration on power/energy estimation and security evaluation.

Conclusion and perspectives

Internet-of-Things, with the core idea of connecting millions of objects is delivering new business value and benefits, simultaneously accelerating new challenges, especially security constraints which have been highlighted throughout this manuscript. Evidently, IoT systems collect massive amount of private data, process them and send them through the Internet to the cloud which is supposedly a dangerous environment. Therefore, security must be a fundamental enabler, instead of an option for IoT.

However, implementing security for IoT is really challenging. The primary reason lies in IoT's association with multiple layers with different capabilities and its integration of thousands or millions of devices. In addition, IoT sensor nodes are expected to be long lifetime, ultra-low-power and ultra-low-cost devices, which makes it somehow vary from the underlying computer-based system or embedded systems. IoT backed devices might use battery-based power supply or self-harvest energy from the environment and have small memory footprint, adequate computation and limited power budget. Meanwhile, underlying security functions normally rely on strong cryptography algorithms such as block ciphers, hash functions, and/or public key cryptography which not only require complicated computations and large power consumption, but also reduce the system throughput. As a consequence, an optimal trade-off among security levels, hardware area and power/energy consumption must be taken into account when designing security functions for these devices.

Recent IoT proposals largely focus on using AES as the main security mechanism because AES has been studied for a long time with the security levels ranging from midterms to long terms. With advancements of cryptography, many new lightweight cryptography algorithms can also be employed to reduce the hardware cost and lower the power consumption. However, lightweight algorithms have not been chosen for the IoT proposals. Furthermore, different IoT applications might have diverse security profiles with different power/energy budgets. Therefore, the security solutions of a wide range of IoT applications need the configurability and flexibility. Depending on the security requirements of the applications, strong security mechanisms or lightweight ones can be selected to minimize the power consumption.

On the other hand, ultra-low-power security solutions for IoT applications gener-

ally focus on implementing the hardware accelerator for some specific algorithms to reduce the hardware cost and power consumption. Meanwhile, fixed hardware accelerators are not really adaptable to ever-evolving standards and threats. Therefore, flexible and configurable designs are needed to keep up with the evolution of IoT standards and security attacks. This helps prolong the lifetime of IoT applications.

This work investigates lightweight data encryption implementation for security in IoT in terms of power consumption and energy consumption. A block cipher module with two algorithms, AES and PRESENT were proposed with power optimization techniques. The proposed block cipher module provides multi-security levels with different key size ranging from 80 bits up to 256 bits. This module can also tailor specific requirements of the applications. The implementation results in SNACk testchip using FD-SOI $28nm$ technology from STMicroelectronics shows that this module is able to provide medium throughput of around $20Mbps$ at $10MHz$ and to consume less than $24\mu W$ at $0.6V$. At the subthreshold voltage of $0.4V$ and the operating frequency of $10MHz$, the proposed design consumes an energy of about $0.4pJ/bit$ with a power consumption of around $10\mu W$, which meets the demand of ultra-low-power consumption. Furthermore, the security evaluation based on Correlation Power Analysis and Test Vector Leakage Assessment using power-estimated traces shows that the proposed optimization does not introduce new leakages when compared with the reference design on Opencores. On the other hand, the block cipher accelerator might have low power consumption, but it still leaves some drawbacks, for instance, the data have to be transmitted from the memory to the system bus and the accelerator for processing. This will increase the power consumption of the system and possibly expose the data to other IP cores which monitor the bus. Therefore, this work continues to investigate an innovative method for doing data encryption which is called In-Memory Encryption. In-Memory Encryption uses a special type of memory which is capable of doing logical operation such as AND, OR, NOT and XOR. Using this type of memory, this work fully implemented the same algorithms but using only the infrastructure of In-Memory Computing. In-memory AES with 32-bit datapath needs 232 clock cycles for one encryption of 128 bits, and more than 873 clock cycles for 8-bit datapath. In the case of PRESENT with 32-bit datapath, because of its bit based hardware permutation, it requires more than 873 clock cycles for one encryption of a 64-bit block. In-memory PRESENT implementation is nearly eight times slower than in-memory AES to encrypt a 128-bit data block. Even with a large number of cycles when compared with the customized ASIC as in SNACk, they still show potentials in terms of security features. Based on this preliminary work on the implementation of cryptography algorithms using In-Memory Computing, further research works on power/energy consumption estimation, security evaluation and countermeasure designs should be conducted to evaluate the effectiveness of In-Memory Computing. Additionally, future studies at architecture levels and system levels are also important to improve the overall system

performance and security features.

