INTERNET OF THINGS HARDWARE SECURITY

By

Amr Mohamed Abbas Dessouky

A Thesis Submitted to the
Faculty of Engineering at Cairo University
In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE
In
Electronics and Electrical Communications Engineering

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2020

INTERNET OF THINGS HARDWARE SECURITY

By

Amr Mohamed Abbas Dessouky

A Thesis Submitted to the
Faculty of Engineering at Cairo University
In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE
In
Electronics and Electrical Communications Engineering

Under the Supervision of

Prof. Dr. Ahmed Nader

………………………………..

Professor of Electronics and Communications
Department of Electronics and Electrical Communications Engineering
Faculty of Engineering, Cairo University

Dr. Hassan Mostafa Hassan

………………………………..

Assistant Professor of Nano electronics, Bioelectronics and Optoelectronics
Department of Electronics and Electrical Communications Engineering
Faculty of Engineering, Cairo University

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2020

INTERNET OF THINGS HARDWARE SECURITY

By

Amr Mohamed Abbas Dessouky

A Thesis Submitted to the
Faculty of Engineering at Cairo University
In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE
In
Electronics and Electrical Communications Engineering

Approved by the
Examining Committee

____
Prof. Dr. First S. Name, External Examiner

____
Prof. Dr. Second E. Name, Internal Examiner

____
Prof. Dr. Third E. Name, Thesis Main Advisor

____
Prof. Dr. Fourth E. Name, Member

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2020

Engineer's Name:	Amr Mohamed Abbas Dessouky
Date of Birth:	06/04/1990
Nationality:	Egyptian
E-mail:	Amr_901@hotmail.com
Phone:	01007415072
Address:	
Registration Date:	01/10/2013
Awarding Date:	
Degree:	Master of Science
Department:	Electronics and Electrical Communications Engineering

Supervisors:

Prof. Dr. Ahmed Nader
Dr. Hassan Mostafa Hassan

Examiners:

Prof. …………………	(External examiner)
Prof. …………………	(Internal examiner)
Prof. …………………	(Thesis main advisor)
Prof. …………………	(Member)

Title of Thesis:

Internet of things hardware security.

Key Words:
Internet of things (IOT), Authenticated Encryption with associated data (AEAD), Dynamic Partial Reconfiguration (DPR)

Summary:
IoT devices are extremely vulnerable to attack, as they are tiny devices, and normally only possess intelligence which is enough to perform a single function, so that they can fit almost anywhere.
In this thesis, we focus on low area low energy hardware security for IoT, five lightweight cryptographic ciphers from The Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR) were optimized for low power low area on Field-programmable gate array (FPGA), namely NORX, SILC, COLM, Tiaoxin, and JAMBU.
Moreover, two new design methodologies using Partial dynamic reconfiguration was proposed in the thesis to achieve resource-efficient and energy-efficient hardware security.

Acknowledgments

Alhamdulillah, all praises and gratitude to Allah, for all his blessings and support me with the strength and health to complete this thesis.

I would like to thank my academic supervisors Dr. Hassan Mustafa and Dr. Ahmed Nader for their guidance and help during the thesis work.

I would like as well to thank my managers and colleagues in Mentor Graphics for their support (Eman El-Mandouh, Haytham Shoukry, Nahla Mohamed, Khaled Nouh and Islam Osama).

Last but absolutely not least, I want to extend my deepest and most sincere gratitude and thanks to my family for their support throughout my study.

Table of Contents

LIST OF Tables

LIST OF Figures

Nomenclature

Abbreviation	Description
Internet Of Things	IoT
Authenticated Encryption with Associated Data	AEAD
Encryption Standard	AES
Competition for Authenticated Encryption Security, Applicability, and Robustness	CAESAR
Pseudorandom Permutation	PRP
Encrypt then MAC	EtM
Encrypt and MAC	E&M
MAC then Encrypt	MtE
ciphertext-only attack	COA
known plaintext attack	KPA
Chosen plaintext attack	CPA
Adaptive chosen plaintext attack	ACPA
Chosen ciphertext attack	CCA
Adaptive chosen ciphertext attack	ACCA
National Institute of Standards and Technology	NIST
Addition-Rotation-XOR	ARX
Dynamic Partial Reconfiguration	DPR
Parallelizable Message Authentication Code	PMAC
Reconfigurable Modules	RM
Application Specific Integrated Circuit	ASIC
Internal Configuration Access Port	ICAP
Processor Configuration Access Port	PCAP
Programmable Logic	PL
Partial Reconfiguration Controller	PRC
Processing System	PS
Competition for Authenticated Encryption: Security, Applicability, and Robustness	CAESAR
Field-programmable gate array	FPGA
application-specific integrated circuit	ASIC

Abstract is too short. Please add more introduction items and also details on your achievements. Try to combine all the published papers abstracts. It should be at least one complete page

Abstract

IoT makes use of data collected from IoT devices to optimize the observation and control of the world in domains such as logistics, retail, military, and healthcare. IoT devices are extremely vulnerable to attack, as they are tiny, and normally possess intelligence which is enough to perform a single function, so that they can fit almost anywhere.

In this thesis, we focus on low area low energy hardware security for IoT, five lightweight cryptographic ciphers from The Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR) were optimized for low power low area on Field-programmable gate array (FPGA) and application-specific integrated circuit (ASIC), namely NORX, SILC, COLM, Tiaoxin, and JAMBU.

Moreover, two new design methodologies using Partial dynamic reconfiguration was proposed to in the thesis achieve resource-efficient and energy-efficient hardware security.

Chapter 1 : Introduction

The Internet of Things (IoT) has effectively spread through billions of tiny devices to every corner of the globe, they are incorporated into almost everything: televisions, shoes, cars, and light bulbs. It is also starting to appear in the processes of manufacturing and military technology. Although the functionality of the IoT devices is individually limited, they easily combine into huge data-collection networks that provide insights that were never available before. The internet of things links billions of devices to the internet and requires the use of billions of data points that must all be protected. IoT protection and IoT privacy are cited as major concerns because of its increased surface of attack.

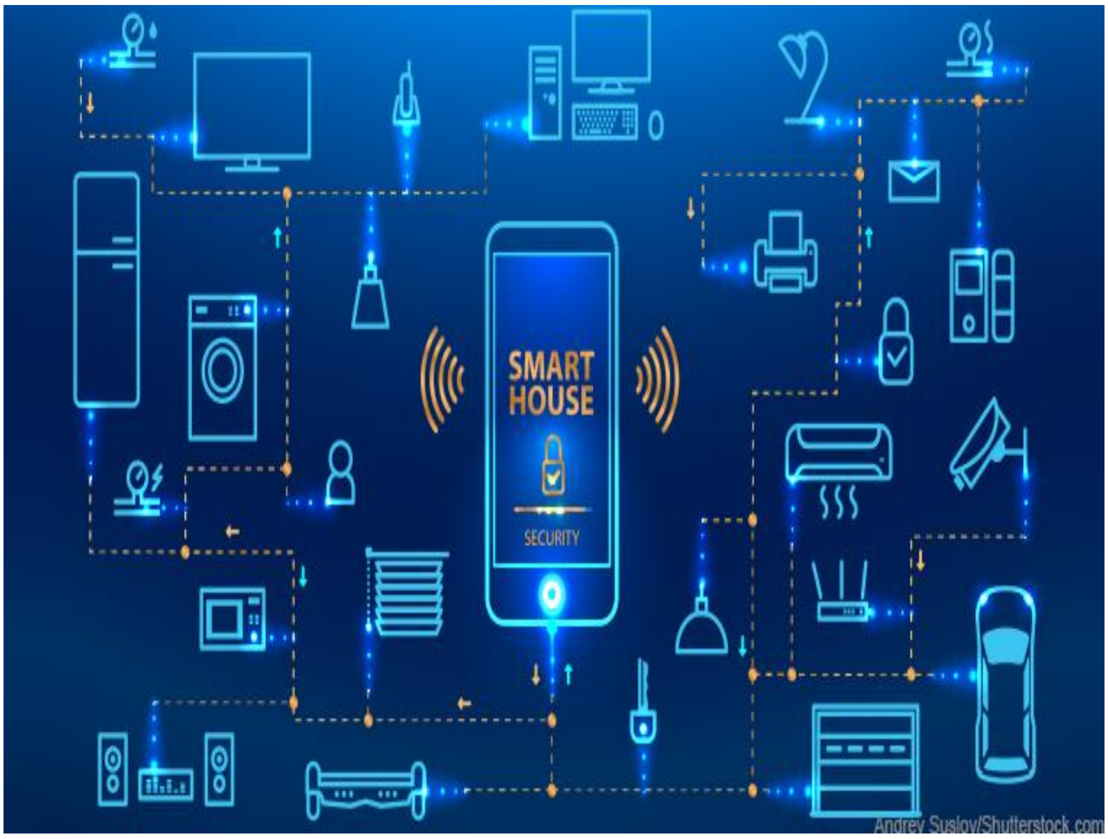


Figure 1 IOT Devices

IoT security concerns have attracted the attention of technology companies and government agencies around the world. The hacking of smart refrigerators, thermostats, baby monitors, cameras and even weapons are a security threat created by the future of IoT. Too many new nodes were added to the networks and the internet will provide malicious actors with countless attack vectors to carry out their evil deeds, particularly when many of them suffer from security holes.

This fear was realized with a massive distributed denial of service attack that crippled the servers of services like Twitter, Netflix, NY Times, and PayPal across the U.S. on October 21st, 2016. It's the result of an immense assault that involved millions of Internet addresses and malicious software, according to Dyn, the prime victim of that attack. "One source of the traffic for the attacks was devices infected by the Mirai botnet". The attack comes amid heightened cybersecurity fears and a rising number of Internet security breaches. Preliminary indications suggested that countless Internet of Things (IoT) devices that power everyday technology like closed-circuit cameras and smart-home devices were hijacked by the malware, and used against the servers. The more important shift in security will come from the fact that IoT will become more ingrained in our lives. Concerns will no longer be limited to the protection of sensitive information and assets. Our very lives and health can become the target of IoT hack attacks.

Since IoT devices are closely related, the only thing a hacker needs to do is exploit one vulnerability to access all of the data, rendering it. Furthermore, manufacturers who do not regularly upgrade their products or at all leave them vulnerable to cyber-attacks. Additionally, connected apps often ask users to enter their personal details, including names, ages, addresses, phone numbers, and even social media profiles, Information that is invaluable to hackers. However, hackers aren't the only threat to IoT, privacy is another big concern for IoT users. For instance, companies producing and selling consumer IoT devices may use those devices to collect and sell personal data from users.

A decision that system designers face in IoT field is deciding between software-based or hardware-based security solutions. The first solution to show up was software-based security which is relatively inexpensive as it shares resources with other programs to secure the data. The software-based implementation is capable of being revisited and upgraded as threats and vulnerabilities evolve. The software approach is the weak link within systems-security architecture because secrets remain vulnerable to discovery and the algorithms typically run on general-purpose non-secure hardware and are an attack risk. Hardware security is achieved through a dedicated integrated circuit (IC), or a processor with specialized security hardware, specifically designed to provide cryptographic functions. In Hardware Security, operations such as encryption/decryption and authentication, will take place at the IC hardware level where crypto algorithm performance is optimized.

IoT is extremely vulnerable to attack because they are tiny devices and usually have intelligence that is adequate to perform a single task to fit almost anywhere. Unfortunately, processors do not have the room to increase the processing capabilities needed for security. The fact is that because of the resource constraints, the biggest problem is the security and privacy issues of the huge amount of data being processed. Security and privacy insurance in IoT devices is very challenging because of low constraints which require innovations in both hardware and hardware and software. Lack of sufficient resources in terms of computing ability is one of the characteristics for Majority of the IoT devices [1], [2]. In addition, Form factor and cost play an important role, further limiting the overall capability of the IoT devices.

Recent advances in ultra-low-power technology enabled the development of smaller, autonomous, more mobile devices. Examples of this trend are smart cards, Radio Frequency Identification (RFID), and wearables. The power available to these devices is less than what common battery powered devices consume. Batteries for these devices are tiny and can supply 10Wfor only one day. Moreover, some of these technologies collect energy from environmental sources, such as light, heat, noise, or vibration using power scavengers which produce between 1 W and 500 W.

Conventional approaches such as advanced encryption standard (AES), though secure and robust, are not suitable for ensuring the integrity of data traveling among resource-constrained devices [3]. This raised the need for Authenticated ciphers which combine the cryptographic services of confidentiality, integrity, and authentication into one algorithm, they can potentially replace distinct block ciphers and hash functions that are required to work together, which both reduces resources, and eliminates potential security vulnerabilities. Authenticated encryption has historically been accomplished by the use of two different algorithms to encrypt and authenticate. Modes are being proposed recently which combine encryption and authentication together. This function is particularly useful for hardware implementations, as it enables a significant reduction in the area of the circuit and power relative to conventional schemes

The Competition for Authenticated Encryption Security, Applicability, and Robustness (CAESAR), evaluates candidates based on several criteria, including performance in hardware, to choose a portfolio of authenticated ciphers that offer advantages over AES-GCM, and are suitable for widespread adoption. The majority of these implementations were optimized for high speed (HS), in that they employed either basic iterative or unrolled architectures, and used full-width data paths and large I/O bus widths. Such design choices are not surprising, in that HW submissions are historically evaluated based on best throughput-to-area (TP/A) ratios, which are achieved using the aforementioned architectures [4].

1.2. Thesis Objectives

The objective of the work in this thesis is to provide a low area low power optimized implementation for cryptography algorithms to match the power constraints imposed by the low power IoT applications. The addressed algorithms are selected from algorithms that have participated in CAESAR. The selected algorithms are NORX, Tiaoxin, SILC, COLM, and JAMBU. The algorithms are implemented using the Field Programmable Gate Array (FPGA) flow and Application Specific Integrated Circuits (ASIC) flow. The Optimized implementations are benchmarked against the high-speed implementations.

Then we investigate the usage of dynamic partial reconfiguration in hardware encryption to have resource-efficient and energy-efficient hardware security.

1.3. Organization of the Thesis

The thesis presents different techniques for hardware security for IoT. The thesis is organized as follows.

Chapter 2 discusses the basic security objectives of any cryptographic system and then the methods that assure the security objectives. Later, we introduce the topic of Authenticated Encryption and its advantages over traditional schemes. Then we move to different types of Attacks on Ciphers. Finlay, we introduce the Competition for Authenticated Encryption: Security, Applicability, and Robustness.

Chapter 3 present the work done for low area low power optimization of five selected candidates namely NORX, SILC, COLM, Tiaoxin, and JAMBU, and compare the optimized results with the implementations available for high-speed implementations and those provided in other research. First, related work is presented, then we summarize the operation of each algorithm of the selected candidates and then present the optimization techniques applied. Finally, the results of optimized implementations are compared against the available high-speed implementations and other implementation provided in the research.

Chapter 4 presents Using Dynamic Partial Reconfiguration to achieve energy-efficient and resource-efficient Hardware Encryption, first, it gives an introduction for dynamic partial reconfiguration, and then it goes to how Partial dynamic reconfiguration could be used to implement resource-efficient and energy-efficient hardware encryption.

Finally, Chapter 5 contains the summary of achievements of the thesis. In addition, it provides potential directions for future work.

Chapter 2 : Introduction to Authenticated Encryption

2.1. Cryptography

Cryptography seeks to achieve information security and for this aim, it uses various techniques to achieve this objective. The main cryptographic objectives are described as follows

1. Confidentiality.
2. Data Integrity.
3. Authentication.
4. Non-repudiation of Message.
5. Availability

1) Confidentiality
It is a technique used against unauthorized disclosure to keep the data secure. In other words, if and only if the message's sender and the recipient can access the message's data then confidentiality is guaranteed. Confidentiality means that someone who is not part of the communication and unauthorized is not going to get access to the message information.

2) Data Integrity
This is a technique used to deal with the situation in which the unauthorized individual changes or alters the data. Changes to data include replacement, deletion, or addition of bits. Accordingly, data integrity guarantees the receiver can detect changes made by the unauthorized person on the message.

3) Authentication
This is identity-based technique of verifying a user's identity who wants to access the message, this form of authentication is called peer-entity authentication. Another form of authentication is authentication of data origin which ensures that the data originates from the actual sender and does not come from any third party.

4) Non-repudiation
This is a technique which prevents an entity from denying previous commitments or Actions. Non-repudiation protects both the recipient and the sender. Imagine receiving an email threatening anyone who denies sending the email. How can you figure the facts out? Digital signatures show that email messages are sent and received, ensuring, guaranteeing nonrepudiation.

5) Availability
This is a technique which provides access to data and techniques in the required manner without delay.

2.2. Encryption scheme

An encryption scheme is an algorithm that encrypts a message, called plaintext, to produce a new message, called ciphertext, using a key. The transformation is called encryption. On reverse transformation, called decryption, the ciphertext is decrypted with the same or a different key to produce the initial plaintext message (depending on the type of scheme). Ciphers history is a long one. Throughout history, there have been various types of cipher known, now classified as classical and modern.
Classical ciphers encrypt at letter level, they are classified into

1) Substitution ciphers, where the ciphertext replace the plaintext symbol
2) Transposition ciphers, where plaintext symbols are rearranged according to a defined scheme.

Modern ciphers operate on a lower bit level. They are further classified into the following

1. Symmetrical Encryption

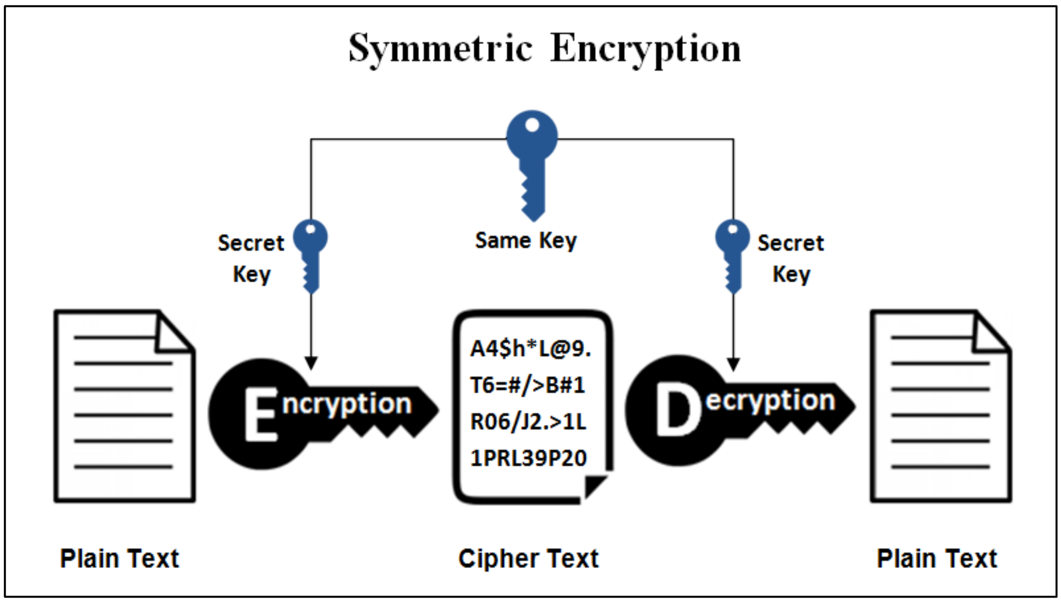


Figure 2: Symmetric Encryption [5]

This is the simplest method of encryption, involving only one secret key for ciphering and deciphering information. Symmetrical encryption is an old technique and is well known. It uses a secret key, which can be either a number, a word, or a random letter string. It is blended with the message's plain text to change the content in a specific way. The sender and the recipient should know the secret key that is used to encrypt and decrypt all the messages. Blowfish, AES, RC4, DES, RC5, and RC6 are examples of symmetric encryption. The most widely used symmetric algorithm are AES-128, AES-192, and AES-256.

The major drawback for symmetric key encryption is that all the interested parties share the key used in encryption to be used in decryption.