There are a number of gaps in this study around adaptable security solutions, particularly flexible and configurable security solutions that calls for further researches. Accordingly, IoT applications should be proactive to different standards and capable of mitigating different attacks. However, the flexibility and configurability have to be traded-off with the throughput and power consumption; thereby the most feasible trade-off should be considered in order to improve the lifetime of devices. First of all, software implementations of security solutions can be updated using the firmware updates or software updates which provide flexibility. However, these approaches tend to work inefficiently in terms of throughput and power consumption. In addition, adaptability in hardware can improve throughput and power consumption when compared with software solutions. On the other hand, it is still less efficient than application specific implementations. Therefore, future applications should put forwards with the adaptability in design and implementation of security solutions in hardware to new standards or new threats to reduce hardware cost. Even renovating architectures from system levels down to device or transistor levels should be thoroughly considered.

In addition, this work does not take into account the countermeasures. Countermeasures are costly regarding power/energy consumption. However, their importance with regard to hardware-security has widely been acknowledged. Accordingly, ultra-low-power and ultra-low-cost countermeasures will be an interesting topic for future researches. In addition, the purpose of countermeasure is to cope with different types of attacks or analyses such as power analysis fault attack or so. Thus, countermeasures could employ additional hardware to facilitate the defense, which leads to a significant increase in hardware cost and power consumption. For this reason, they are currently not suitable for ultra-low-power and ultra-low-cost devices. Moreover, countermeasures for one type of attacks might not protect designs from the other types of attacks. Therefore, adaptive countermeasures should be prioritized for future practice on the IoT system. Furthermore, the cost and power consumption trade-offs should also be figured out.

Furthermore, In-Memory Computing sheds a new light on designing flexible security solutions as well as hardware security solutions. Accordingly, multiple countermeasures for various kinds of attacks including both power analysis attacks and fault attacks can be integrated using the current facilities of memory designs. For example, Error Correcting Code designs included in high reliable memories can be utilized to detect errors during the encryption in memory operation to deal with the fault attacks. In order to mitigate power analysis attacks, conventional methods such as masking or code polymorphism can be used at a small cost using In-Memory Computing. Also, masking eliminates the correlation of the key-dependent power consumption by running random operations at the same time with the key dependent operations. The parallel operations are supported by In-Memory Computing by default. Nevertheless, code polymorphism diminishes the correlation by reorder-

ing the operations of algorithms. Because of the serial operations of In-Memory Computing, code polymorphism can be easily deployed using this technology. These examples reveal the potentials of using In-Memory Computing to create adaptable solutions for both security algorithm mapping and countermeasures. This can be a promising research area to improve the overall security of a system without sacrificing power/energy consumption.

Bibliography

[Abid2009ecg] Z. Abid, A. Alma'aitah, M. Barua, and W. Wang. Efficient CMOL gate designs for cryptography applications. *IEEE Transactions on Nanotechnology*, 8(3):315–321, 05 2009.

[Achiya2018ikr] Achiya Bar-On, Orr Dunkelman, Nathan Keller, Eyal Ronen, and Adi Shamir. Improved key recovery attacks on reduced-round aes with practical data and memory complexities. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, pages 185–212, Cham, 2018. Springer International Publishing.

[Aes128Opencore] Hemanth Satyanarayana. AES crypto core.