2. Asymmetrical Encryption

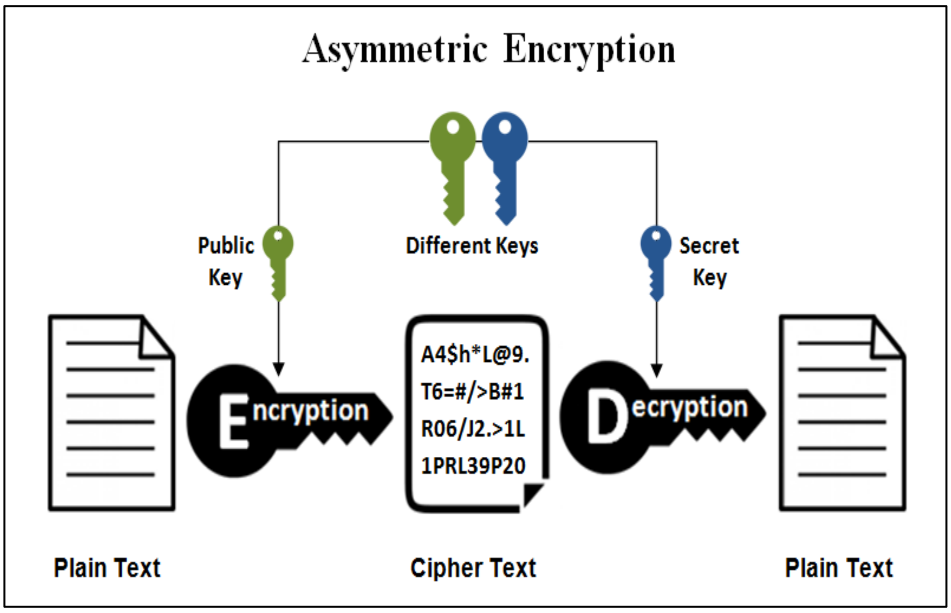


Figure 3: Asymmetric Encryption [5]

Asymmetric encryption is also known as public-key cryptography, compared to symmetric encryption it is a fairly new technique. For asymmetric encryption, two keys are used to encrypt a plain text. Secret keys are shared over the network. It ensures no abuse of the keys by malicious persons. It is important to remember that someone with a secret key can decrypt the message and this is why asymmetric encryption uses two related keys to improve security. Anyone who may want to send you a message will be given a public key free. The second private key is kept a secret for you to know only.

A message encrypted using a public key can only be decrypted using a private key, while using a public key can also decrypt a message encrypted using a private key. Public key protection is not needed because it is open to the public and can be transmitted over the internet. Asymmetric key has a far better power in ensuring the security of information transmitted during communication. Asymmetric encryption is mostly used in day-to-day communication channels, especially over the Internet. Popular asymmetric key encryption algorithm includes RSA, DSA, Elliptic curve techniques, PKCS.

Difference between Symmetric and Asymmetric Encryption

- Symmetric encryption uses a single key that needs to be shared between the people who need to receive the message while asymmetrical encryption uses a pair of public key and a private key to encrypt and decrypt messages when communicating.
- Asymmetric encryption is relatively new while Symmetric encryption is old technique.

- Asymmetric encryption has been implemented to counter the inherent issue of sharing the key in Symmetric encryption, removing the need to exchange the key using a pair of public-private keys.
- Asymmetric encryption is slower than the symmetric encryption.

2.3. Authenticated Encryption

Authenticated encryption has historically been accomplished by the use of two different algorithms to encrypt and authenticate. Modes are being proposed recently which combine encryption and authentication together. This function is particularly useful for hardware implementations, as it enables a significant reduction in the area of the circuit and power relative to conventional schemes.

Authenticated encryption (AE) is primarily a combination of authentication and encryption that provides both privacy and authenticity of the data that is encapsulated. Authenticated encryption ciphers take a message (M), an associated data (AD), a public message number (Npub), and an optional secret message number (Nsec) as an input and generate resulting ciphertext (C) and optional encrypted (Nsec). Integrity of data and authenticity of sender are ensured by a keyed-hash computation which occurs on all blocks of (Npub), (AD) and (M).

The result of these computations is forwarded to the recipient as a Tag, as shown in Figure.4. In authenticated decryption, the recipient receives original (AD) and (Npub), along with (C) and (Tag), and uses Key to decrypt (C) to (M). The authenticated decryption recreates a Tag (Tag'), and releases the ciphertext if and only if Tag = Tag', then authentication and integrity of the transaction are assured, otherwise the decrypted ciphertext is not released.

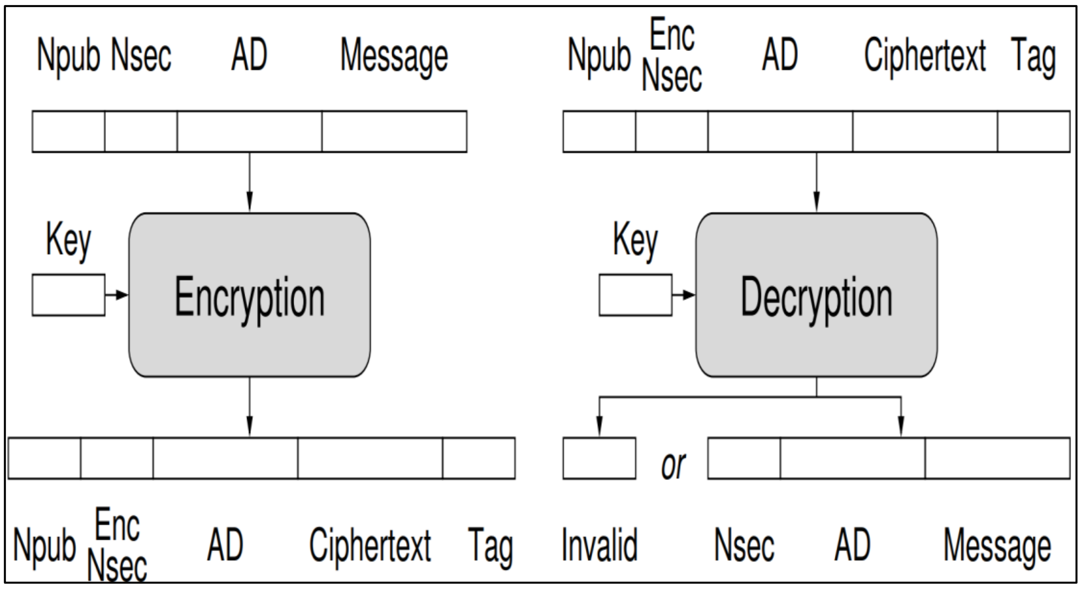


Figure 4: Input and Output of an Authenticated Cipher [6]

Combining authentication and encryption into one single algorithm in hardware might possibly provide the advantages listed below

- Area requirement could be smaller for a single algorithm there by reducing the cost.
- It is a good option for low-power applications where designs with smaller area requirements are needed.
- A combined algorithm needs only a single key and so has an advantage in key storage and key management.

2.4. Authentication Techniques

This section covers a detailed explanation of authentication techniques.

2.4.1. Cryptographic Hash Functions

A hash function which is identical to a checksum. The key difference is that while the purpose of a checksum is to detect unintentional changes in the data, a cryptographic hash function is designed to detect deliberate alterations. When a cryptographic hash function processes the data, a short string of bits, known as a hash, is generated. Typically the smallest change to the message causes a major difference in the resulting hash. There is no cryptographic key needed for a cryptographic hash function. The basic requirements of a hash function are as listed below.

1) One-way function.
2) Could be computed easily.
3) Fixed length Output.

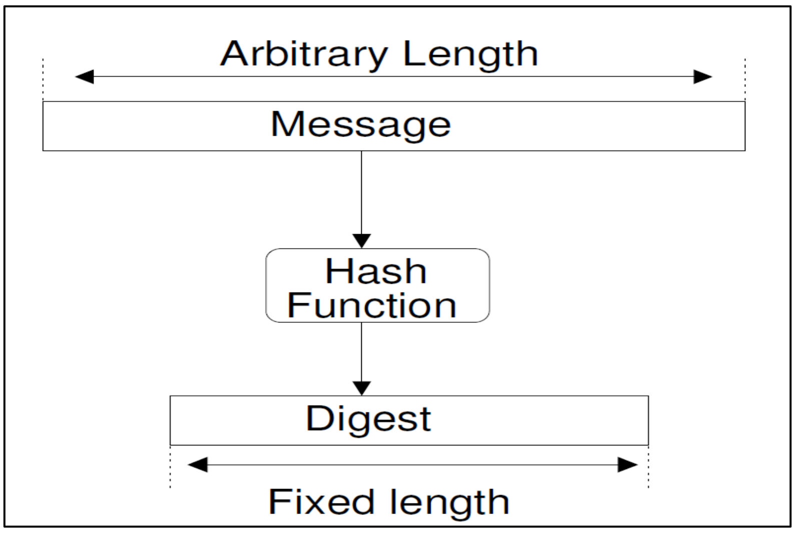


Figure 5: Cryptographic Hash Functions [7]

2.4.2. Message authentication code (MAC)

A message authentication code (MAC) is a cryptographic checksum on data that uses a session key to detect both accidental and intentional modifications of the data. A MAC needs two inputs: a message and a secret key known only to the message's originator and its intended receiver(s). It helps message recipients to check the message’s integrity and to authenticate that the sender of the message has the shared secret key. If a sender does not know the secret key, then the hash value will be different, which would inform the receiver that the message did not come from the original sender. The basic requirements of a MAC are as listed below

1) It should contain a key.
2) Fixed length output.
3) It must be computationally easy.

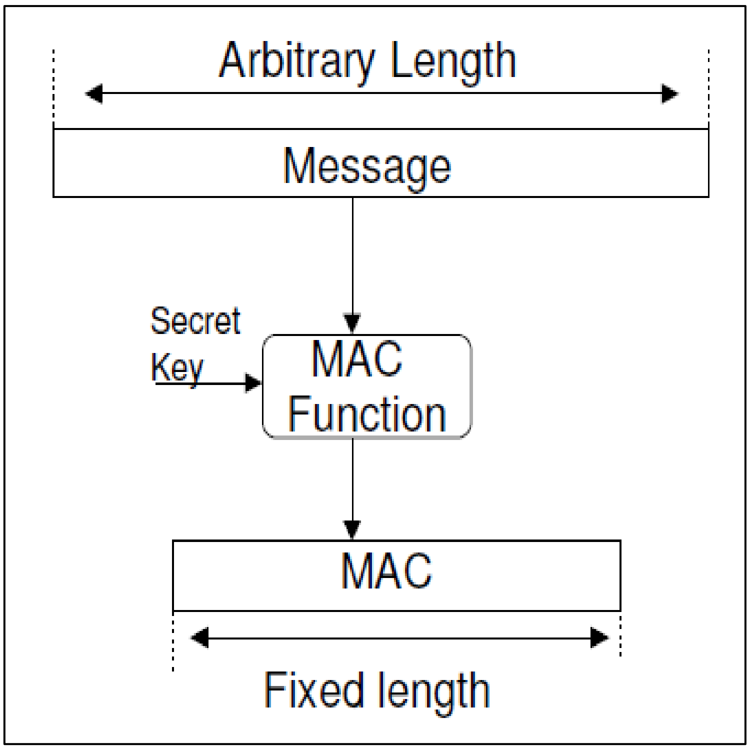


Figure 6 Message authentication code

Based on the way they are built there are two types of MACs

1) Hash function based MACs.

It is a special form of message authentication code (MAC) which includes a cryptographic hash function and a cryptographic secret key. It can check at the same time both the integrity of data and the authentication of a message. Any cryptographic hash function, such as SHA256 or SHA-3, can be used to compute an HMAC, the resulting MAC algorithm is called HMAC-X, where X is the hash function used.

2) Block cipher based MACs.
It is a technique for constructing a message authentication code from a block cipher. The message is encrypted with some block cipher algorithm in a certain way to create blocks chains, so that each block depends on the previous block's proper encryption. This interdependence ensures that a change to any of the plaintext bits will cause the final encrypted block to change in an unpredictable way that cannot be predicted or counteracted without knowing the block cipher key.

2.5. Authentication Ciphers Types

An Authenticated Cipher can be based on Block Ciphers, Tweakable Block ciphers, Stream Ciphers.

2.5.1. Block ciphers

A block cipher is an encryption method that applies a deterministic algorithm along with a symmetric key to encrypt a block of text. For example, a common block cipher, AES, encrypts 128 bit blocks with a key of predetermined length: 128, 192, or 256 bits. Block ciphers are pseudorandom permutation (PRP) families that operate on the fixed size block of bits. PRPs are functions that cannot be differentiated from completely random permutations and thus, are considered reliable, until proven unreliable.

Block cipher operation modes have been developed to eliminate the chance of encrypting identical blocks of text the same way, the ciphertext formed from the previous encrypted block is applied to the next block. A block of bits called an initialization vector (IV) is also used by modes of operation to ensure cipher texts remain different even when the same plaintext message is encrypted multiple times. Some of the various modes of operation for block ciphers include

- CBC (cipher block chaining)
- CFB (cipher feedback)
- CTR (counter)
- GCM (Galois/Counter Mode).

Popular block ciphers

- **DES**

DES used to be the most popular block cipher in the world which stands for Data Encryption Standard. It's not used widely nowadays but still popular because it's usually included in the encryption historical discussions. The DES algorithm became a standard in the US in 1977. However, it's already been proven to be vulnerable to brute force attacks and other cryptanalytic methods. DES works with a 64-bit key and it is a 64-bit cipher. Actually, the key size is technically 56 bits long as 8 of the 64 bits in the key are parity bits.

- **3DES**

As its name implies, 3DES is a cipher based on DES. It's practically DES that's run three times. Each DES operation can use a different key, with each key being 56 bits long. 3DES has a block size of 64 bits Like DES. Although 3DES is many times stronger than DES, it is also much slower (about 3x slower). It never became the ultimate successor of DES because many organizations found 3DES to be too slow for many applications.

- **AES**

A US Federal Government standard since 2002, AES or Advanced Encryption Standard is arguably the most widely used block cipher in the world. It has a block size of 128 bits and supports three possible key sizes - 128, 192, and 256 bits. The longer the key size, the stronger the encryption. However, longer keys also result in slower processes of encryption.

- **Blowfish**

This is another popular block cipher. It has a block size of 64 bits and supports a variable-length key that can range from 32 to 448 bits. Blowfish is unpatented and royalty-free which makes it so appealing.

- **Twofish**

This cipher is related to Blowfish but it's not as popular. It's a 128-bit block cipher that supports key sizes up to 256 bits long.

2.5.2. Stream Cipher

A stream cipher is an encryption algorithm that encrypts 1 bit or byte of plaintext at a time. The key of this cipher is infinite stream of pseudorandom bits. The key should never be reused and the pseudorandom generator should be unpredictable so that the cipher implementation remain secure. Stream ciphers are designed to approximate an idealized cipher, known as the One-Time Pad.

The One-Time Pad, which is supposed to employ a purely random key, can potentially achieve "perfect secrecy". That is, it's supposed to be fully immune to brute force attacks. The problem with the one-time pad is that, its key should be as long as or even longer than the plaintext in order to create such a cipher. In other words, if you have 500 Megabyte video file that you need to encrypt, a key that's at least 4 Gigabits long is needed. Stream ciphers can be divided in two major groups:

1) Synchronous and self-synchronous.
   These ciphers generate key streams independently and separately from the plaintext. To form the cipher text the key stream is then combined with the plaintext.

2) Self-synchronizing and asynchronous
   Stream cipher is a stream cipher in which the key stream is a function of the key and a fixed number of previous cipher text characters.

Popular stream ciphers

- **RC4**

RC4, which stands for Rivest Cipher 4, is the most widely used of all stream ciphers, particularly in software. It's also known as ARCFOUR or ARC4. RC4 has been used in various protocols like WEP and WPA (both security protocols for wireless networks) as well as in TLS. Unfortunately, recent studies have revealed vulnerabilities in RC4, prompting Mozilla and Microsoft to recommend that it should be disabled where possible.

2.5.3. Tweakable Block ciphers

A tweakable block cipher accepts a second input with its usual plaintext or cipher text input called the tweak. The tweak, along with the key, selects the permutation computed by the cipher. There are have been many proposed constructions Most of which rely on a block cipher, and generically introduce the tweak.

2.6. Authenticated Encryption approaches

There are three main approaches which are adopted for AEAD:

- Encrypt then MAC (EtM)
- Encrypt and MAC (E&M)
- MAC then Encrypt (MtE)

2.6.1. Encrypt-then-MAC (EtM)

In this scheme a message is first encrypted and the tag is calculated by taking the MAC over the obtained cipher text. In addition, on the receiver's side first the tag gets verified and if it matches decryption will take place to get the plaintext.

2.6.2. MAC-then-Encrypt (MtE)

In this scheme first the tag is calculated by taking the MAC over the message. The obtained is tag is then appended to the message and the resultant is encrypted to generate the cipher text. In addition, on the receiver's side first decryption will takes place to get plaintext and tag pair, and then verifies the tag.

2.6.3. Encrypt-and-MAC (E&M)

The message is encrypted to get the cipher text and the tag is also calculated on the original message. In addition, on the receiver's side first decryption is done to get the plaintext and then verifies the tag.

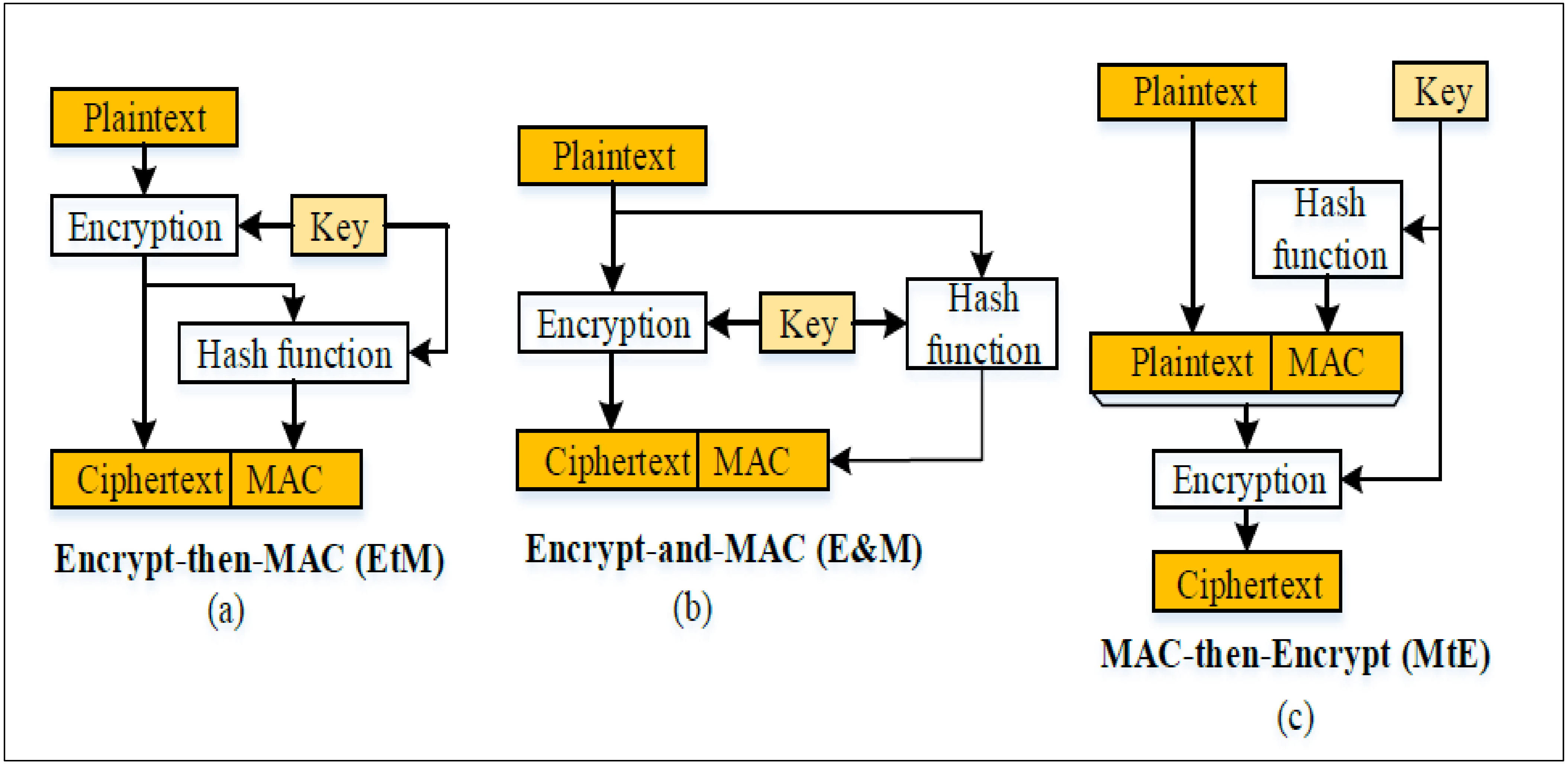


Figure 7: Authenticated Encryption approaches [15]

2.7. Attacks on ciphers

Total breaking of an encryption scheme means that an attacker can retrieve the secret encryption/decryption key and this way decrypt the cipher text. A cipher is partially broken if the attacker is able to retrieve part of the plaintext (but not the key) from cipher text.
Following is classification for the different types of Attacks.

1) A ciphertext-only attack (COA):

During ciphertext-only attacks, the attacker has no idea what the plaintext data or the secret key may be, he has access only to a number of encrypted messages. The goal is to guess the secret key or to recover as much plaintext messages as possible. It will be possible to break all the other messages which have been encrypted by this key if the encryption key is discovered.

It is particularly important to secure the encryption algorithms against ciphertext-only attacks while designing them, as they are the most obvious starting point for every cryptanalysis. Well prepared and reviewed ciphers are usually not very vulnerable to these attacks. However, there are some examples of protocols that have been broken by attacks based on the ciphertext-only approach. There are a few techniques which are based only on the knowledge of the ciphertext messages and were proved to be very effective even when targeting modern ciphers. The most important methods are:

- Attack on Two-Time Pad
- Frequency Analysis

2) A known plaintext attack (KPA):

During these attacks, the attacker has an access to the plaintext and its corresponding ciphertext. His goal is to guess the secret key or to develop an algorithm which make him able to do decryption for any further messages.
This gives the attacker much bigger possibilities to break the cipher than just by performing only attacks. However, he is no able to actively provide customized data or secret keys which would be processed by the cipher.

3) Chosen plaintext attack (CPA):

During the chosen-plaintext attack, a cryptanalyst can choose arbitrary plaintext data to be encrypted and then he receives the corresponding ciphertext. He tries to acquire the secret encryption key or alternatively to create an algorithm which would allow him to decrypt any ciphertext messages encrypted using this key (but without actually knowing the secret key).

This is a more comfortable situation for the attacker. He can get more information about the secret key and about the attacked system, because he has the ability to choose any plaintext to be processed by the system. Based on any kind of input data, he can analyze the system behavior and output ciphertext, based on any kind of input data. During breaking deterministic ciphers with the public key, the intruder can easily create a database with popular ciphertexts. After that by simply comparing many intercepted encrypted messages with his own database entries, he will be able to find the meaning of many them.

4) An adaptive chosen plaintext attack (ACPA):

In this kind of chosen-plaintext attack, the attacker has the ability to choose plaintext for encryption many times. Instead of using one big block of text, he can choose the smaller plaintext, get its encrypted ciphertext and then based on it, choose another one, and so on. This allows him to study the attacked system in much more details.

5) Chosen ciphertext attack (CCA):

During the chosen-ciphertext attack, a cryptanalyst can analyze any chosen ciphertexts together with their corresponding plaintexts. His goal is to acquire a secret key or to get as many information about the attacked system as possible. The attacker has the ability to make the victim decrypt any ciphertext and send him back the ciphertext. By analyzing the chosen ciphertext and the corresponding received plaintext, the attacker tries to guess the secret key which has been used in the decryption. Chosen-ciphertext attacks are used usually for attacking public key encryption systems. For example, The RSA cipher were vulnerable to such attacks. They are used less often for attacking symmetric ciphers. Some self-synchronizing stream ciphers have been also attacked successfully in that way.

6) An adaptive chosen ciphertext attack (ACCA):

The adaptive-chosen-ciphertext attack is a kind of chosen-ciphertext attacks, during which an attacker can make the attacked system decrypt many different ciphertexts. This means that the new ciphertexts are decrypted based on responses (plaintexts) received previously. The attacker has the ability to request decrypting of many ciphertexts. This model is used for the analysis of the security of encryption system. Proving that this attack doesn't break the system confirms that any realistic chosen-ciphertext attack will not be able to break the system.

2.8. Attack analysis: differential and linear cryptanalysis

A CPA or CCA gives the attacker the freedom to analyze a cipher from plain-text cipher pairs. Adaptive cases give him even more power, since he can use certain dependencies from previous queries. These types of attacks allow the collection of certain advantageous data that can be analyzed and used for the following types of cryptanalysis.

1) Linear cryptanalysis

It exploits correlations between a two linear function, the first one is the function of the input blocks and the second one is the function of the output blocks. The two linear functions combination is called linear approximation. The most widely used linear function involves calculating the bitwise dot product block operation with a specific binary vector. The value combined with the input blocks is called the input mask, while that applied to the output blocks is called the output mask.

2) Differential cryptanalysis

It takes advantage of how a specific difference in a pair of inputs of a cipher can affect a difference in the pair of outputs of the cipher, where the pair of outputs are obtained by encrypting the pair of inputs using the same key. The notion of difference can be defined in several ways; the most widely used is using XOR operation. The difference between the inputs is called the input difference, and the difference between the outputs is called the output difference. The combination of the input difference and the output difference is called a differential.

2.9. Competition for Authenticated Encryption: Security, Applicability, and Robustness

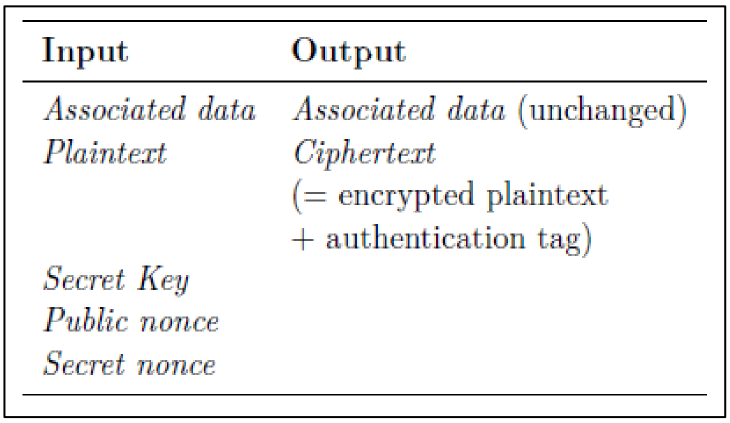
In 1970 public developments made high quality cryptography accessible to the general public. Governments tried to keep their monopoly on it, and until this day there are still laws restricting the export of cryptography. The United States relaxed their laws on the export of cryptography around the year 2000. At the time, the US National Institute of Standards and Technology (NIST) announced an open competition to find an unclassified, publicly disclosed encryption algorithm capable of protecting government sensitive information well into the next century. The winning algorithm of that competition (Advanced Encryption Standard or AES), is still one of the most used symmetric-key algorithms today.

Based on the same open principles, several other competitions, were organized in the new millennium. ESTREAM, organized from 2004 to 2008, resulted in a portfolio of seven stream ciphers [8]. Later, the SHA-3 competition was organized to find a new hash algorithm to augment and revise the SHA-1 and SHA-2 standards [9].
The competition was organized because of fears that SHA-2 would be broken. Due to theoretical attacks on SHA-1 [10, 11] and successful attacks on MD5, there was no other backup algorithm. The competition went from 2007 to 2012.

In 2013, The CAESAR Competition for Authenticated Encryption: Security, Applicability, and Robustness was announced in order to encourage the design of AE algorithms. In round 1, 57 different proposals have been submitted to the competition. Out of these 57 submissions, only 28 submissions qualified to the second round. FPGA implementations of all the 28 candidates have been developed and benchmarked for comparison. In September 2016, 15 candidates have been selected for the third round of the competition. In 2019 the final portfolio was announced which includes 6 ciphers.

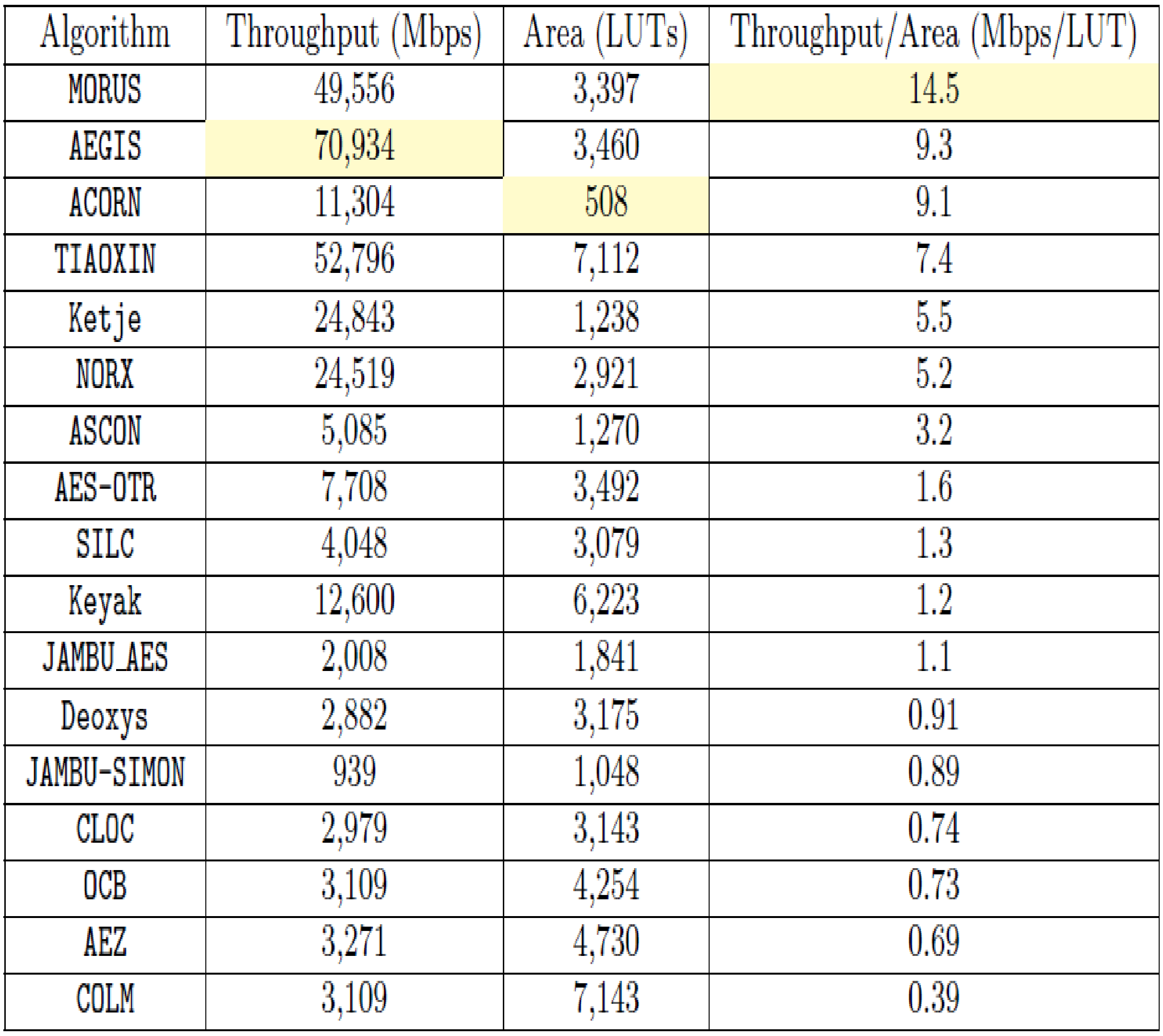
The Cryptographic Engineering Research Group (CERG) at George Mason University (GMU), USA, operates and maintains the online platform ATHENa aimed at fair, comprehensive, and automated evaluation of hardware cryptographic cores targeting Systems on Chip, FPGAs, and ASICs. Comparing of the FPGA implementations of the CAESAR competition candidates is one of their ongoing projects. They have also provided round-based high-speed implementations. The most recent benchmarking results are published in [12], where the authors provided a summary of available implementations for candidates that are designed by either the CERG research group or other members of the cryptographic community. The benchmarking process was performed only for FPGA and some of the designs were implemented using High-Level Synthesis (HLS) as opposed to manual Register-Transfer-Level (RTL) design [13]. Moreover, only the implementations that are compliant with the CAESAR Hardware API [15] were considered. The authors of [13] and [12] adopted a few assumptions that motivated their benchmarking process. These assumptions include:

- The rankings of different implementations will be the same regardless of whether the benchmarking is done using FPGA or ASIC.
- The rankings of different implementations will be the same regardless of whether the benchmarking is done using HLS or manual RTL.
- It is only fair to compare implementations with the same hardware API.



Input	Output
Associated data	*Associated data* (unchanged)
Plaintext	*Ciphertext*
	(= encrypted plaintext
	+ authentication tag)
Secret Key	
Public nonce	
Secret nonce	

Figure 8 the inputs and output of authenticated ciphers participating in the CAESAR competition [15]



Algorithm	Throughput (Mbps)	Area (LUTs)	Throughput/Area (Mbps/LUT)
MORUS	49,556	3,397	14.5
AEGIS	70,934	3,460	9.3
ACORN	11,304	508	9.1
TIAOXIN	52,796	7,112	7.4
Ketje	24,843	1,238	5.5
NORX	24,519	2,921	5.2
ASCON	5,085	1,270	3.2
AES-OTR	7,708	3,492	1.6
SILC	4,048	3,079	1.3
Keyak	12,600	6,223	1.2
JAMBU_AES	2,008	1,841	1.1
Deoxys	2,882	3,175	0.91
JAMBU-SIMON	939	1,048	0.89
CLOC	2,979	3,143	0.74
OCB	3,109	4,254	0.73
AEZ	3,271	4,730	0.69
COLM	3,109	7,143	0.39

Figure 9: FPGA Benchmarking Results of Round 3 Candidates [15]

Ciphers participating in the CAESAR competition will accept plaintext and variable length Associated Data (AD), and convert these into a cipher text with the help of a fixed-length public nonce, secret nonce and key (the use of a secret nonce is optional). Integrity is provided for the Associated Data, and both integrity and confidentiality are provided for the plaintext. AD is a part of a message that should be authenticated, but doesn't need confidentiality. For example the payload of a packet should by encrypted (this is the plaintext) and authenticated, but the header should be authenticated only. Integrity is important for both parts, so no attacker can fool the receiver in to thinking that he is communicating with someone else. The nonce is an extra number aside from the key. For example it prevents the same cipher text to be obtained when the same message is encrypted twice with the same key. Often it cannot be reused multiple times for the same key, without the loss of the cipher security.

The evaluation of different candidates is based on a common interface and protocol, it is desirable to have a common API. At the beginning of the CAESAR competition in 2012 SW API was specified, and in 2016 the official HW API – the CAESAR HW API for Authenticated Ciphers – was established. The first version of the CAESAR HW Development Package (v1.0) [17] was implemented in 2016 to support CAESAR Round 2 submissions. Although the API allows for LW implementations with external data bus widths of 8, 16, or 32 bits, Development Package v1.0 did not support bus widths of less than 32 bits. Though work-around are possible, this shortfall discouraged the submission of true LW CAESAR Round 3 candidates, which were due in summer 2017. Subsequently, the LW CAESAR HW Development Package (v2.0) was released in Dec. 2017 [18], and facilitates lower-area implementations by

1) Permitting external bus widths of 8, 16, or 32 bits,

2) Reducing the amount of functionality automatically provided by Pre- and Postprocessor.

The CAESAR HW API was introduced to provide a common interface for the hardware implementations of the ciphers participating in the competition. It makes the comparison of different algorithms easier and fairer. The Hardware API separates the external communication and the development of the Core (which is called Cipher Core), containing the cipher specific part. One of the useful features is the support for a wide range of data port widths (ranging from 8 to 256 bytes), which are functionally completely separated from the Cipher Core. Furthermore, it also supports an arbitrary length of the input stream. There is support for encryption and decryption with the same core. It can communicate with simple devices like FIFOs to use as memory and it is relative lightweight.

The AEAD core consists of three main blocks: pre-processor, cipher core, and post-processor, as shown in Figure.10 The main difference between the different algorithms is in the cipher core implementation, as it contains the hardware blocks that perform either encryption or decryption and authentication algorithm steps.
The George Mason University Application Programming Interface (GMU-API) blocks are described as follows:

1) Pre-processor
The first block of the AEAD core is the Pre-processor, it receives public and secret data and start processing them.

2) Post-processor
The post-processor is the output stage of the API.

3) Cipher core
The cipher core is divided into two blocks: core data path and core controller.
The core data path contains the hardware which is responsible for encryption or decryption and tag generation through processing the associative data, in addition to the hardware which is responsible for the key scheduling and the round keys generation.

The cipher core controller is an algorithmic state machine that generates control signals to the core data path based on the some information signals takes from the pre-processor.

4) Bypass first-in-first-out (FIFO)
Small FIFO which bypasses the tags, header, associated data and any data blocks that are used in the authentication Process and will not be encrypted.

5) Auxiliary FIFO
The memory used by the post-processor to temporarily store the decrypted message till the result of authentication is ready.

The area overhead introduced by the API is mainly determined by the block size used by the ciphers and the size of the input words.

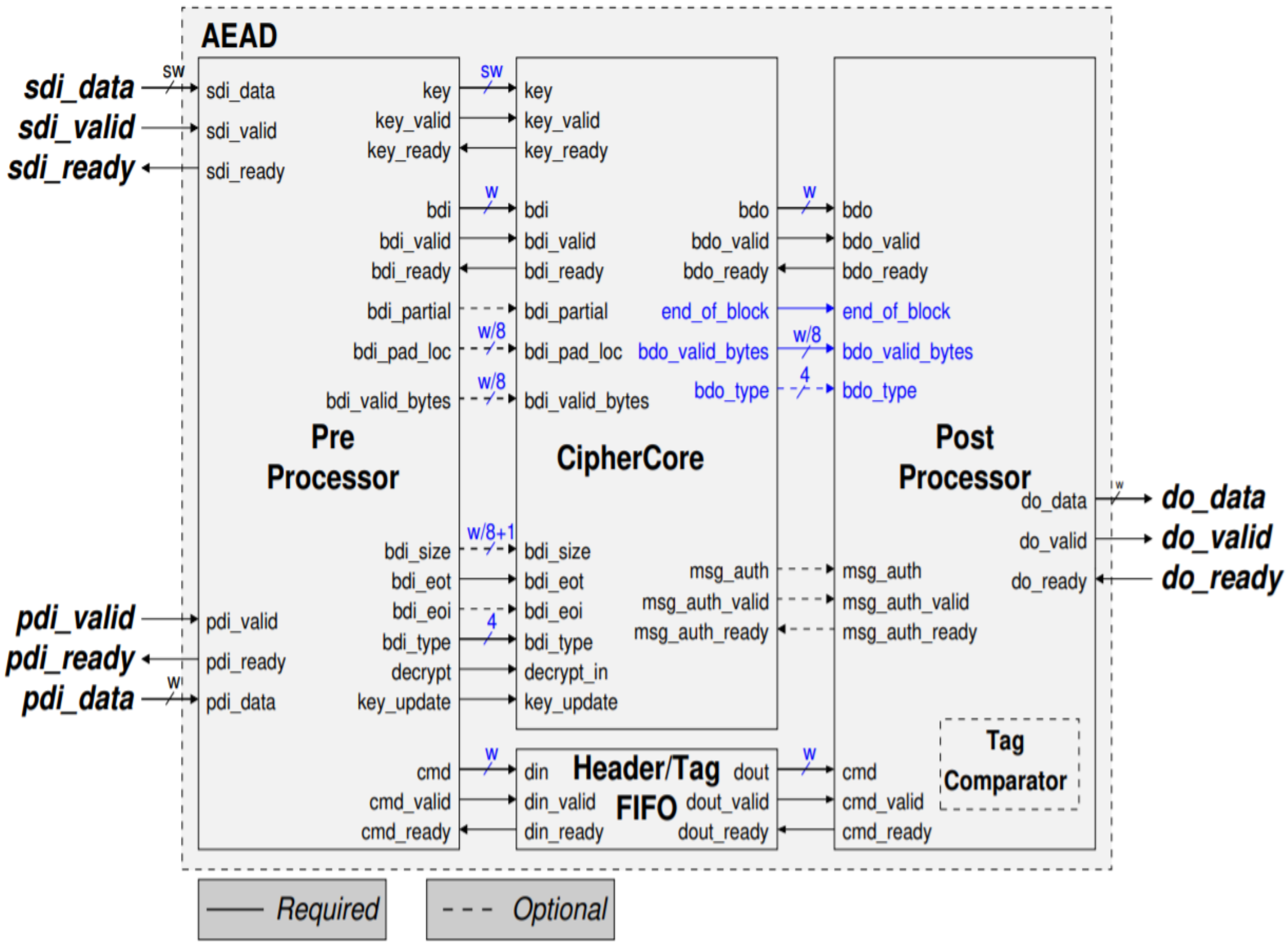


Figure 10: Top-level block diagram of a lightweight architecture of a single-pass authenticated cipher core, AEAD [16]

Chapter 3 : Low Area and Low Power Implementation of CAESAR Authenticated Ciphers

In this chapter we present the work done for low area low power optimization of five selected candidates namely NORX, SILC, Tiaoxin, COLM, and JAMBU, and compare the optimized results with the implementations available for high-speed implementations and those provided in other research. First the Related work is presented, then we summarize the operation of each algorithm of the selected candidates and then present the optimization techniques applied. Finally the results of optimized implementations are compared against the available high-speed implementations and other implementation provided in research.