URL `https://opencores.org/project,aes\_crypto\_core`

[Agrawal2003tes] Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. The em side—channel(s). In Burton S. Kaliski, çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, pages 29–45, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[Aguilar2013rai] C. Aguilar-Melchor, S. Fau, C. Fontaine, G. Gogniat, and R. Sirdey. Recent Advances in Homomorphic Encryption: A Possible Future for Signal Processing in the Encrypted Domain. *IEEE Signal Processing Magazine*, 30(2):108–117, March 2013.

[Akyel2016ddr] K. C. Akyel, H. P. Charles, J. Mottin, B. Giraud, G. Suraci, S. Thuries, and J. P. Noel. DRC2: Dynamically Reconfigurable Computing Circuit based on memory architecture. In *2016 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–8, October 2016.

[Al-Fuqaha2015iot] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys Tutorials*, 17(4):2347–2376, 2015.

[Albrecht2015cfm] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for mpc and fhe. In Elisabeth Oswald

and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, pages 430–454, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[Atzori2010tio] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The Internet of Things: A survey. *Computer Networks*, 54(15):2787–2805, October 2010.

[Banik2015EEE] Subhadeep Banik, Andrey Bogdanov, and Francesco Regazzoni. Exploring energy efficiency of lightweight block ciphers. In *Selected Areas in Cryptography – SAC 2015: 22nd International Conference, Revised Selected Papers*, pages 178–194. Springer International Publishing, 2016.

[Beaulieu2013tsa] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The simon and speck families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404, 2013. https://eprint.iacr.org/2013/404.

[Beaulieu2015tsa] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK lightweight block ciphers. In *Proceedings of the 52nd Annual Design Automation Conference on - DAC '15*, pages 1–6, San Francisco, California, 2015. ACM Press.

URL http://dl.acm.org/citation.cfm?doid=2744769.2747946

[Beigne2011ail] E. Beigné and P. Vivet. An innovative local adaptive voltage scaling architecture for on-chip variability compensation. In *2011 IEEE 9th International New Circuits and systems conference*, pages 510–513, June 2011.

[Bellovin2003smi] S. Bellovin, J. Schiller, and C. Kaufman. Security mechanisms for the internet, 2003.

[Benadjila2013ilb] Ryad Benadjila, Jian Guo, Victor Lomné, and Thomas Peyrin. Implementing lightweight block ciphers on x86 architectures. In Tanja Lange, Kristin Lauter, and Petr Lisoněk, editors, *Selected Areas in Cryptography – SAC 2013*, number 8282 in Lecture Notes in Computer Science, pages 324–351. Springer Berlin Heidelberg. DOI: 10.1007/978-3-662-43414-7_17.

URL http://link.springer.com/chapter/10.1007/978-3-662-43414-7_17

[Benini1994spb] L. Benini, P. Siegel, and G. De Micheli. Saving power by synthesizing gated clocks for sequential circuits. *IEEE Design Test of Computers*, 11(4):32–41, Winter 1994.

[Bertoni2004PAS] Guido Bertoni, Marco Macchetti, Luca Negri, and Pasqualina Fragneto. Power-efficient ASIC synthesis of cryptographic sboxes. In *Proceedings of the 14th ACM Great Lakes Symposium on VLSI*, GLSVLSI '04, pages 277–281. ACM, 2004.

[Bertoni2011tks] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. The keccak sha-3 submission. Submission to NIST (Round 3), 2011.

URL `http://keccak.noekeon.org/Keccak-submission-3.pdf`

[Bhasin2014nicv] S. Bhasin, J. Danger, S. Guilley, and Z. Najm. Nicv: Normalized inter-class variance for detection of side-channel leakage. In *2014 International Symposium on Electromagnetic Compatibility, Tokyo*, pages 310–313, May 2014.

[Biham1990dcd] Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. In *Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '90, pages 2–21, Berlin, Heidelberg, 1991. Springer-Verlag.

URL `http://dl.acm.org/citation.cfm?id=646755.705229`

[Bogdanov2007pau] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007*, number 4727 in Lecture Notes in Computer Science, pages 450–466. Springer Berlin Heidelberg, 2007.