3.1. Power Measurements

After viable implementation versions of the ciphers are found, low-level improvements can be performed on them. There have been some previous studies about these optimizations on FPGA and some recommendations for energy-efficient designs have been published. The power consumption of an FPGA consists of two main parts: dynamic power consumption and static power consumption. The static power is mainly due to the complicated wiring and gate leakage in FPGA. It is not something the designer can control, so it does not differ much between designs. The dynamic power consumption is due to the switching activity, and the change of status of the wires. This can be improved by good design.

Several general strategies can be used to reduce the dynamic power consumption. One is to make use of the embedded blocks as much as possible, as they are designed at the gate level instead of using less efficient LUTs. Besides, a second strategy is clock gating, to avoid switching activity in certain parts of the design when it is not used. In Smaller designs the last technique is not of much use. Glitches often cause power consumption. Glitches are unwanted switching activities that happen before a signal settles down to its correct value. Avoiding too long logic paths can reduce this, for example pipelining could be applied on such paths. Also rearranging the logic, can also help in some case.

The energy consumption can be calculated by using certain energy estimation tools or using a hardware setup to calculate the power .Two common power estimation tools used by Xilinx boards are the Xilinx Power Estimator (XPE) and Xilinx Power Analyzer (XPA). Both tools are simple and quick to use, as they are designed to work together with Vivado or ISE, which is used to synthesize and generate the bit stream of designs on Xilinx FPGA's. XPA is used to evaluate the design when the full HDL code is available while XPE is used before the full HDL code is written. The XPA is the one of interest for this thesis because the throughput per clock cycle of the implementations can be calculated from simulation, the power consumption could be used to deduce the energy consumption. XPA uses the actual design parameters like the board voltage, clock frequency and the load on the output pins, to estimate the power consumption.

3.2. Literature Survey

Hardware submissions of CAESAR Round 3 candidates were made available for public evaluation and FPGA benchmarking in July 2017 in the form of VHDL or Verilog code compatible with the CAESAR Hardware Applications Programming Interface (HW API).
However, the majority of these implementations were designed for high speed (HS), in that they employed simple either unrolled or iterative architectures, and used full-width data paths and wide I / O bus widths. Such design choices are not surprising, given that HW applications are historically evaluated based on best throughput-to-area (TP / A) ratios, which are accomplished using the aforementioned architectures.

Additionally, the majority of HW submissions were implemented using the CAESAR HW Development Package, discussed at [17]. At the time, the HS package was the only available version, but was not optimal for LW implementations, in that the minimum I/O bus width was 32 bits, and I/O modules often contained resource intensive units (e.g., a universal padding unit) not necessary for certain designs. As a result, the true LW potential of candidates stating a LW use case, as intended by the CAESAR committee, was not evaluated. Additionally, third-party evaluations of these implementations in resource-constrained environments (e.g., low-cost FPGAs with minimum area budgets) are more difficult.

Certain CAESAR candidates can be realized using low area implementations. An example in [19] which present low area implementation of Ascon, it uses 2.57 Kilo-Gate Equivalent (KGE) and requires as little as 15µW for a 1MHz clock source in 90 nm ASIC technology. The Architecture of the data path uses a radical low-area approach, which can be described as “one bit operation per cycle”. This version is not compatible with the CAESAR Hardware Application Programming Interface (HW API) and it results in 512 clock cycles per round transformation.

In [29], a very compact AEGIS design is introduced. They used the Canright’s implementation [30] of SBOX to process 8 bits of state at a time. They implemented an optimization to reduce the number of 128-bit registers necessary to store the next state. The proposed design has a low area which requires 18 KGE.

There were attempts to provide dedicated lightweight authenticated encryption schemes. An example Hummingbird-2 [2] which is an authenticating encryption primitive that has been developed specifically for resource-constrained devices such as RFID tags, wireless sensors, smart meters and industrial controllers. Hummingbird-2 can be implemented with very limited hardware or software footprint and is therefore ideal for providing security in low-cost ubiquitous devices. Hummingbird-2, needs 2.2 kGE in ASIC.

In [21], authors proposed a new Authenticated Lightweight Encryption algorithm coined ALE. The AES round transformation and the AES-128 key schedule are the basic operation of ALE. ALE is an online single-pass authenticated encryption algorithm that supports optional associated data. Its security relies on using nonce. An optimized low-area implementation of ALE in ASIC hardware was provided which is about 2.5 kGE. This area is almost two times smaller than that of the lightweight implementations for AES-OCB and ASC-1 using the same lightweight AES engine. At the same time, its performance is at least 2.5 times higher than the alternatives in their smallest implementations as it requires only 4 AES rounds to both encrypt and authenticate a 128-bit data block.

The majority of HW submissions of CAESAR are implemented using the CAESAR HW Development Package v1.0 [17] then a new version of the CAESAR HW Development Package v2.0 supporting lightweight (LW) implementations [18] was released. In [22] authors present LW implementations of CAESAR candidates Ketje Sr, Ascon-128, and Ascon-128a. They demonstrate that the use of a prototype version of the LW Development Package v2.0 significantly reduces the overhead of interface modules compared to the previous CAESAR HW Development Package v1.0.

In [23] authors improved upon the HS implementations of ACORN, NORX, CLOC, and SILC ciphers by designing true LW implementations. Their design methodology consists of two aspects:

- Use of the LW CAESAR HW Development Package v2.0, with I/O bus widths of 8, 16, or 32 bits.
- Use of internal data paths for cryptographic primitives and authenticated cipher layer operations, which are matched to their corresponding I/O bus widths.

3.3. Low Power low Area Optimization

The optimization methodology depends on resource sharing as the addressed Ciphers (NORX, Tiaoxin, SILC, COLM, and JAMBU) are found to use resource duplication in their implementations. The CAESAR HW Development Package v1.0 is used in the proposed work.

3.3.1. The General Approach

Very high speed integrated circuit Hardware Description Language (VHDL) is the used hardware description language (HDL) to implement algorithms in register transfer level (RTL). The operating frequency is chosen to be 100 MHz for all algorithms. The implementation is done for the Cipher Core only, the postprocessor, preprocessor, and FIFOs are excluded from implementation as the optimization is done for Cipher Core only.

3.3.2. Implementations Evaluation

To evaluate the hardware performance of the proposed optimized implementations, pairs of corresponding publicly available HS implementations (donated by High-Speed Implementations) and proposed Optimized implementations (denoted by Optimized Implementations) are benchmarked for FPGA and ASIC implementations

3.3.2.1. Implementation on field programmable gate array (FPGA)

FPGAs take advantage of the size and power efficiency of ASICs, while still providing flexibility in the form of reprogrammability. Making them cheaper and faster to develop on, but at the cost of worse specifications and cost-per-unit when made in large batches. Because of the exponential growth of transistor density through Moore's law, FPGAs have become feasible for more and more applications since their invention in 1982.

The FPGA implementation of the candidates is performed using the Xilinx Vivado 2016.2 design suite. The algorithms are synthesized using the Virtex-7 FPGA device. Vivado tool is used to perform the logic synthesis, mapping, placing, and routing. Vivado results report the area and power consumption of the algorithms.
For power consumption the Inputs/Outputs (IOs) power is excluded from the total dynamic power as in real case the Cipher Core IOs will be connected to internal signals not primary IOs of the FPGA.

3.3.2.2. Implementation on application specific integrated circuit (ASIC)

Synthesis step is done using Synopsys Design Compiler (DC) B-2008.09 for Linux. CMOS UMC 130nm technology is the used technology for synthesis and place and route steps. DC takes RTL codes, technology libraries, and constraints file as an input and produced the gate level netlist as an output. The switching activity file generated from Modelsim is included for accurate power consumption results.

3.3.3. The GMU Hardware API for Authenticated Ciphers

In the implementations made in this thesis, the Cipher Core is implemented for each cipher and fit in to this API. The size of the input words and the block size used by the ciphers determine the area overhead introduced by the API. Since the five chosen ciphers have small block sizes (maximum 128-bit), and small word widths are used in the inputs (32-bit), this overhead in absolute terms is not that high. However, since the ciphers are all fairly small in area, the overhead in relative terms it is not negligible.

When the ciphers will be used they will be integrated in a bigger design the majority of time, and not use this specific API. However the API forces the Cipher Core of all ciphers to be structured a certain way, which makes the comparison of them more accurate. It is also useful to verify the correct functionality of the implementations with the API, and the structure makes it easier to understand and possibly reuse the implementations with different APIs in the future.

3.3.4. Common Features for CASEAR Candidates

The following selected algorithms: SILC, Tiaoxin, COLM, and JAMBU-AES are based on Advanced Encryption Standard (AES) to perform the encryption and the decryption processes. AES is a symmetric block cipher that uses several key sizes. AES has various standard versions: AES-128, AES-192, and AES-25611.

The number of rounds for each version depends on the key size. It uses 10, 12, and 14 rounds for a key size of 128, 192, and 256 respectively. Figure. 3 shows a flowchart for the AES encryption algorithm. AES operates on a 4 4 column-major order array of bytes, termed the state and applies four permutation functions in each round which are

• Substitute bytes:
Uses an S-box to perform a byte-by-byte substitution of the block.

• ShiftRows:
A simple permutation that rotates the state rows right with a different number of positions.

• MixColumns:
A substitution that combines the four bytes of each column of the state using an invertible linear transformation.

• AddRoundKey:
A simple bitwise XOR of the current block with a portion of the round key.

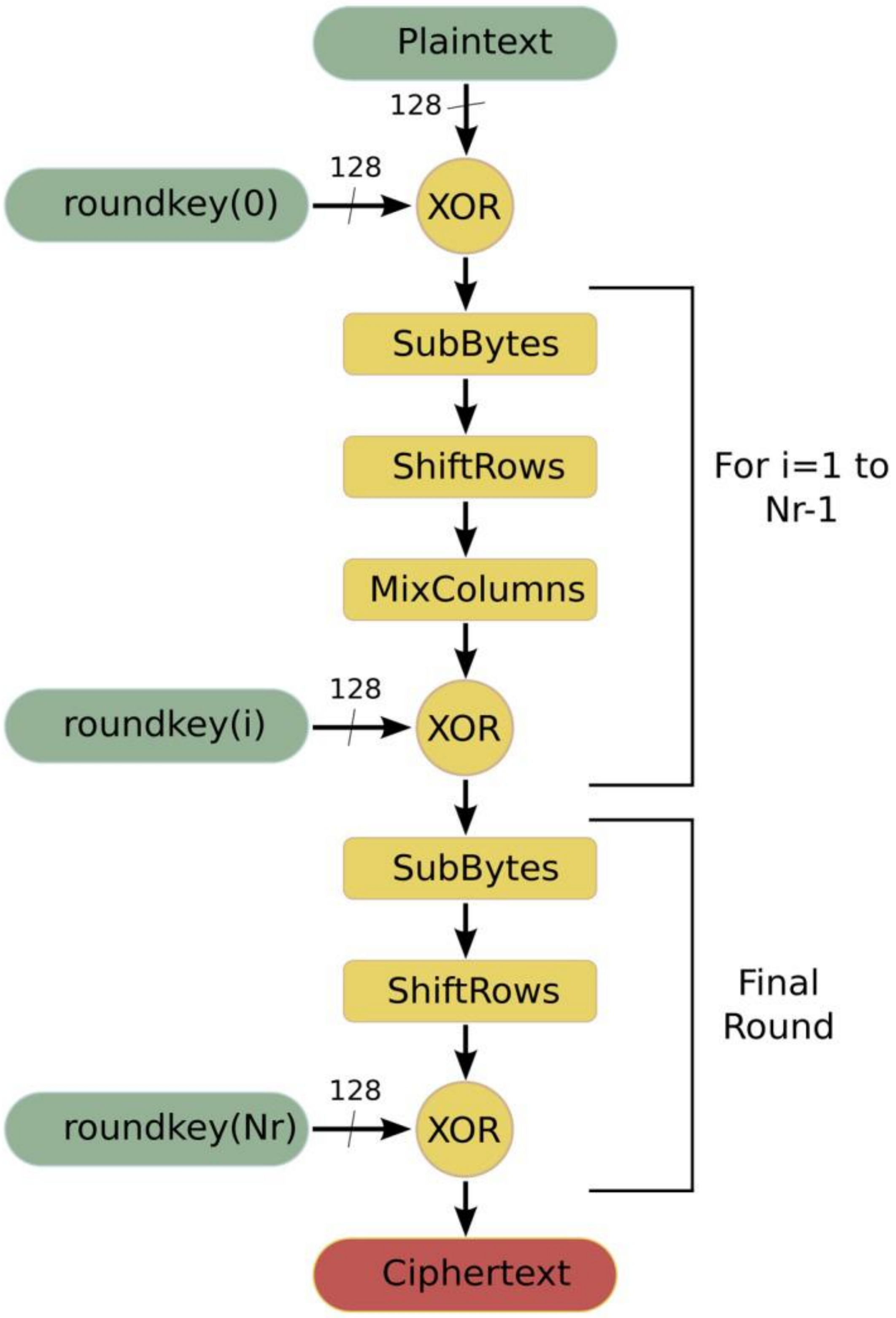


Figure 11 AES encryption algorithm

3.4. NORX

NORX [24] has a unique parallel architecture based on monkey duplex construction, where the degree of parallelism and tag size can be changed arbitrarily. The scheme is based on Addition-Rotation-XOR (ARX).

NORX was optimized for hardware and software efficiency, with a SIMD-friendly core, almost byte-aligned rotations, only bitwise operations and no secret-dependent memory lookups. The NORX core is inspired by the ARX primitive, however integer addition is replaced with the approximation $a \oplus b \oplus (a \wedge b) \ll 1$. This improves hardware efficiency and simplifies cryptanalysis. Furthermore, NORX specifies a dedicated datagram to avoid users the trouble of defining custom signaling and encoding, and to facilitate interoperability.

3.4.1. Operation

NORX has High security as it supports 128-bit and 256-bit keys and authentication tags of arbitrary size, thanks to its duplex construction. The core permutation of NORX was designed and evaluated to be cryptographically strong. NORX uses the monkey Duplex construction enhanced with the capability to process payload in parallel.

The duplex construction and sponge function are being used widely to implement many algorithms for cryptography including AE schemes. Some of those AE algorithms are submitted to CAESAR such as ASCON, NORX, and Ketje. The fixed permutation F in the duplex construction function is determined by the following two parameters: capacity (c) and bitrate (r). The state size (S) are computed by adding both parameters. There is a trade-off for a fixed state size between speed and security because of assigning different values for the capacity and bit-rate. For example, to make the algorithm faster higher bitrate is needed which makes the algorithm has lower security and vice versa.

The bitrate part of the permutation F get the input blocks (plaintext), the input blocks gets padded to achieve the full r bits if it is smaller than r bits. The r-bit output blocks (ciphertext) are squeezed out after being processed by the permutation F. The duplex construction is used to develop the Authentication encryption with associated data as described in Figure 12. First, the concatenation of a nonce and a secret key K are given into the initial state which is applied to the permutation F. The following steps show the process of duplex construction on n blocks of plain text P_i.

For the plain text P_i, the plaintext with the bitrate part are XOR-ed to compute C_i. The permutation F is kept calling until the last block. Similarly, the public associated data blocks could be processed by absorbing them without encryption. The r-bit authentication tag T is get by completing the encryption. The security sponge-based authenticated encryption schemes largely depends on the internal structure of permutation F which is implemented usually as a sequence of elementary operations named rounds. As the number of rounds in the permutation F increase, the cryptanalysis is more complex and the relevant AEAD becomes more secure but this will require more hardware resources.

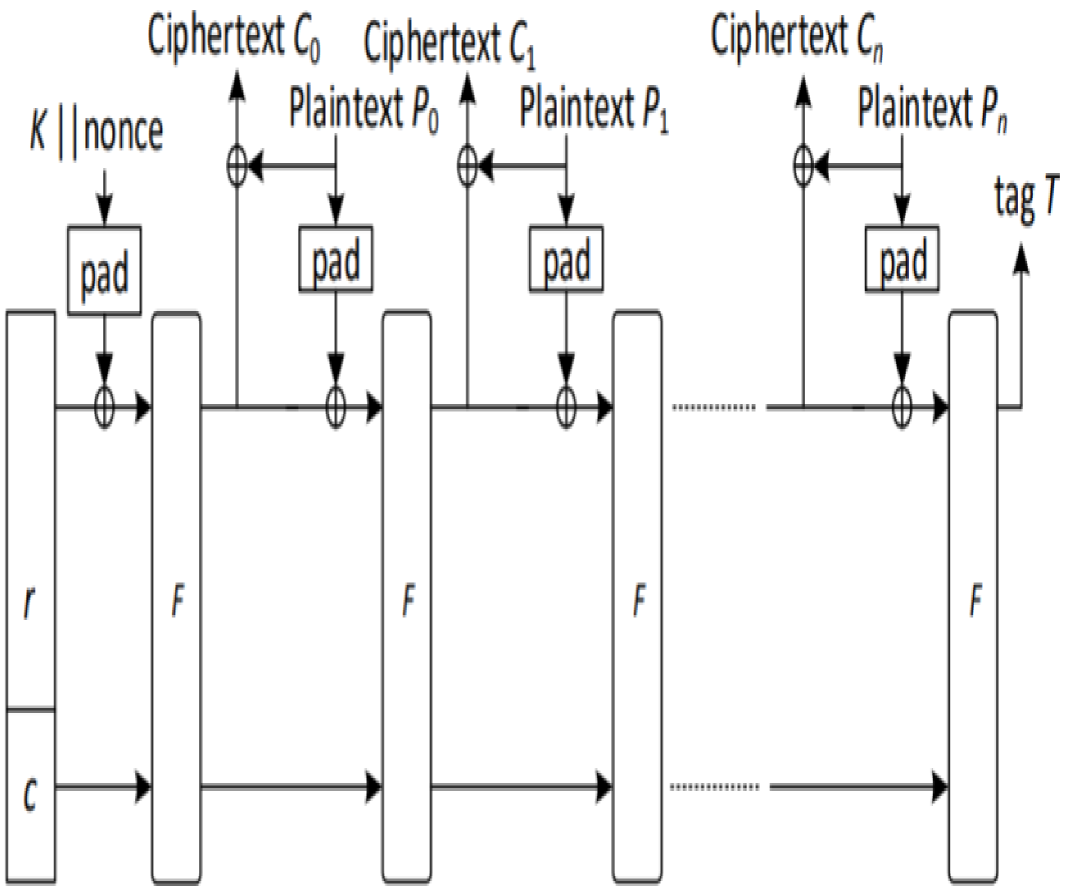
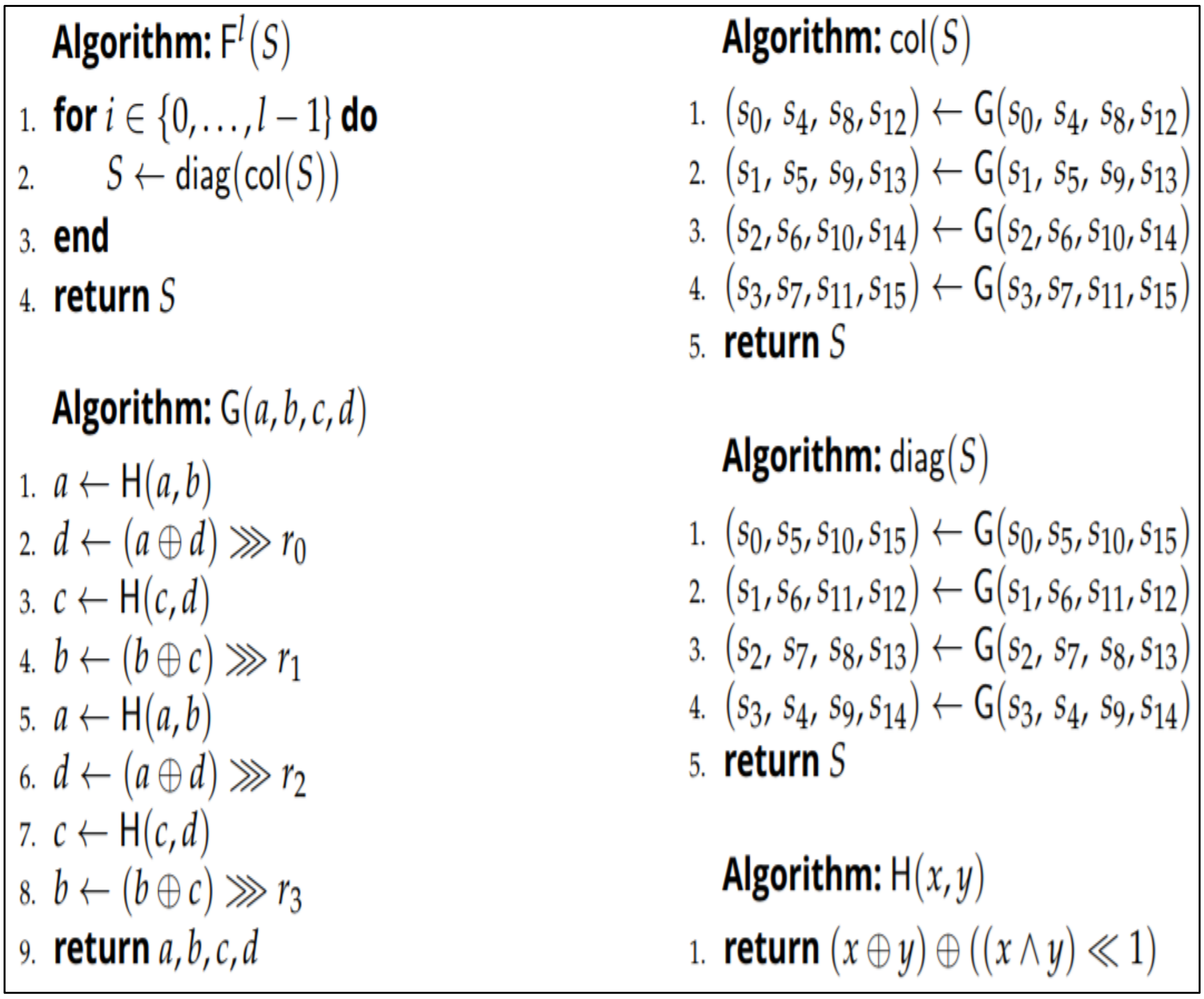


Figure 12 Duplex construction based Authenticated encryption scheme [24]

The core algorithm F of NORX is a permutation of $S = r + c$ bits. F is called a round and F' denotes its $l -$ fold iteration. The state is viewed as a concatenation of 16 words, i.e. $S = s0||$ ... $||s15$, out of which $s0||$ ... $||s11$ are called the rate words where data blocks are injected and $s12||$ ... $||s15$ are called the capacity words which remain untouched. Conceptually, the 16 state words are arranged in a 4x4 matrix.