[Borghoff2012pal] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçın. PRINCE – A Low-Latency Block Cipher for Pervasive Computing Applications. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, pages 208–225, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[Brakerski2012fhe] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 309–325, New York, NY, USA, 2012. ACM.

URL `http://doi.acm.org/10.1145/2090236.2090262`

[Brier2004cpa] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, pages 16–29, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[Canniere2009kak] Christophe De Cannière, Orr Dunkelman, and Miroslav Knežević. Katan and ktantan — a family of small and efficient hardware-oriented block ciphers. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009*, pages 272–288, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[Canright2005AVC] D. Canright. *A Very Compact S-Box for AES*, pages 441–455. Number 3659 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005.

[Canteaut2016sca] Anne Canteaut, Sergiu Carpov, Caroline Fontaine, Tancrède Lepoint, María Naya-Plasencia, Pascal Paillier, and Renaud Sirdey. Stream ciphers: A Practical Solution for Efficient Homomorphic-Ciphertext Compression. In *23rd International Conference on Fast Software Encryption (FSE)*, volume 9783 - LNCS (Lecture Notes in Computer Science) of *Fast Software Encryption 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016,*, pages 313–333, Bochum, Germany, March 2016. Springer.

URL `https://hal.archives-ouvertes.fr/hal-01280479`

[Chari2003ta] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski, çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, pages 13–28, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[Cho2009lco] Joo Yeon Cho. Linear cryptanalysis of reduced-round present. Cryptology ePrint Archive, Report 2009/397, 2009. https://eprint.iacr.org/2009/397.

[Dupuis2018pah] S. Dupuis, M. Flottes, G. Di Natale, and B. Rouzeyre. Protection against hardware trojans with logic testing: Proposed solutions and challenges ahead. *IEEE Design Test*, 35(2):73–90, April 2018.

[ECRYPT-II] European Network of Excellence in Cryptology II. Ecrypt ii yearly report on algorithms and keysizes, 2012.

URL `http://www.ecrypt.eu.org/ecrypt2/documents/D.SPA.20.pdf`

[FIPS-197] U.S National Institute of Standards and Technology. Advanced Encryption Standard, 2001.

[FreePDK45nm] North Carolina State University. Free pdk 45nm.

URL `{https://www.eda.ncsu.edu/wiki/FreePDK}`

[Gao2017acf] M. Gao, Q. Wang, M. T. Arafin, Y. Lyu, and G. Qu. Approximate computing for low power and security in the Internet of Things. *Computer*, 50(6):27–34, 2017.

[Gong2012kan] Zheng Gong, Svetla Nikova, and Yee Wei Law. Klein: A new family of lightweight block ciphers. In Ari Juels and Christof Paar, editors, *RFID. Security and Privacy*, pages 1–18, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[Goodwill2011atm] Gilbert Goodwill, Benjamin Jun, Josh Jaffe, and Pankaj Rohatgi. A testing methodology for side-channel resistance validation, 2011.

URL `\url{http://csrc.nist.gov/news\_events/non-invasive-attack-testing-wo`

[Guthaus2016oao] Matthew R. Guthaus, James E. Stine, Samira Ataei, Brian Chen, Bin Wu, and Mehedi Sarwar. OpenRAM: An Open-source Memory Compiler. In *Proceedings of the 35th International Conference on Computer-Aided Design*, ICCAD '16, pages 93:1–93:6, New York, NY, USA, 2016. ACM.

URL `http://doi.acm.org/10.1145/2966986.2980098`

[Hamalainen2006dai] P. Hamalainen, T. Alho, M. Hannikainen, and T.D. Hamalainen. Design and implementation of low-area and low-power AES encryption hardware core. In *9th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools, 2006. DSD 2006*, pages 577–583.