The pseudo code for the NORX core permutation F' is given in Figure 13. A single NORX round F processes the state S by first transforming its columns with the function G using function Col(S), and then transforming its diagonals using function Diag(S). The G function uses cyclic rotations and non-linear operation interchangeably to update its four input words.



```
Algorithm: $\mathsf{F}^l(S)$
1. for $i \in \{0,\ldots,l-1\}$ do
2.     $S \leftarrow \mathsf{diag}(\mathsf{col}(S))$
3. end
4. return $S$

Algorithm: $\mathsf{G}(a,b,c,d)$
1. $a \leftarrow \mathsf{H}(a,b)$
2. $d \leftarrow (a \oplus d) \ggg r_0$
3. $c \leftarrow \mathsf{H}(c,d)$
4. $b \leftarrow (b \oplus c) \ggg r_1$
5. $a \leftarrow \mathsf{H}(a,b)$
6. $d \leftarrow (a \oplus d) \ggg r_2$
7. $c \leftarrow \mathsf{H}(c,d)$
8. $b \leftarrow (b \oplus c) \ggg r_3$
9. return $a,b,c,d$

Algorithm: $\mathsf{col}(S)$
1. $(s_0, s_4, s_8, s_{12}) \leftarrow \mathsf{G}(s_0, s_4, s_8, s_{12})$
2. $(s_1, s_5, s_9, s_{13}) \leftarrow \mathsf{G}(s_1, s_5, s_9, s_{13})$
3. $(s_2, s_6, s_{10}, s_{14}) \leftarrow \mathsf{G}(s_2, s_6, s_{10}, s_{14})$
4. $(s_3, s_7, s_{11}, s_{15}) \leftarrow \mathsf{G}(s_3, s_7, s_{11}, s_{15})$
5. return $S$

Algorithm: $\mathsf{diag}(S)$
1. $(s_0, s_5, s_{10}, s_{15}) \leftarrow \mathsf{G}(s_0, s_5, s_{10}, s_{15})$
2. $(s_1, s_6, s_{11}, s_{12}) \leftarrow \mathsf{G}(s_1, s_6, s_{11}, s_{12})$
3. $(s_2, s_7, s_8, s_{13}) \leftarrow \mathsf{G}(s_2, s_7, s_8, s_{13})$
4. $(s_3, s_4, s_9, s_{14}) \leftarrow \mathsf{G}(s_3, s_4, s_9, s_{14})$
5. return $S$

Algorithm: $\mathsf{H}(x,y)$
1. return $(x \oplus y) \oplus ((x \wedge y) \ll 1)$
```

Figure 13 NORX Algorithm [24]

3.4.2. Low Area Low Power Optimization

The high-speed NORX hardware implementation duplicates the G function 8 times. The round operation is done in 2 steps, at the first step, 4 G functions operate on the columns, and at the second step, and the other 4 G functions operate on the diagonals so the same process is done on columns and diagonals sequentially.

In order to optimize NORX for low area, only one G function is used so that the Round operation is processed in 8 cycles instead of 1 cycle. In this research, a mux is added which select the input to G function and the state register is used to store the output of the G function, so no more sequential elements are added to reduce the switching power.

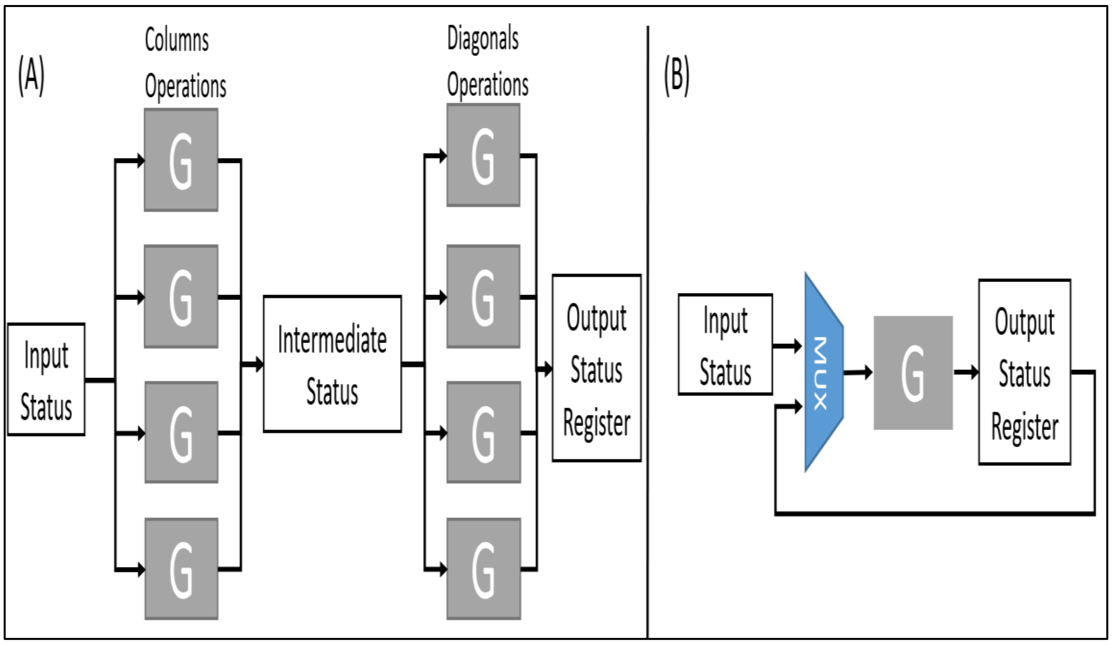


Figure 14: NORX High speed Block diagram

A small FSM is added to control the flow of data, and in order to account for the additional delay due to the insertion of the pipe stage, few counters are added. The optimization removes 7 instances of the G function by converting the implementation to be sequential

3.4.3. Results

1) FPGA Results

The results show that the proposed optimized implementations achieve an area reduction for NORX with 31%, and a Dynamic Power consumption reduction by 70% respectively. As a cost, throughput (TP) decreases for NORX by 87.5%, and throughput-to-area (TP/A) decreases by 82%.The reduction in TP and TP/A ratio is expected as latency and throughput are sacrificed for area reduction.

The proposed optimized implementations are compared to work proposed in [16]. In [16] virtex-6 FPGA is used for implementation, while virtex-7 FPGA is used in this research so a comparison is done between Area reduction, Dynamic Power reduction and throughput-to-area change achieved by proposed work and the work in [16]. For NORX proposed implementation has lower area reduction with 31% compared to 53.3% in [16] and the throughput-to-area (TP/A) has decreased with 82% while it increased with 25.5 % in [16].

2) ASIC Results

The results show that the proposed optimized implementations achieve an area reduction for NORX with 44%, and a Dynamic Power consumption reduction by 54% respectively. As a cost, throughput (TP) decreases for NORX by 87.5%, and throughput-to-area (TP/A) decreases by 77%.

3.5. SILC

SILC [27] (Simple Lightweight CFB (Cipher Feedback)) is a mode of operation with a block cipher as the underlying base function. It is a lightweight function that is suitable for use in constrained hardware devices as the hardware implementation cost is very low. SILC doesn't need much precomputation other than key scheduling so less hardware is needed there by reducing computational cost.

SILC uses AES-128 block cipher which improves memory utilization and latency more than LED and PRESENT block ciphers [31]. The S-box in AES allows pseudo-randomness which provides provable security against birthday attack (i.e., cryptographic attack that applies the birthday paradox mathematics) [27].

3.5.1. Operation

The encryption and decryption operations of SILC can be done with the use of the encryption function alone. Both encryption and decryption are online processes that means i^{th} input block M_i depends only on the blocks $M_i. . . , M_{i-1}$. It is inverse free which means it only requires encryption for both encryption and decryption processes. For verification of the tag it use EtM composition scheme which means the tag is verified before decryption. It is a two pass scheme i.e., first the authentication is executed and then the encryption.

The round keys for key scheduling is the only pre-computation needed in SILC. For this reason, no extra hardware register is needed for storing the pre-computed result. SILC use a GF multiplier only in AES encryption function which reduces the area as GF multiplier requires huge number of gates [32]. The S-box block dominates the SILC power consumption due to its large size (16 x 16 x 8) in look-up table implementation.

SILC uses four subroutines to perform the encryption and decryption operations:

1) HASH.
2) Encryption (ENC)
3) Pseudo Random Function (PRF)
4) Decryption(DEC)

1. HASH Function

The HASH function takes the input as key (K), associated data (A), nonce (N) concatenated with parameter and returns the output which is intermediate tag (V). In the case of an empty associated data string it encrypts the zero prepended nonce. In the case of a non-empty associated data the XOR-ed associated data value is encrypted with the previous encrypted associated data value. The encryption function output is then XOR-ed with the associated data length and then sent to the tweak function (g), and finally the intermediate tag (V) is returned.

2. Encryption (ENC)

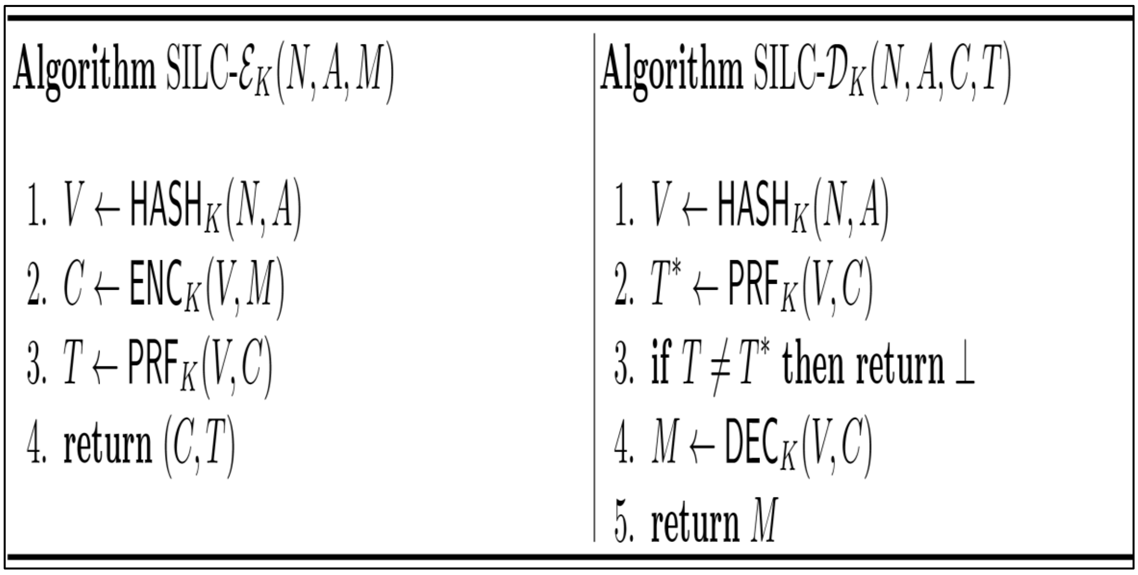
The ENC function takes the input as the message (M) and the intermediate tag (V) and returns output which is ciphertext(C). If the message length is zero then the cipher text is a string of zeros. In the case of the message is not empty, the cipher text first block is obtained by XOR-ing the message first block with encrypted output of the intermediate tag (V). The i^{th} cipher text block is generated by XOR-ing the i^{th}message block with encrypted value of the $(i-1)^{th}$ ciphertext.

3. Pseudo Random Function (PRF)

The PRF function takes the input as the cipher text (C) and the intermediate tag (V) and returns the output which is tag (T). The tweaked (V) is encrypted first and then XOR-ed with the cipher text first block. The next encryption block receive the output as feedback. The most significant 64 bits out of the tweak of last block's encryption is taken to generate the tag.

4. Decryption(DEC)

The DEC function takes the input the cipher text (C) and the intermediate tag (V) as the inputs and returns the message (M) as output. If the cipher text length is zero then the message output is a string of zeroes. In the case of the cipher test is not empty, the message first block is obtained by XOR-ing the cipher text first block with encrypted output of the intermediate tag (V). The i^{th} message block is generated by XOR-ing the i^{th} cipher text block with encrypted value of $(i-1)^{th}$ cipher.



```
Algorithm SILC-$\mathcal{E}_K(N, A, M)$

1. $V \leftarrow \mathsf{HASH}_K(N, A)$
2. $C \leftarrow \mathsf{ENC}_K(V, M)$
3. $T \leftarrow \mathsf{PRF}_K(V, C)$
4. return $(C, T)$
```

```
Algorithm SILC-$\mathcal{D}_K(N, A, C, T)$

1. $V \leftarrow \mathsf{HASH}_K(N, A)$
2. $T^* \leftarrow \mathsf{PRF}_K(V, C)$
3. if $T \neq T^*$ then return $\perp$
4. $M \leftarrow \mathsf{DEC}_K(V, C)$
5. return $M$
```

Figure 15: SILC Algorithm [27]

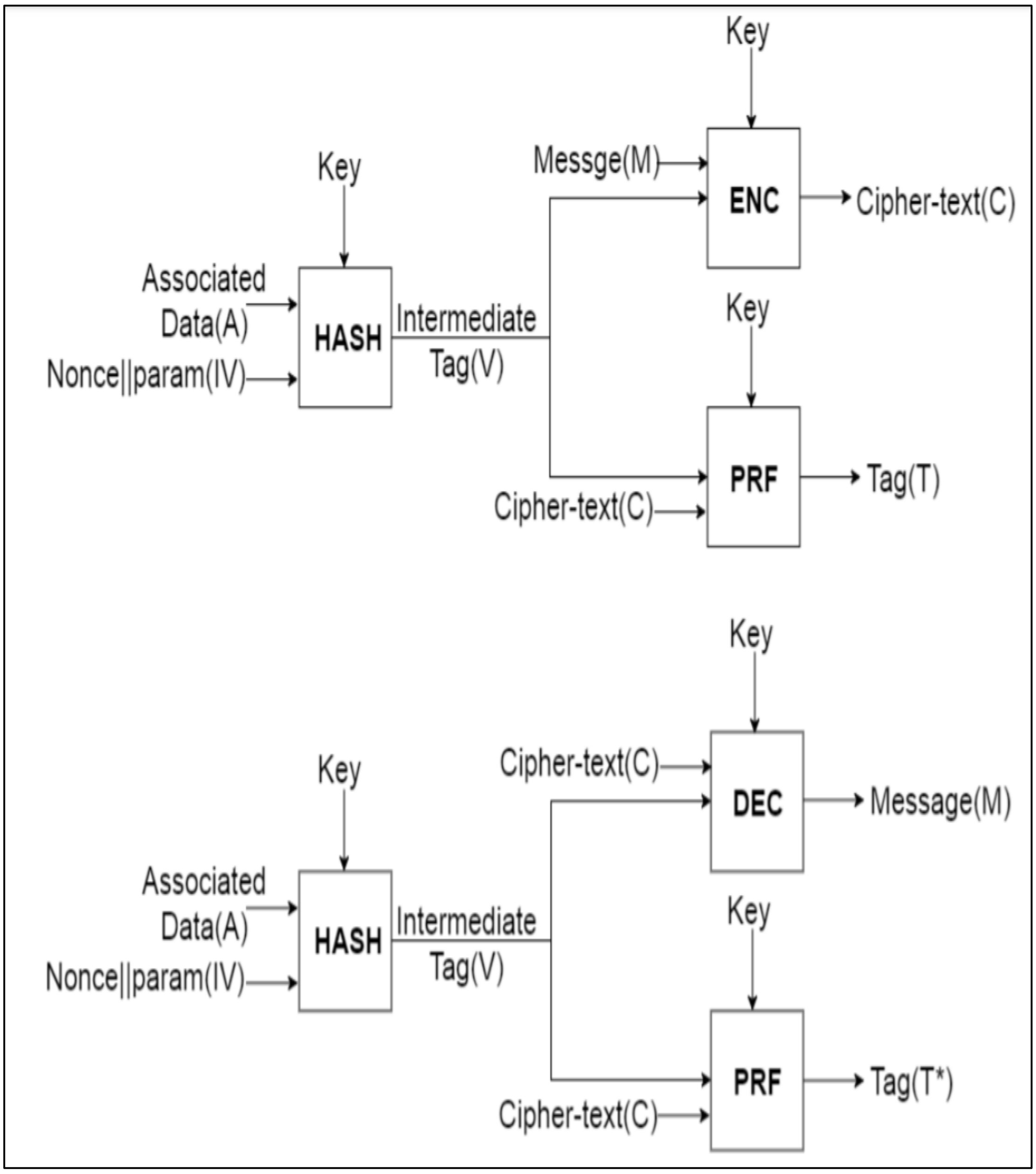


Fig. 10. SILC encryption and decryption block diagrams [27]

3.5.2. Low Area Low Power Optimization

The four functions used in SILC HASH, ENC, DEC, and HASH, are all sequential. The high-speed implementation using two regular 128-bit AES cores to achieve the highest possible throughput. However, at the expense of large area. Since the high speed implementation was intended for maximum possible throughput. It used two AES cores exploiting the parallelism of ENC and PRF subroutines, which in turn increased the area massively.