[Hutter2011acp] Michael Hutter, Martin Feldhofer, and Johannes Wolkerstorfer. A Cryptographic Processor for Low-Resource Devices: Canning ECDSA and AES Like Sardines. In Claudio A. Ardagna and Jianying Zhou, editors, *Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication*, pages 144–159, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[ISO-18033-3] ISO. Information technology – Security techniques – Encryption algorithms – Part 3: Block ciphers. ISO/IEC 18033-3:2005, International Organization for Standardization, Geneva, Switzerland, 2005.

[ISO-29167-1] ISO. Information technology – Automatic identification and data capture techniques – Part 1: Security services for RFID air interfaces. ISO/IEC 29167-1:2014, International Organization for Standardization, Geneva, Switzerland, 2014.

[ISO-29192-2] ISO. Information technology – Security techniques – Lightweight cryptography – Part 2: Block ciphers. ISO/IEC 29192-2:2012, International Organization for Standardization, Geneva, Switzerland, 2012.

[Jean2017bag] Jérémy Jean, Amir Moradi, Thomas Peyrin, and Pascal Sasdrich. Bit-Sliding: A Generic Technique for Bit-Serial Implementations of SPN-based Primitives. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems – CHES 2017*, Lecture Notes in Computer Science, pages 687–707. Springer International Publishing, 2017.

[Jeloka2016anc] S. Jeloka, N. B. Akesh, D. Sylvester, and D. Blaauw. A 28 nm Configurable Memory (TCAM/BCAM/SRAM) Using Push-Rule 6t Bit Cell Enabling Logic-in-Memory. *IEEE Journal of Solid-State Circuits*, 51(4):1009–1021, April 2016.

[Koblitz1978ecc] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48:203–209, 1987.

[Kocher1999dpa] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, pages 388–397, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[Kocher2018sae] Paul Kocher, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre Attacks: Exploiting Speculative Execution. *arXiv:1801.01203 [cs]*, January 2018. arXiv: 1801.01203.

URL `http://arxiv.org/abs/1801.01203`

[LRWPAN] IEEE Standardization Group. IEEE Standard for Local and Mtropolitan Area Networks – part 15.4: Low-rate wireless persional area networks (lr-wpans), 2011.

[Labbe2004ehi] A. Labbe, A. Perez, and J. M. Portal. Efficient hardware implementation of a CRYPTO-MEMORY based on AES algorithm and SRAM architecture. In *2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No.04CH37512)*, volume 2, pages II–637–40 Vol.2, 05 2004.

[LeSueur2010dvf] Etienne Le Sueur and Gernot Heiser. Dynamic voltage and frequency scaling: The laws of diminishing returns. In *Proceedings of the 2010 International Conference on Power Aware Computing and Systems*, HotPower'10, pages 1–8, Berkeley, CA, USA, 2010. USENIX Association.

URL `http://dl.acm.org/citation.cfm?id=1924920.1924921`

[Li2005anc] Hua Li. A new CAM based s/s-1-box look-up table in AES. In *IEEE International Symposium on Circuits and Systems, 2005. ISCAS 2005*, pages 4634–4636 Vol. 5, 05 2005.

[Lin2017aso] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao. A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications. *IEEE Internet of Things Journal*, 4(5):1125–1142, October 2017.

[Lipp2018m] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown. *arXiv:1801.01207 [cs]*, January 2018. arXiv: 1801.01207.

URL `http://arxiv.org/abs/1801.01207`

[Liu2011A2G] P. C. Liu, J. H. Hsiao, H. C. Chang, and C. Y. Lee. A 2.97 gb/s dpa-resistant aes engine with self-generated random sequence. In *Proceedings of the 2011 European Solid-State Circuit Conference (ESSCIRC)*, pages 71–74, Sept 2011.

[LoRaWan] Lora Alliance. LoraWan Specification, 2015.

[Lu2005rpf] Yongqiang Lu, C. N. Sze, Xianlong Hong, Qiang Zhou, Yici Cai, Liang Huang, and Jiang Hu. Register placement for low power clock network. In *Proceedings of the ASP-DAC 2005. Asia and South Pacific Design Automation Conference, 2005.*, volume 1, pages 588–593 Vol. 1, Jan 2005.

[Maene2015sio] Pieter Maene and Ingrid Verbauwhede. Single-cycle implementations of block ciphers. In *Lightweight Cryptography for Security and Privacy*, number 9542 in Lecture Notes in Computer Science, pages 131–147. Springer International Publishing, 2015.

[Mantin2005pad] Itsik Mantin. Predicting and distinguishing attacks on rc4 keystream generator. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, pages 491–506, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[Mathew20115GN] S.K. Mathew, F. Sheikh, M. Kounavis, S. Gueron, A. Agarwal, S.K. Hsu, H. Kaul, M.A. Anders, and R.K. Krishnamurthy. 53 gbps native composite-field AES-encrypt/decrypt accelerator for content-protection in 45 nm high-performance microprocessors. *IEEE Journal of Solid-State Circuits*, 46(4):767–776, 2011.

[Mathew20153m1] S. Mathew, S. Satpathy, V. Suresh, M. Anders, H. Kaul, A. Agarwal, S. Hsu, G. Chen, and R. Krishnamurthy. 340 mV – 1.1 v, 289 gbps/w, 2090-gate NanoAES hardware accelerator with area-optimized encrypt/decrypt $GF(2^4)^2$ polynomials in 22 nm tri-gate CMOS. *IEEE Journal of Solid-State Circuits*, 50(4):1048–1058, 2015.

[Matsui1994lcm] Mitsuru Matsui. Linear cryptanalysis method for des cipher. In Tor Helleseth, editor, *Advances in Cryptology — EUROCRYPT '93*, pages 386–397, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[Moradi2011ptl] Amir Moradi, Axel Poschmann, San Ling, Christof Paar, and Huaxiong Wang. *Pushing the Limits: A Very Compact and a Threshold Implementation of AES*, pages 69–88. Number 6632 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011.

[Moradi2014dhl] Amir Moradi, Sylvain Guilley, and Annelie Heuser. Detecting hidden leakages. In Ioana Boureanu, Philippe Owesarski, and Serge Vaudenay, editors, *Applied Cryptography and Network Security*, pages 324–342, Cham, 2014. Springer International Publishing.

[Mosenia2017acs] A. Mosenia and N. K. Jha. A Comprehensive Study of Security of Internet-of-Things. *IEEE Transactions on Emerging Topics in Computing*, 5(4):586–602, October 2017.

[Muir2013ato] James A. Muir. A tutorial on white-box aes. Cryptology ePrint Archive, Report 2013/104, 2013. https://eprint.iacr.org/2013/104.

[NIST-SP-800-98] National Institute Of Standards and Technology. *NIST Special Publication 800-98 Guidelines for Securing Radio Frequency Identification (RFID) Systems*. CreateSpace, Paramount, CA, 2007.

[Nangate2011OCL] Nangate Inc. Nangate freepdk45 open cell library, 2011.

URL {http://www.nangate.com/?page_id=2325}

[Pagliari2018fbb] D. J. Pagliari, Y. Durand, D. Coriat, E. Beigne, E. Macii, and M. Poncino. Fine-grain back biasing for the design of energy-quality scalable operators. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1–1, 2018.

[Rivest1978amf] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.

[Roy2016lmi] Debapriya Basu Roy, Shivam Bhasin, Sylvain Guilley, Annelie Heuser, Sikhar Patranabis, and Debdeep Mukhopadhyay. Leak me if you can: Does tvla reveal success rate? Cryptology ePrint Archive, Report 2016/1152, 2016. https://eprint.iacr.org/2016/1152.

[Rozic2012dsf] V. Rožić, W. Dehaene, and I. Verbauwhede. Design solutions for securing SRAM cell against power analysis. In *2012 IEEE International Symposium on Hardware-Oriented Security and Trust*, pages 122–127, June 2012.