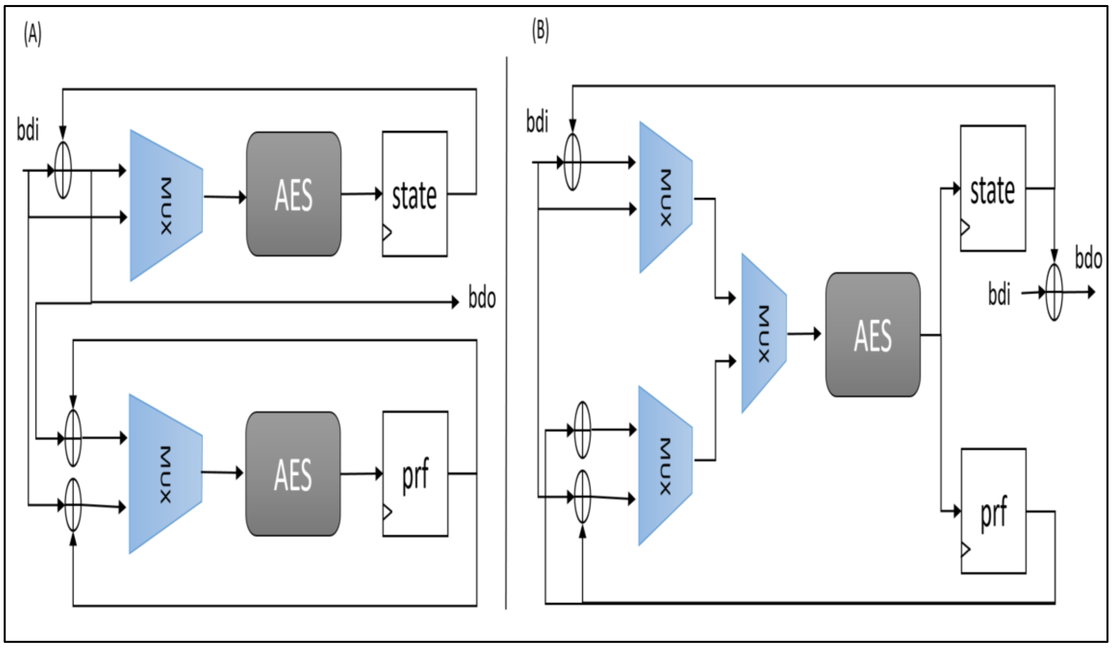


Figure 16: SILC high speed block diagram

The idea of making the design efficient is to use only one instance of AES and reuse it in a multi-cycle approach. In order to optimize SILC for low area, the block cipher calls in ENC and PRF are done sequentially. Each function of ENC and PRF use One AES core, so calling the ENC and PRF sequentially will require only one AES core to perform both functions. As a result, one round operation is done in 2 cycles instead of 1 cycle.

3.5.3. Results

1) FPGA Results

The results show that the proposed optimized implementations achieve an area reduction for SILC with 33%, and a Dynamic Power consumption reduction by 39% respectively. As a cost, throughput (TP) decreases for SILC by 50%, and throughput-to-area (TP/A) decreases by 25%.

The proposed optimized implementations are compared to work proposed in [16]. For SILC proposed implementation has lower area reduction with 33% compared to 69% in [16] and the throughput-to-area (TP/A) has decreased with 25% while it increased with 65 % in [16].

2) ASIC Results

The results show that the proposed optimized implementations achieve an area reduction for SILC with 26%, and a Dynamic Power consumption reduction by 39% respectively. As a cost, throughput (TP) decreases for SILC by 50%, and throughput-to-area (TP/A) decreases by 33%.

3.6. Tiaoxin-346

Tiaoxin-346 [25] is a nonce-based authenticated encryption scheme. It is the first to use only 3 AES round calls per 16-byte message (6 per 32-bytemessage). All 6 calls are fully parallelizable. It achieves 0.28 cycles per byte on Intel Haswell. Twice faster than AES-128 in counter mode, 3.5 to 6.5 times faster than AES-GCM.
It is analyzed against different attacks types. The design decisions (choice of state sizes, output function, etc.) were taken in order to make the cipher secure. Tiaoxin provides full security for nonce-respecting adversaries. Security claims include related-key attacks and distinguishers.

The internal state consists of 13 words each of 16 bytes. The 13 words are divided into three groups of 3, 4 and 6 words each (this is also the why it is named Tiaoxin-346). The function of state update for Tiaoxin-346 absorbs a 32 bytes message block and a new internal state is produced, as illustrated in Figure 17.

3.6.1. Operation

Tiaoxin-346 has three states T3, T4, T6 composed of 3, 4, 6 words, respectively. The Update operation (round function) is computes the new value of the states (in the different phases). As inputs, beside the three states, Update takes three additional words M0, M1, M2. I.e. Update: T3 × T4 × T5 × M0 × M1 × M2 → T3 × T4 × T6.

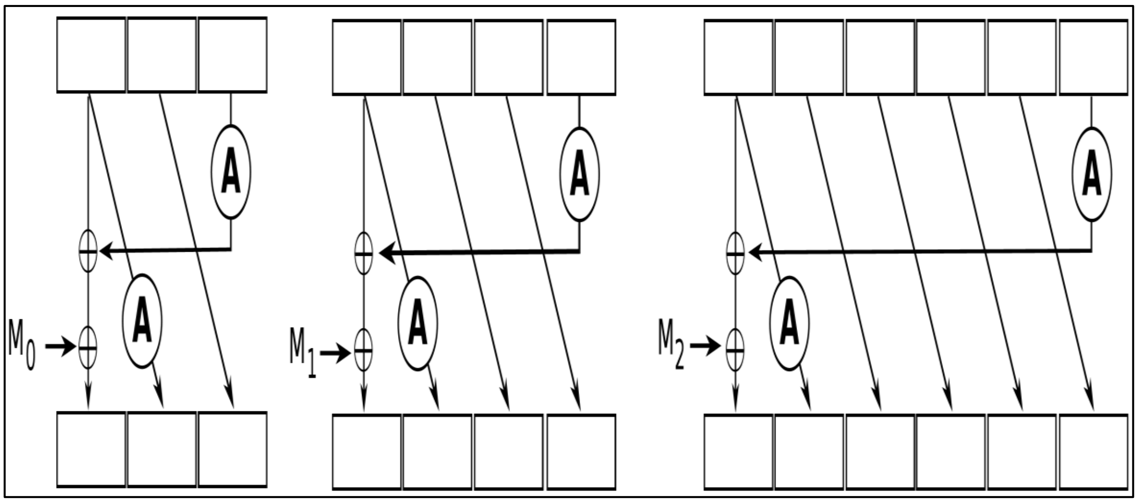


Figure 17: Tiaoxin-346 Algorithm [25]

Tiaoxin-346 works in four phases:
Initialization, Processing associated data, Encryption, and Finalization which are executed respectively

1. Initialization.

In the initialization, the key (K) and the public message number nonce (IV) are loaded into the three states T3, T4, T6 which are applied to 15 rounds

2. Processing associated data.

Assume the padded associated data has d blocks $AD = AD_1, \ldots . AD_d$. Each block is composed of two words. The Processing associated data is defined as:

```
$for\ i = 1\ to\ d$
        $Update(T_3, T_4, T_6, AD_i^0, AD_i^1, AD_i^0 \oplus AD_i^1);$
$end\ for$
```

3. Encryption.

Assume the padded message has m blocks: $M_1, \ldots, M_l$ Recall that each block is composed of two words, i.e. $M_i = M_i^0 || M_i^1$. In the encryption, a block M_i is processed in one round, and a block of ciphertext $C_i = C_i^0 || C_i^1$ is output. The encryption is defined as

```
$for\ i = 1\ to\ m$
        $Update(T_3, T_4, T_6, M_i^0, M_i^1, M_i^0 \oplus AM_i^1);$
        $C_i^0 = (T_3[0] \oplus T_3[2] \oplus T_4[1] \oplus (T_6[3] \,\&\, T_4[3]))$
        $C_i^1 = (T_6[0] \oplus T_4[2] \oplus T_3[1] \oplus (T_6[5] \,\&\, T_3[2]))$
$end\ for$
```

4. Finalization/Tag production.

After all message blocks have been processed, the words holding the lengths of the associated data and message are processed, then the states go through 20 more rounds, and the tag is produced as an XOR of all words of all states. This final phase is defined as:

```
$Update(T_3, T_4, T_6, AD_{lenght}, M_{length}, AD_{lenght} \oplus M_{length})$
$for\ i = 1\ to\ 20$
        $Update(T_3, T_4, T_6, Z_1, Z_0, Z_1);$
$end\ for$
$Tag = T_3[0] \oplus T_3[1] \oplus T_3[2] \oplus T_4[0] \oplus T_4[1] \oplus T_4[2] \oplus T_4[3]$
        $\oplus T_6[0] \oplus T_6[1] \oplus T_6[2] \oplus T_6[3] \oplus T_6[4] \oplus T_6[5]$
```

3.6.2. Low Area Low Power Optimization

The high-speed Tiaoxin-346 hardware implementation duplicates AES 6 times. In order to optimize Tiaoxin-346 for low area, only one AES is used. The round operation is processed in 6 cycles instead of 1 cycle. A small FSM is added to control the flow of data, Multiplexes are added to control data to the AES, and counters are added to account for the additional delay in the round operation. The block diagram in Figure 18 shows the optimized implementation, as it shown only one AES round is used.

The state updated takes 6 clock cycles. A MUX controls the input to AES and select one of $T_3[2]$, $T_3[0]$, $T_4[3]$, $T_4[0]$, $T_6[5]$ then $T_6[0]$ to computes $T_3[0]$, $T_3[1]$, $T_4[0]$, $T_4[1]$, $T_6[0]$ then $T_6[1]$ respectively. The optimization removes 5 instances of AES.

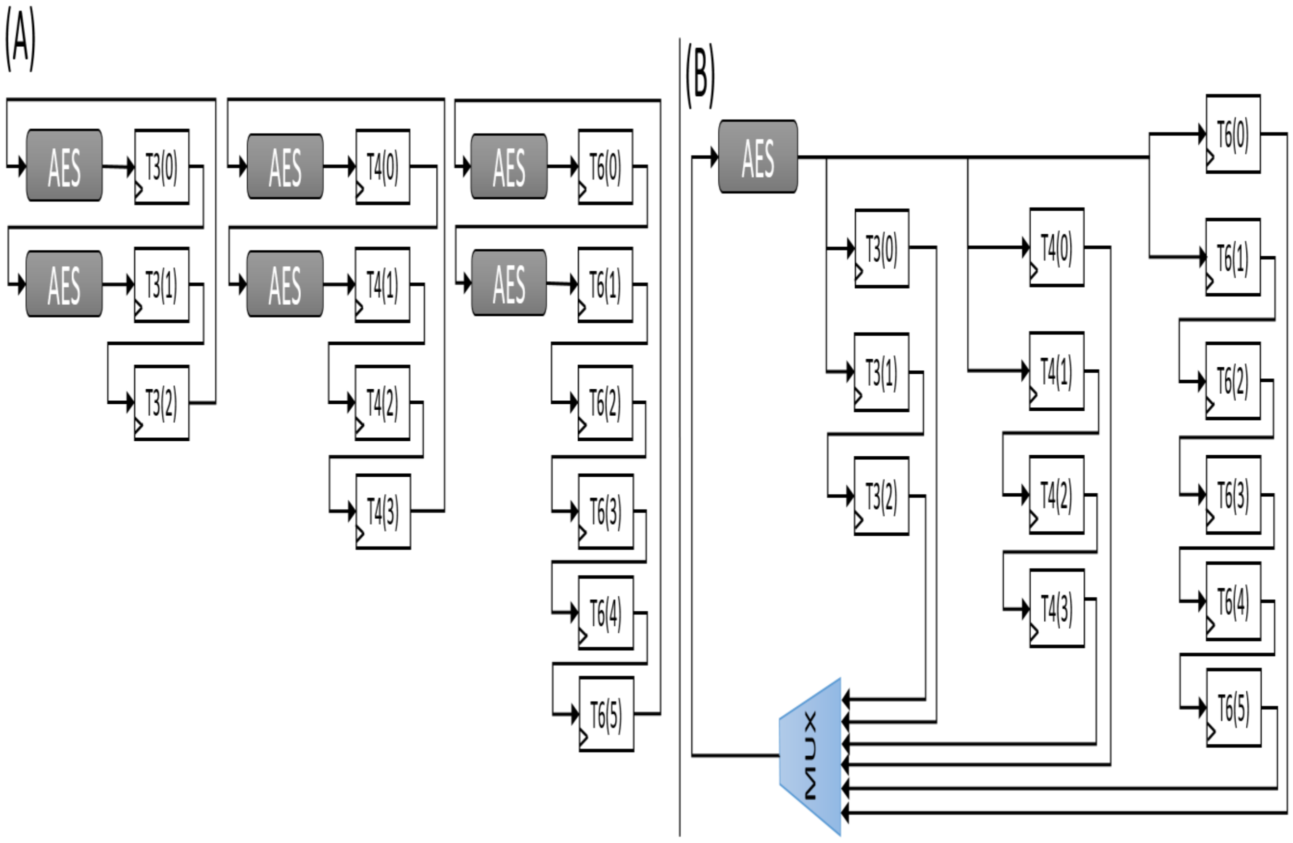


Figure 18: Tiaoxin-346 high speed state update block diagram

3.6.3. Results

1) FPGA Results

The results show that the proposed optimized implementations achieve an area reduction for Tiaoxin-346 with 38%, and a Dynamic Power consumption reduction by 65% respectively. As a cost, throughput (TP) decreases for Tiaoxin-346 by 83%, and throughput-to-area (TP/A) decreases by 73%.

2) ASIC Results

The results show that the proposed optimized implementations achieve an area reduction for Tiaoxin-346 with 43%, and a Dynamic Power consumption reduction by 46% respectively. As a cost, throughput (TP) decreases for Tiaoxin-346 by 83%, and throughput-to-area (TP/A) decreases by 70%.

3.7. COLM

COLM [26] is a block cipher which is based on Encrypt-Linear mix-Encrypt mode. The COLM was designed to achieve online misuse resistance, to be secure against block wise adaptive attacks and to be fully parallelizable. COLM cipher is formulated as a mixture of characteristics inherited from COPA and ELmD. COLM consists of two layers of encryption that are parallelizable and connected by a linear mixing function. While COPA uses plain XOR mixing, ELmE, ELmD, and COLM use a more involved invertible mixing function.

COLM is parameterized based on the enumeration blocks after which intermediate tags will be created (τ). For example, COLM127 has intermediate tags while COLM0 does not.

3.7.1. Operation

Encryption key (K), original message, associated data, Npub and a set of parameters are combined so that the ciphertext (C) is generated and intermediate tags (T), which will be used during decryption to retrieve original message M. Once this is complete, tag verification is done for authenticity validation.The general structure of COLM is given in Figure 19. Where, E is an n-bit block cipher, K denotes the key, N the nonce, A associated data, M the message, C the ciphertext, and T the tag.

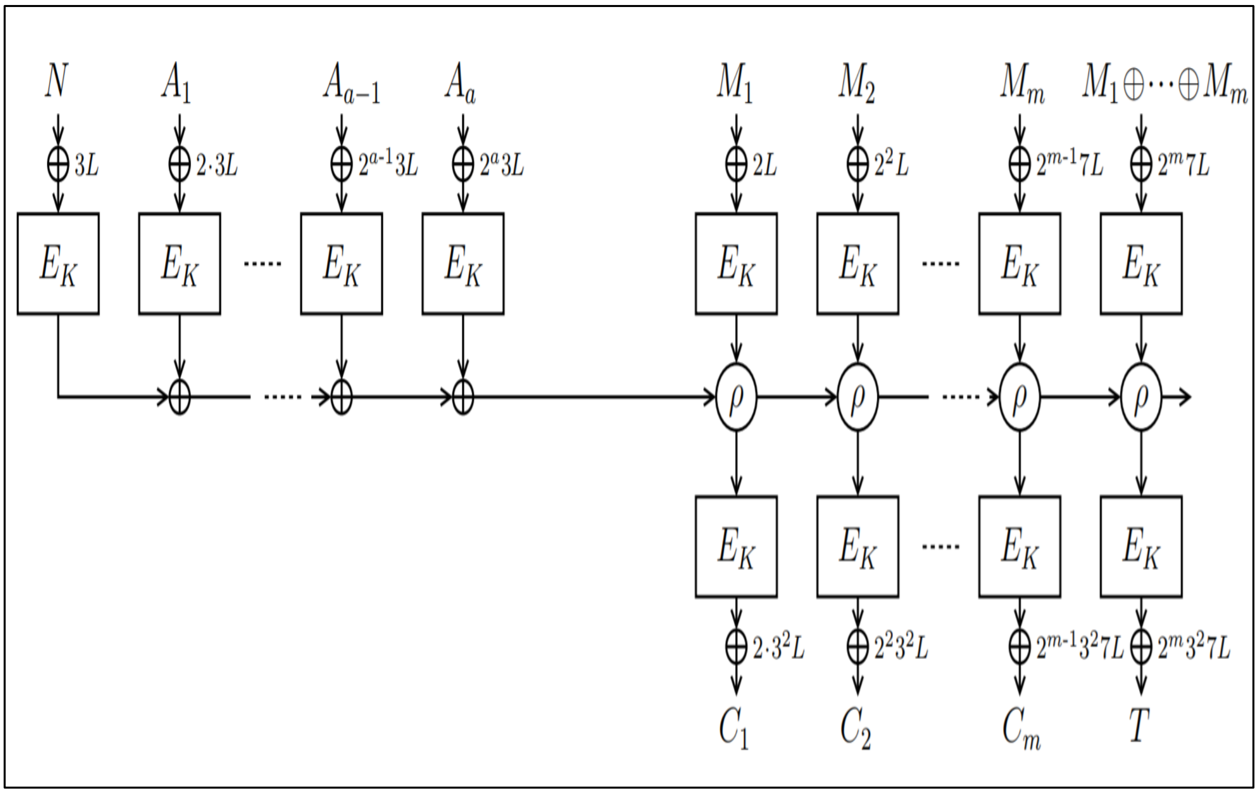


Figure 19: COLM Algorithm [26]

The generic COLM type structure consists of two-layer parallelizable encryption masked with the sub key $L = E_k(0)$ and a counter. COLM mixes the output of the first encryption layer to generate the input to the second encryption layer, using the linear mixing function. COLM is composed of three processes

1) Associated data processing
2) Authenticated encryption
3) Tag Generation.

Associated data processing:

The inputs for COLM are the string of associated data D and L which is computed as $L = E_k(0^{128})$. D is processed in blocks of 128 bit length. Figure 20 show how the associated data is processed basically, they it is based on Parallelizable Message Authentication Code (PMAC) structure which consists in three phases

1) Masking
2) Encryption
3) Mixing.

1) Masking.

The input blocks $D[1], \dots \dots D[d]$ are masked using the masking generation function and L value, generating $DD[i]$ blocks.

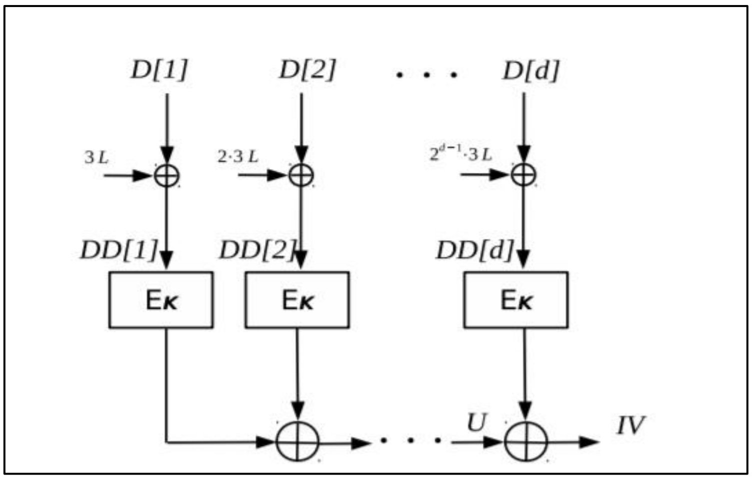


Figure 20: Associated data processing in COLM

2) Encryption.

The second step is the encryption of masked blocks $DD[i]$.

3) Mixing.

The specific mixing function is used to combine the encrypted blocks such that the generated IV depends on all associated data blocks.

Authenticated Encryption:

The input message M is divided in to 128-bits blocks using the function split (M) getting blocks $M[1], M[2], \dots \dots, M[l]$. The encryption is divided into four phases:

- Input-masking
  Each block message $M[i]$ is masked generating the blocks $MM[i]$ using the values generated by $\triangle_M\ [i]$.
- Mixing up
  Blocks $MM[i]$ feed the first encryption layer giving; as a result, $X[i]$ blocks which are fed to the mixing function.
- Mixing down.
  The input of the second encryption layer is the output of the mixing function.
- Output-masking.
  The final step is masking the output of the second encryption layer using $\triangle_c\ [i]$.

Tag Generation

The tag is generated processing the block $M[l+1]$ which contains the checksum of the input blocks. After determined number of blocks are processed, an intermediate tag is generated. A corrupted message can be detected using intermediate tags during verification before all message processing.

Following are the operation performed on message to get the tagged cipher text C

$$\begin{aligned}
\mathrm{W}[0] &= \mathrm{IV}, && \\
\mathrm{MM}[\mathrm{i}] &= \mathrm{M}[\mathrm{i}] \oplus \triangle_M\ [i] && for\ i = 1, \ldots\ldots l+1, \\
\mathrm{X}[\mathrm{i}] &= E_k(\mathrm{MM}[i]) && for\ i = 1, \ldots\ldots l+1, \\
(\mathrm{Y}[\mathrm{i}], \mathrm{W}[\mathrm{i}]) &= \rho(\mathrm{X}[i], \mathrm{W}[i-1]) && for\ i = 1, \ldots\ldots l+1, \\
CC[\mathrm{i}] &= E_k(Y[i]) && for\ i = 1, \ldots\ldots l+1, \\
C[\mathrm{i}] &= CC[\mathrm{i}] \oplus \triangle_c\ [i] && for\ i = 1, \ldots\ldots l+1,
\end{aligned}$$

For decryption, the same steps of encryption are applied in inverse order and using the inverse functions.

3.7.2. Low Area Low Power Optimization

The high-speed COLM implementation instantiates two instances of AES to implement the two layers of encryption. In order to optimize COLM for low area, only one instance of AES is used to perform the two encryption layers. A Finite state machine and Multiplexers are added to control the data flow to the AES. The optimized encryption operation is processed in twice the clock cycles of the non-optimized one and the same applies for the decryption operation.

3.7.3. Results

1) FPGA Results

The results show that the proposed optimized implementations achieve an area reduction for COLM with 28%, and a Dynamic Power consumption reduction by 51%

respectively. As a cost, throughput (TP) decreases for COLM by 50%, and throughput-to-area (TP/A) decreases by 30%.

2) ASIC Results

The results show that the proposed optimized implementations achieve an area reduction for COLM with 38%, and a Dynamic Power consumption reduction by 48% respectively. As a cost, throughput (TP) decreases for COLM by 50%, and throughput-to-area (TP/A) decreases by 19%.

3.8. JAMBU

JAMBU is a nonce-based authenticated encryption operating mode proposed by Wu and Huang [36] that can be instantiated with any block cipher. Yet, the submission AES-JAMBU to the CAESAR competition uses AES-128 [37] as the internal block cipher. The main advantage of JAMBU mode is its low memory requirement, which places it in the group of lightweight authenticated encryption modes. It is not as fast as the parallelizable schemes such as OCB [38] and OTR [39], but it is inverse-free, using only XOR operations, and has a lower state size in the cost of a shorter nonce and tag length [36]. In the encryption of JAMBU, the plaintext is divided into blocks of n-bit. In each step of the encryption, a plaintext block Pi is encrypted to a ciphertext block Ci.

3.8.1. Operation

JAMBU uses a k-bit secret key K and an n-bit public nonce value IV to authenticate a variable length associated data AD and to encrypt and authenticate a variable length plaintext P. It produces a ciphertext C, which has the same bit length with plaintext, and an n-bit tag T.
The encryption process of JAMBU consists of 5 phases:

1) Padding
2) Initialization
3) Processing of the associated data
4) Processing of the plaintext
5) Finalization/Tag generation

The internal state of JAMBU will be represented by the variables (S_i, R_i) with $S_i = (U_i, V_i)$, where R_i , U_i and R_i are n-bit values.

1) Padding

The associated data AD is padded with 10* padding first. The length of the associated date need to be a multiple of n-bit, to achieve this, ‘1’ bit is appended followed by ‘0’ bits. The same methodology is applied on plain text.

2) Initialization

An n-bit public nonce value IV is used by JAMBU to initialize the internal state

3) Processing of the associated data

The padded associated data are processed block by block after being divided to n-bit blocks.

4) Processing of the plaintext

P Is the number of plaintext blocks after padding and $P = (P_1, P_2, \dots . P_p)$. The plaintext is processed block by block. At round i, the internal state is updated with the plaintext block P_i by

$$S_{i+1} = (U_{i+1}, V_{i+1}) = E_k(S_i) \oplus (P_i || \; R_i)$$

And

$$R_{i+1} = R_i \oplus U_{i+1}.$$

The ciphertext block C_i is then computed with $C_i = P_i \oplus V_{i+1}$.

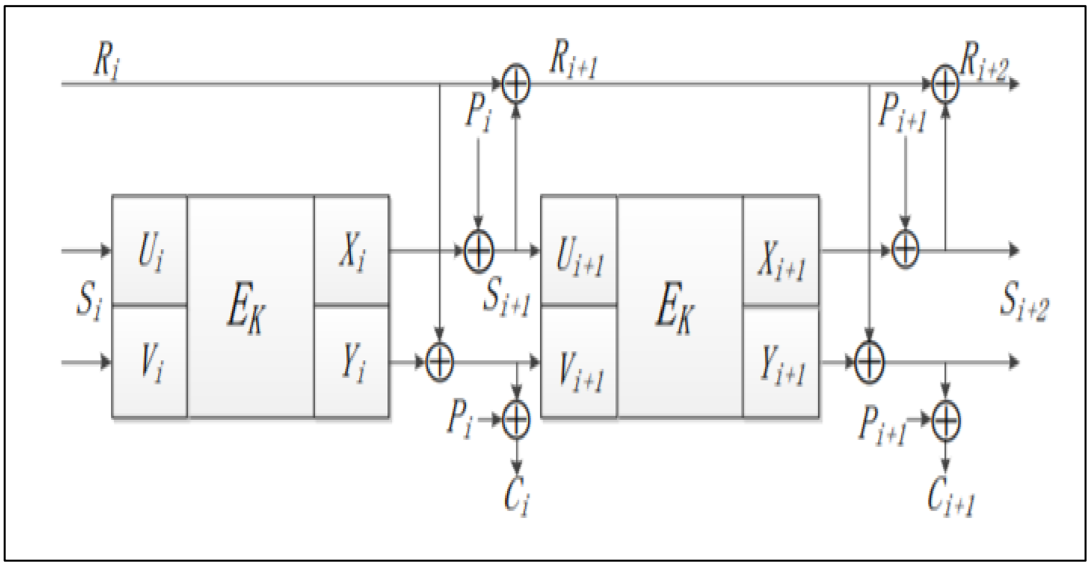


Figure 21: Processing of the plaintext

5) Finalization and tag generation.

After, the procession of all the plaintext blocks. The authentication tag T is generated with two internal block cipher calls.

3.8.2. Low Area Low Power Optimization

In order to optimize JAMBU for low area, The AES core adopts an iterative architecture with an 8-bit data path [28].The AES round operations as well as the key expansion operations are performed sequentially.The MixColumns multiplier performs the matrix multiplication of MixColumns. One column of State is operated separately in four clock cycles. The data is processed byte by byte and four registers are used to maintain the results.

As the same multiplier coefficients are used for each row of a column, only in a cyclically shifted order, a 32-bit part of the MixColumns operation can be performed by adding and cyclically shifting the intermediate results in the unit. The contents of the registers are masked to zero with the en signal during inputting the first byte of a column (bytes 0, 4, 8, and 12).

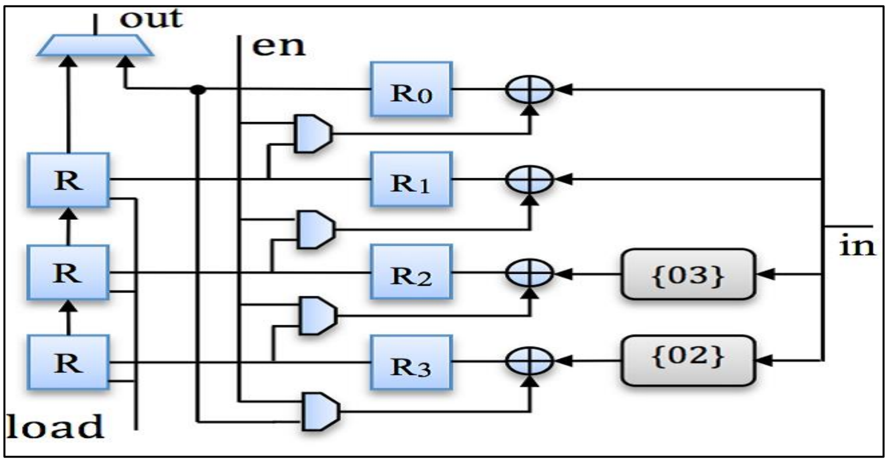


Figure 22: MixColumns multiplier

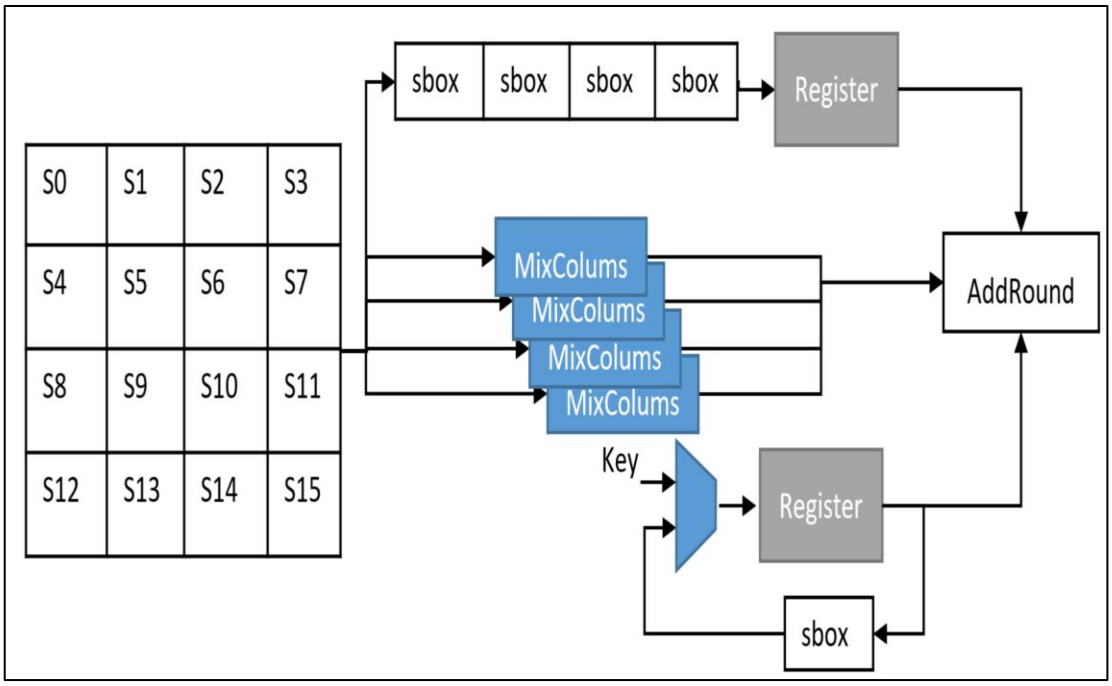


Figure 23: Processing of the plaintext

The MixColumns multiplier performs a complete MixColumns operation in 16 cycles in parallel with the rest of the operations of the AES core. The State register is shifted every clock cycle from the 5 cycles to process each row of the state matrix iteratively and the data from sbox is stored in a register to be used with the output of Mixcolumns when it is ready after 4 clock cycles then it is introduced to the AddRoundKey module with the round key. A small FSM is added to control the flow of data, and in order to account for the additional delay due to the insertion of the pipe stage, few counters are added. As a result of the proposed optimization, one round operation is done in 5 cycles instead of 1 cycle.

3.8.3. Results

3) FPGA Results

The results show that the proposed optimized implementations achieve an area reduction for JAMBU with 30%, and a Dynamic Power consumption reduction by 57% respectively. As a cost, throughput (TP) decreases for COLM by 80%, and throughput-to-area (TP/A) decreases by 71%.

4) ASIC Results

The results show that the proposed optimized implementations achieve an area reduction for JAMBU with 28%, and a Dynamic Power consumption reduction by 30% respectively. As a cost, throughput (TP) decreases for JAMBU by 80%, and throughput-to-area (TP/A) decreases by 72%.

3.9. Conclusion

The hardware performance of the proposed optimized implementations, pairs of corresponding publicly available high speed implementations and proposed Optimized implementations are benchmarked in the FPGA and ASIC implementations.

1) FPGA Results

Results for benchmarking the proposed optimized implementations (denoted by Optimized Implementations) and the corresponding publicly-available HS implementations [4] (donated by High-Speed Implementations) are shown in Table 1. The results show that the proposed optimized implementations achieve an area reduction for NORX, Tiaoxin, SILC, COLM, and JAMBU with 31%, 38%, 33%, 28% and 30% respectively, and a Dynamic Power consumption reduction by 70%, 65%, 39%, 51%, 57% respectively. As a cost, throughput (TP) decreases for NORX, Tiaoxin, SILC, COLM and JAMBU by 87.5%, 83%, 50%, 50%, and 80% respectively, and throughput-to-area (TP/A) decreases by 82%, 73%, 25%, 30% and 71% respectively.

For NORX and SILC the proposed optimized implementations are compared to work proposed in [16]. In [16] virtex-6 FPGA is used for implementation, while virtex-7 FPGA is used in this research so a comparison is done between Area reduction, Dynamic Power reduction and throughput-to-area change achieved by proposed work and the work in [16]. The comparison is summarized in table 2. For NORX proposed implementation has lower area reduction with 31% compared to 53.3% in [16] and the throughput-to-area (TP/A) has decreased with 82% while it increased with 25.5 % in [16]. For SILC proposed implementation has lower area reduction with 33% compared to 69 % in [16] while proposed implementation has less reduction in throughput-to-area (TP/A) with 25% compared to 65% in [16].

2) ASIC Results

Results for benchmarking the proposed optimized implementations and the corresponding publicly-available HS implementations [4] are shown in Table 2. The results show that the proposed optimized implementations achieve an area reduction for NORX, Tiaoxin, SILC, COLM, and JAMBU with 44%, 43%, 26%, 38% and 28% respectively, and a Dynamic Power consumption reduction by 54%, 46%, 39%, 48%, 30% respectively. As a cost, throughput (TP) decreases for NORX, Tiaoxin, SILC, COLM and JAMBU by 87.5%, 83%, 50%, 50%, 48%, and 80% respectively, and throughput-to-area (TP/A) decreases by 77%, 70%, 33%, 19% and 72% respectively.

Low area and low power implementations for five candidates (NORX, Tiaoxin-346, SILC, COLM, and JAMBU) of CAESAR Round 3 are proposed. The optimized implementations and the corresponding high-speed implementations are benchmarked in the Virtex-7 FPGA flow and ASIC flow. For FPGA flow a reduction in area with an average of 32% and a reduction in dynamic power with an average of 56% are achieved compared to their corresponding high-speed architectures. Moreover, throughput (TP) in (Mbps) decreases by an average of 70% and throughput-to-area (TP/A) in (Mbps/Slices) decreases by an average of 56%.

For ASIC flow a reduction in area with an average of 36% and a reduction in dynamic power with an average of 43% are achieved compared to their corresponding high-speed architectures. Moreover, throughput (TP) in (Mbps) decreases by an average of 70% and throughput-to-area (TP/A) in (Gbps/mm2) decreases by an average of 54%. The reduction in TP and TP/A ratio is expected as latency and throughput are sacrificed for area reduction.

Table 1 Comparison of Results to Work in [16]

Algorithm	Area Reduction [%]	Dynamic Power Reduction [%]	TP/Area Change [%]
Work Proposed in [18]			
NORX	53.3	82	+25.5
SILC	69.1	29	-65
Optimized implementation			
NORX	31	70	-82
SILC	33	39	-25

Table 2 Results of Implementation in virtex 7 FPGA

Algorithm	Area [Slices]	Reduction [%]	Dynamic Power [mW]	Reduction [%]	Freq [MHz]	TP [Gb/Sec]	Reduction	TP/Area [Mbps/Slices]	Reducti on [%]
High Speed Implementations									
NORX	1367	-	416	-	100	19.2	-	14.04	-
Tiaoxin	2030	-	527	-	100	25.6	-	12.6	-
SILC	984	-	230	-	100	1.28	-	1.3	-
COLM	2149	-	149	-	100	1.16	-	0.54	-
JAMPU	511	-	106	-	100	0.64	-	1.25	-
Optimized Implementations									
NORX	949	31	127	70	100	2.4	87.5	2.53	82
Tiaoxin	1250	38	183	65	100	4.27	83	3.42	73
SILC	662	33	140	39	100	0.64	50	0.97	25
COLM	1543	28	73	51	100	0.58	50	0.38	30
JAMPU	357	30	46	57	100	0.128	80	0.36	71

Table 3 Results of Implementation in ASIC

Algorithm	Area [Slices]	Reduction [%]	Dynamic Power [mW]	Reduction [%]	Freq [MHz]	TP [Gb/Sec]	Reduction	TP/Area [Mbps/Slices]	Reducti on [%]
High Speed Implementations									
NORX	187266	-	9.47	-	100	19.2	-	102.5	-
Tiaoxin	402603	-	221.84	-	100	25.6	-	63.6	-
SILC	139225	-	11.53	-	100	1.28	-	9.2	-
COLM	529777	-	19.15	-	100	1.16	-	2.2	-
JAMPU	80924	-	4.79	-	100	0.64	-	7.9	-
Optimized Implementations									
NORX	103992	44	4.37	54	100	2.4	87.5	23.1	77
Tiaoxin	228271	43	11.68	46	100	4.27	83	18.7	70
SILC	103109	26	7	39	100	0.64	50	6.2	33
COLM	327476	38	12.08	48	100	0.58	50	1.8	19
JAMPU	58073	28	3.39	30	100	0.128	80	2.2	72

Chapter 4 Using Dynamic partial Reconfiguration to achieve energy efficient and resource efficient Hardware Encryption

This chapter presents Using Dynamic partial Reconfiguration to achieve energy efficient and resource efficient Hardware Encryption , first it gives introduction for dynamic partial reconfiguration ,then it goes to how Partial dynamic reconfiguration could be used to implement resource efficient and energy efficient hardware encryption.

4.1. Configuration Definition

Configuration is a complete design programmed on the FPGA. FPGA can be considered as device with two-layers: configuration memory layer and logic layer. The configuration stored on the configuration memory layer, will control the logic layer.

There are three types of FPGA configurations:

1) Fixed Configuration:
At power-on data is loaded, till the end of the FPGA cycle the configuration will remain fixed.

2) Partial Reconfiguration:
At power-on the initial full bit file is loaded into the FPGA. The FPGA will stop whenever something to be altered, then a partial bit file that contains the modification is loaded.

3) Dynamic Partial Reconfiguration:
Unlike the partial reconfiguration, during the loading of the data into the FPGA, the FPGA continues its normal operation, except for the part subjected to the modification.

4.2. Dynamic Partial Reconfiguration

DPR is a SRAM-FPGAs feature that gives the flexibility to reconfigure a part of FPGA during runtime reusing the same resources of hardware. The DPR design flow in Xilinx requires partitioning of the design into two parts, a static part and a dynamic part. The static part contains the static modules that are not going to change during the reconfiguration, while the dynamic part contains the system reconfigurable modules (RM). The dynamic part contains multiple Reconfigurable Regionss (RRs), each RR has contains a set of RMs which can be exchanged without the interruption of the system during runtime. During configuration, for each RM there will be a partial bit stream generated to be mapped into a specific RR. Partial bit streams are loaded from a nonvolatile memory to the FPGA configuration memory through dedicated configuration interfaces.

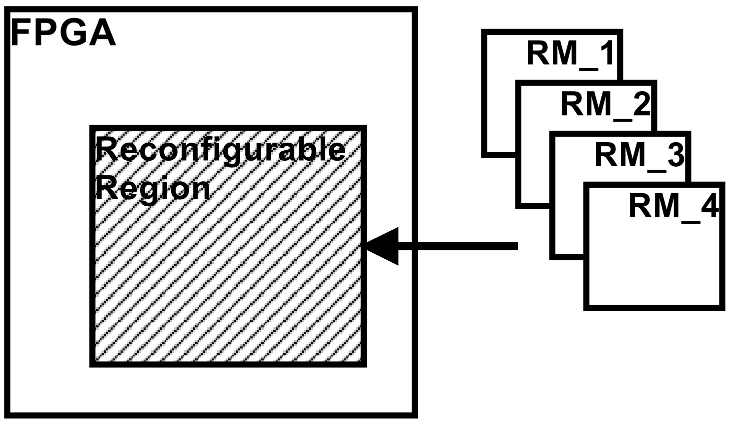


Figure24 **: Dynamic Partial Reconfiguration in SRAM-FPGAS**

4.3. Configuration Modes

During DPR, RMs partial bit streams are loaded to the FPGA configuration memory. Accessing the configuration memory is done through various FPGA configuration modes or configuration ports. The configuration modes are classified based on the type of configuration interface used in accessing the configuration memory. Table 4 shows the different configuration modes for Zynq FPGA.

4.3.1. External Modes

In External configuration modes, external FPGA interfaces are used to load the partial bit files to the FPGA configuration memory. Zynq FPGA has only one external configuration port which is JTAG. The partial bit streams are transferred from an external storage source, for example, the PC to the configuration memory through the JTAG serial interface.

4.3.2. Internal Modes

In Internal configuration modes, internal FPGA interfaces are used to load the partial bit files to the FPGA configuration memory. Xilinx Zynq FPGA has two internal configuration modes.