[Satoh2001acr] Akashi Satoh, Sumio Morioka, Kohji Takano, and Seiji Munetoh. A compact rijndael hardware architecture with s-box optimization. In *Advances in Cryptology – ASIACRYPT 2001*, number 2248 in Lecture Notes in Computer Science, pages 239–254. Springer, Berlin, Heidelberg, 2001.

[Satpathy2018grg] S. Satpathy, V. Suresh, S. Mathew, M. Anders, H. Kaul, A. Agarwal, S. Hsu, and R. Krishnamurthy. 220mv-900mv 794/584/754 gbps/w reconfigurable gf(24)2 aes/sms4/camellia symmetric-key cipher accelerator in 14nm tri-gate cmos. In *2018 IEEE Symposium on VLSI Circuits*, pages 175–176, June 2018.

[Sayilar2014cht] G. Sayilar and D. Chiou. Cryptoraptor: High throughput reconfigurable cryptographic processor. In *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 155–161, Nov 2014.

[Schneider2015lam] Tobias Schneider and Amir Moradi. Leakage Assessment Methodology - a clear roadmap for side-channel evaluations. Technical Report 207, 2015.

URL https://eprint.iacr.org/2015/207

[Shirai2007t1b] Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai, and Tetsu Iwata. The 128-bit blockcipher clefia. In *Proceedings of the 14th International Conference on Fast Software Encryption*, FSE'07, pages 181–195, Berlin, Heidelberg, 2007. Springer-Verlag.

[Sinanovic2017aom] H. Sinanović and S. Mrdovic. Analysis of Mirai malicious software. In *2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–5, September 2017.

[Student1908tpe] STUDENT. The probable error of a mean. *Biometrika*, 6(1):1–25, 1908.

URL http://dx.doi.org/10.1093/biomet/6.1.1

[Suzaki2011tal] Tomoyasu Suzaki, Kazuhiko Minematsu, Sumio Morioka, and Eita Kobayashi. Twine: A lightweight, versatile block cipher. In *ECRYPT Workshop on Lightweight Cryptography*, pages 146–169, 2011.

[Wamser2017ptl] M. S. Wamser and G. Sigl. Pushing the limits further: Sub-atomic AES. In *2017 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 1–6, October 2017.

[Wang2016dad] Y. Wang, L. Ni, C. H. Chang, and H. Yu. DW-AES: A Domain-Wall Nanowire-Based AES for High Throughput and Energy-Efficient Data Encryption in Non-Volatile Memory. *IEEE Transactions on Information Forensics and Security*, 11(11):2426–2440, November 2016.

[Welch1947tgo] B. L. WELCH. The generalization of 'student's' problem when several different population varlances are involved. *Biometrika*, 34(1-2):28–35, 1947.

URL http://dx.doi.org/10.1093/biomet/34.1-2.28

[Yang2017aso] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao. A Survey on Security and Privacy Issues in Internet-of-Things. *IEEE Internet of Things Journal*, 4(5):1250–1258, October 2017.

[Yang2017hdf] K. Yang, D. Blaauw, and D. Sylvester. Hardware Designs for Security in Ultra-Low-Power IoT Systems: An Overview and Survey. *IEEE Micro*, 37(6):72–89, November 2017.

[Zhang2016AC4] Yiqun Zhang, Kaiyuan Yang, M. Saligane, D. Blaauw, and D. Sylvester. A compact 446 gbps/w aes accelerator for mobile soc and iot in 40nm. In *2016 IEEE Symposium on VLSI Circuits (VLSI-Circuits)*, pages 1–2, June 2016.

[Zhang2018rar] Y. Zhang, L. Xu, Q. Dong, J. Wang, D. Blaauw, and D. Sylvester. Recryptor: A Reconfigurable Cryptographic Cortex-M0 Processor With In-Memory and Near-Memory Computing for IoT Security. *IEEE Journal of Solid-State Circuits*, 53(4):995–1005, April 2018.

[Zhao2015nsa] W. Zhao, Y. Ha, and M. Alioto. Novel self-body-biasing and statistical design for near-threshold circuits with ultra energy-efficient AES as case study. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 23(8):1390–1401, 2015.

[Zigbee] Zigbee Alliance. Zigbee ip specification.

[dpacontestV3] Telecom ParisTech. Dpa contest v3.

URL http://www.dpacontest.org/v3

List of Abbreviations

6LoWPAN	IPv6 over Low-power Wireless Personal Area Network
AI	Artificial Intelligence
ALU	Arithmetic Logic Unit
APB	Advanced Peripheral Bus
ASIC	Application Specific Integrated Circuit
AU	Arithmetic Unit
AVFS	Adaptive Voltage Frequency Scaling
BLE	Bluetooth Low Energy
CAM	Content Addressable Memory
CBC	Coded Block Chaining
CCM	Counter with CBC-MAC
CMOL	CMOS-molecular nanoelectronics
CMOS	Complementary metal–oxide–semiconductor
CPA	Correlation Power Analysis
CPU	Central Processing Unit
DDoS	Distributed Denial of Service
DES	Data Encryption Standard
DNS	Domain Name Server
DNSSEC	Domain Name Server Security Extension
DPA	Differential Power Analysis

DRAM	Dynamic Random-Access Memory
DSE	Decode Switch Encode
DTLS	Datagram Transport Layer Security
DVFS	Dynamic Voltage Frequency Scaling
ECC	Eliptic Curve Cryptography
EDA	Electronic Design Automation
EM	Electromagnetic
FBB	Forward Back-Biasing
FD-SOI	Fully Depleted Silicon-On-Insulator
FIPS	Federal Information Processing Standards
FLL	Frequency-Locked Loop
FPGA	Field-Programmable Gate Array
FSM	Finite State Machine
GE	Gate Equivalent
GF	Galois Field or Finite Field
GSM	Global System for Mobile communication
GSS-API	Generic Security Services Application Program Interface
HMAC	Hash-based Message Authentication Code
ICG	Integrated Clock Gating
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Intellectual Property
IP	Internet Protocol
IPsec	Internet Protocol Security
IV	Initialization Vector
LAN	Local-Area Network

LOU	Logical Operation Unit
LPWAN	Low-Power Wide-Area-Network
LUT	Look-Up Table
MAC	Message Authentication Code
MIC	Message Integrity Code
N/A	Not Available
NIST	U.S National Institute of Standard and Technology
NMOS	N-Type Metal–Oxide–Semiconductor
NVM	Non-Volatile Memory
OS	Operations of SmartMem
PE	Processing Element
PGP	Pretty Good Privacy
PHY	Physical layer
PIM	Processing In Memory
PMOS	P-Type Metal–Oxide–Semiconductor
RBB	Reverse Back-Biasing
RBL	Read Bitline Left
RBR	Read Bitline Right
RCON	Round Constant
RFID	Radio Frequency Identification
RIRO	Round-Input Round-Output
ROM	Read-Only Memory
ROUT	Round-Output
RWL	Read Wordline Left
RWR	Read Wordline Right
S-box	Substitution box

S/MIME	Secure MIME
SASL	Simple Authentication and Security Layer
SDF	Standard Delay Format
SEL	Selection Line
SoC	Systems on Chip
SOUT	S-box Output
SPA	Simple Power Analysis
SPI	Serial Peripheral Interface
SPN	Substitution-Permutation Network
SRAM	Static Random Access Memory
SRU	Shifter-Rotator Unit
SSH	Secure Shell
SW	Software
TCAM	Ternary Content-Addressable Memory
TCP	Transmission Control Protocol
TLS	Secure Transport Layer
TLS	Transport Layer Security
TLU	Table Lookup Unit
TVLA	Test vector Leakage Assessment
ULP	Ultra-Low-Power
VLSI	Very Large Scale Integration
WBL	Write Bitline Left
WBR	Write Bitline Right
Wi-Fi	a technology for Wireless Local Area Network
WWL	Write WordLine

List of Figures

List of Tables