1) The Internal Configuration Access Port (ICAP) configuration mode is based on the ICAP hard macro 32-bit configuration port primitive located on the PL side to access the configuration memory with a theoretical data rate of 400 MB/S.

2) Processor Configuration Access Port (PCAP) configuration mode is based on the PCAP 32-bit configuration port in the PS side controlled by the ARM processor to access the configuration memory with a data rate of 400 MB/S

Table 4 Zynq FPGA Configuration Modes

Configuration Mode	Type	Max Clock	Data Width	Max Bandwidth
ICAP	Internal	100 MHZ	32-bit	400 MB/S
PCAP	Internal	100 MHZ	32-bit	400 MB/S
JTAG	External	66 MHZ	1-bit	8.25MB/S

4.4. Advantage and Disadvantage of DPR

The main advantages of the reconfigurable systems are:

1) Resources utilization: DPR will increase the resource utilization as each part of design is implemented in the required time, and it allows time multiplexing between the modules of the design according to schedule of activity.

2) Scalability: Using reconfigurable systems gives the ability to update the system to handle new tasks defined due to the growth. It also make it easier to deploy enhancements and bug fixes to the system without the need to redeploy new hardware.

3) Reusability: Reusing the resources for different design implementations, where a system can be customized for adaptability.

4) Power reduction: This is considered the most important item. In the Integrated Circuits (IC) design the static power is consumed although the device is idle. FPGA reconfiguration can be used to delay implementation of a specific part until its time of operation, which will decrease the static power consumption.

5) Area: Rather than horizontally implementing a complete system that consumes area, the system can be optimized vertically by implementing the concept of space and time programming. Where the block stack is stored and loaded at the time of operation. This saves the area used by the same blocks in the horizontal design.

On the contrary, there are some disadvantages for the DPR and they are improving by research such as:

1) Latency: The reconfiguration time will add latency to the design. It could be improved by speeding the reconfiguration time through using high-speed PR controller.

2) Memory: storing the reconfigurable blocks will require extra memory.

4.5. Dynamic Partial Reconfiguration Controller

To enhance the reconfiguration speed and maximize the reconfiguration throughput researchers proposed Partial Reconfiguration (PR) controllers [40]. PR controller provides the interface for loading the partial bit stream from an external or internal memory to the FPGA internal configuration port (i.e., ICAP or PCAP) with a high data throughput. Moreover, some PR controller architectures have the ability to monitor the system performance by determining the status of RMs and measuring the reconfiguration time.

Reconfiguration time depends on the generated partial bit stream data size, the dimension of RP, and the memory configuration setups used for data transfer. The key factor in the PR controller design is the reconfiguration time as it measures how fast the controller can handle the reconfiguration process. There are many PR controllers that are used in DPR, they are either conventional controllers provided by the FPGA vendors or novel controllers developed to have more efficient DPR with less configuration time. PR controller is implemented by a dedicated custom processor using or a Finite State Machine (FSM) to control and manage the DPR

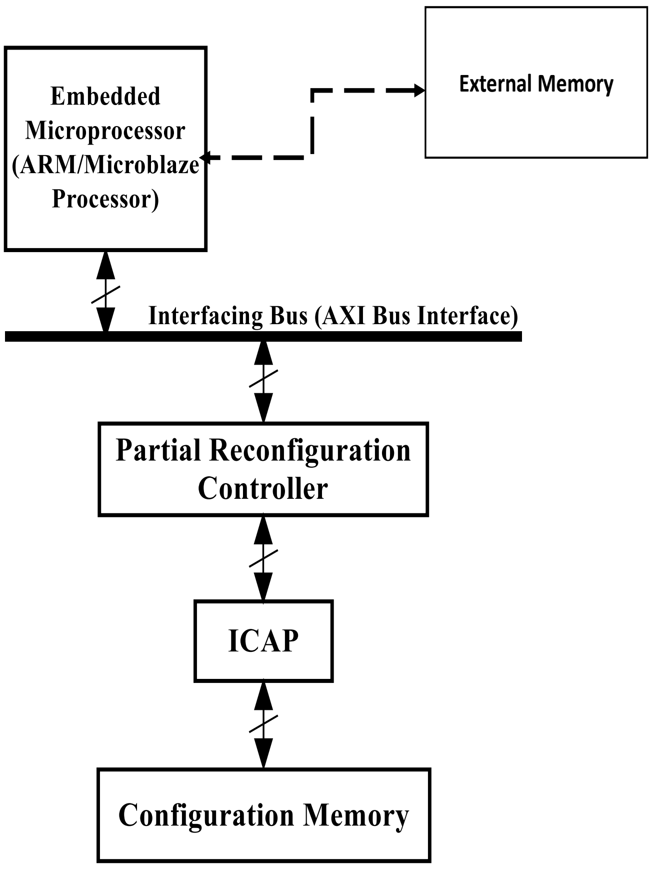

Embedded Microprocessor (ARM/Microblaze Processor)
External Memory
Interfacing Bus (AXI Bus Interface)
Partial Reconfiguration Controller
ICAP
Configuration Memory

Figure :25 **Partial Reconfiguration Controller in DPR System [45]**

Following are the most common Dynamic Partial Reconfiguration Controllers available:

4.5.1. Xilinx ICAP Controller (AXI-HWICAP)

Multiple IP cores are provided by Xilinx for interfacing the Xilinx's ICAP primitive with the user design. ICAP Controllers enable an embedded microprocessor such as ARM processors or Microblaze to be used in accessing the configuration memory. ICAP provides access directly to the configuration memory both in read and write modes. In Xilinx 7-series, the ratio of the data width of the ICAP interface to the configuration memory is 8, 16 or 32 bit wide. The ICAP provides a maximum throughput of reconfiguration theoretically equal to 400 MB/S at a clock frequency of 100MHZ and data width of 32 bits.

4.5.2. Xilinx Partial Reconfiguration Manager

Xilinx Partial Reconfiguration Controller depends on the concept of Virtual Sockets (VS) [47]. Xil-PRC is released for enclosed systems where all the design RMs are known to the controller. The VS represents the Reconfigurable Partition (RP) associated with some logic blocks used to isolate it from the static region during reconfiguration process. VSMs are connected to a fetch path that fetches the partial bitstream data from an external memory to the ICAP without passing by the processor which leads to a short reconfiguration time.

4.5.3. Custom DMA Based ICAP Controller

Various Open Source ICAP Controllers are proposed by researchers [48-52] to improve the reconfiguration time while using the ICAP through an embedded microprocessor. A custom reconfiguration controller for Xilinx Zynq FPGA (ZYCAP) is presented in [52]. ZYCAP achieves a reconfiguration throughput of 382 MB/S. ZYCAP is a DMA based AXI-HWICAP controller equipped with two AXI slave bus interfaces connected to the ARM processor. The AXI4 bus interface receive partial bitstream data from DDR memory and the AXI-Lite bus interface receive control signals.

4.5.4. Software-Controlled Partial Reconfiguration

Xilinx Zynq FPGA provides the potential to implement a software-controlled (S-C) DPR through the processing system (PS) device configuration (DevC)/PCAP interface [53]. This scheme does not require any programmable logic (PL) resources during DPR. The ARM processor control the DevC unit is to select the internal configuration interface to be ICAP or PCAP according to the user design.

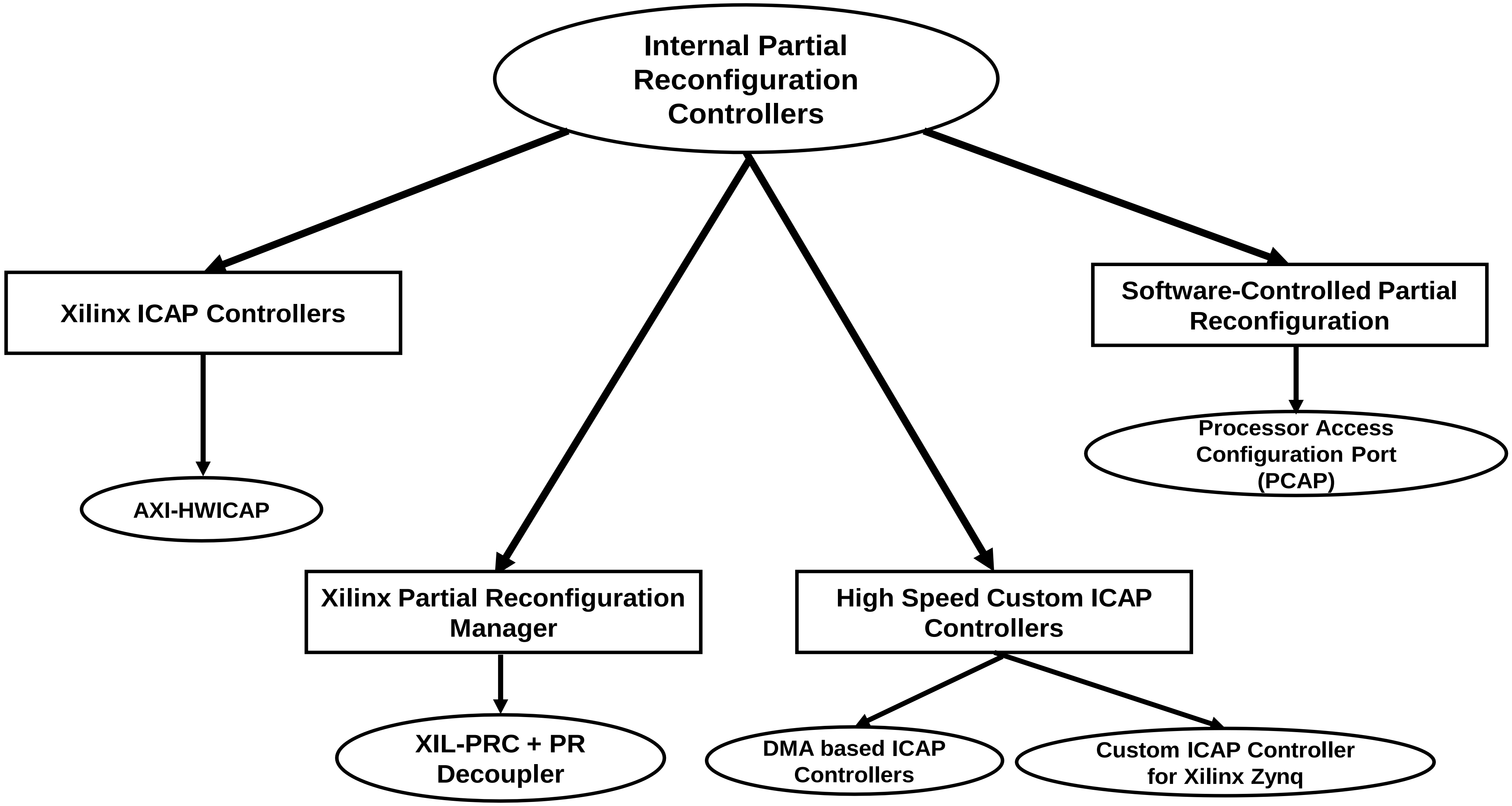

Internal Partial Reconfiguration Controllers
Xilinx ICAP Controllers
Software-Controlled Partial Reconfiguration
Processor Access Configuration Port (PCAP)
AXI-HWICAP
Xilinx Partial Reconfiguration Manager
High Speed Custom ICAP Controllers
XIL-PRC + PR Decoupler
DMA based ICAP Controllers
Custom ICAP Controller for Xilinx Zynq

Figure:26 **Types of Internal Partial Reconfiguration Controllers [45]**

In [45] the partial reconfiguration controllers discussed above (AXI-HWICAP, S-c/PCAP, ZYCAP, Xil-PRC) are used to implement a high-speed reconfigurable Software Defined Radio (SDR) system targeting a Xilinx Zynq FPGA. A reconfigurable convolutional encoder is benchmarked to do performance evaluation of the four partial reconfiguration controllers.

Figure 30. Shows the reconfiguration time of the four types of PR controllers discussed with various reconfigurable partition regions sizes. The minimum reconfiguration time is achieved by Xil-PRC and DMA-based ICAP controller (ZYCAP). AXI slave memory mapped AXI-HWICAP is the worst PR controller in term of reconfiguration time.

Xil-PRC and ZYCAP will be always recommended for designs that require high reconfiguration speed as shown in Figure 31. For designs that require low Power S-C / PCAP will be recommended as it do not require any resources on the PL side for DPR Therefore, the power consumed by the ARM during the reconfiguration is the only power consumed, but this scheme has downside which is the ARM processor during the reconfiguration time is blocked from doing other tasks. In this work we will be using the S-C/PCAP as has the less power consumed and no area overhead, and this will fit for the low area low power constrained designs for IoT.

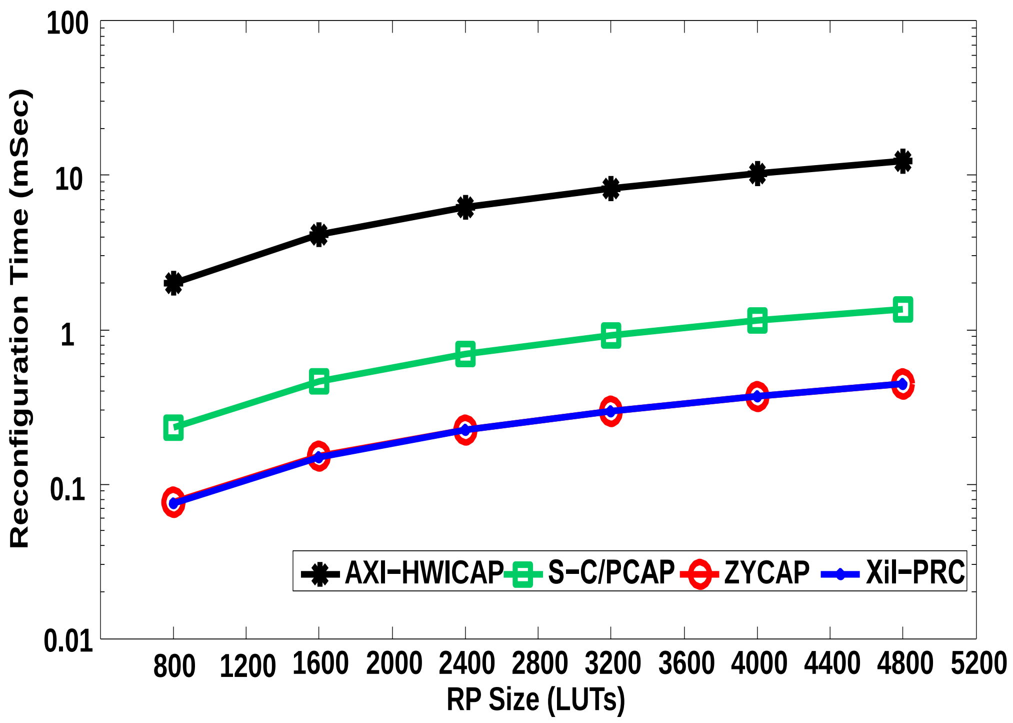


Figure27 **Reconfiguration Time for Different PR Controllers [45]**

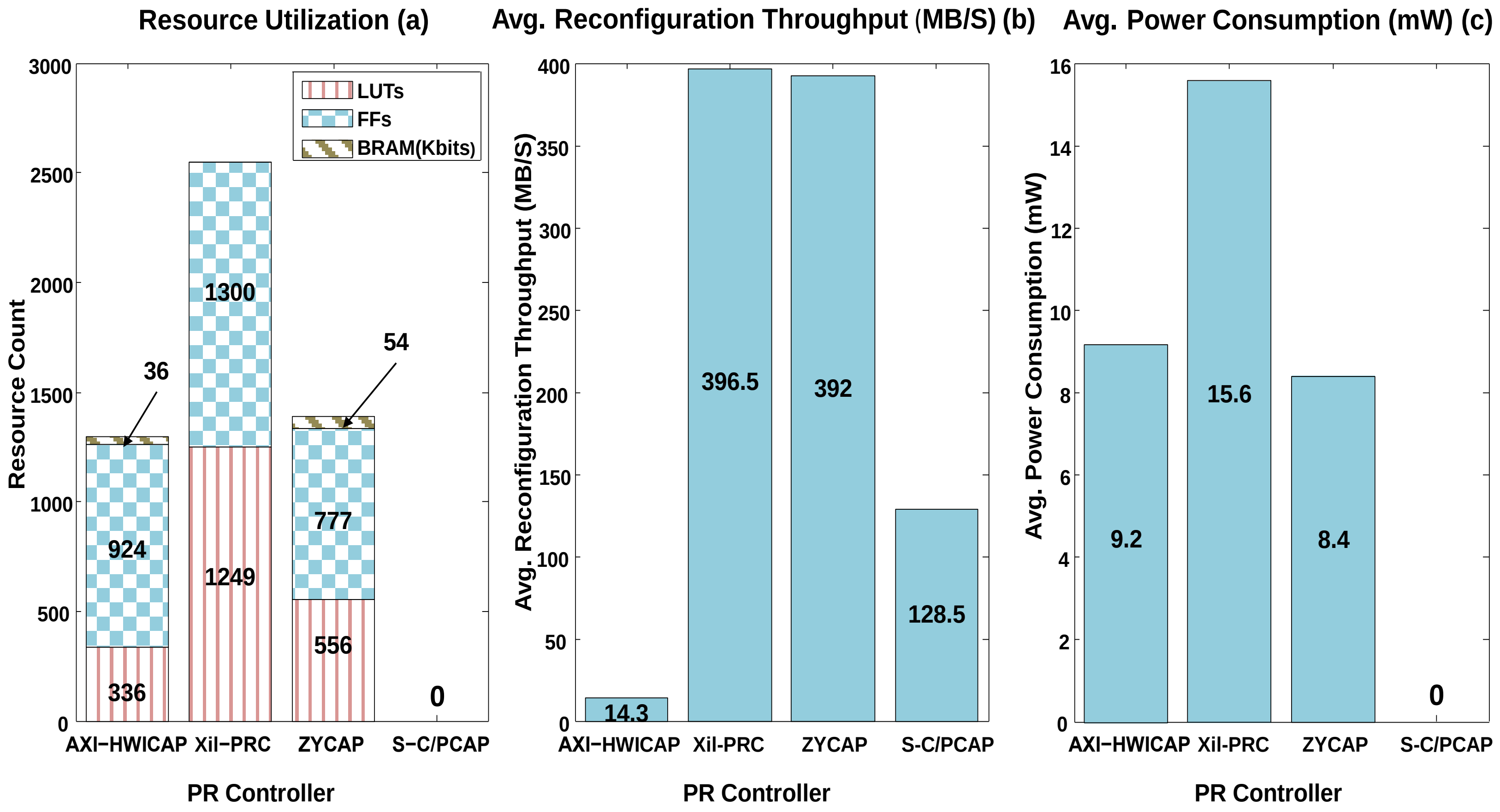


Figure28 **(a) Resource Utilization, (b) Avg. Reconfiguration Throughput and (c) Power Consumption Comparisons between different PR controllers [45**

4.6. Resource Efficient Hardware Implementation for AEAD Ciphers Using PDR

The Crypto processor is a processor that can be used to work as encryption processor or decryption processor based on a 1-bit control signal that goes to the processor. The control signal selects whether the data-buffer output will hold the cipher text or decrypted message data. The presence of this control signal increased the flexibility of the processor in a way that with the help of run-time reconfiguration technology. Our Proposed design methodology use the PDR to switch between decryption and encryption based on the value of the control signal. The proposed methodology will be sufficient in case the decryption logic is different from the encryption logic, in this case the area for decryption can be saved during encryption as it is not needed which will decrease the power consumption as well the area.

We studied some of the Caesar candidates and defined two categories of ciphers

1) Ciphers that have a separate round for encryption and decryption, here the PDR would be beneficial and could be used to switch between the Encryption and Decryption modes, as the resources for decryption will be unused during encryption and vice versa. We will focus on those ciphers in the thesis.

2) Ciphers that do not have a separate round for encryption and decryption, the DPR would not be so beneficial because the same resources are used in both encryption and decryption

Based on the study we have identified two Ciphers that will be targeted for our work, COLM, OCB. Figure 29 Describe the Round function for OCB, and COLM it is very similar in both cases. The Round function is used in encryption and the InvRound is used in decryption, as it both have a separate data path and separate logic.

COLM was discussed in the previous chapter. Offset Codebook (OCB) mode of operation is based on block cipher, it provides simultaneously authenticity and privacy for the plaintext. Despite this, OCB is clean, easy and simple to be implemented in either software or hardware.

OCB accomplishes its work without bringing in the machinery of universal hashing, a technique that does not seem to lend itself to implementations that are simple and fast in both hardware and software. In 2001 the initial version of OCB was presented. In FSE 2011, (OCB3) which is the current version was presented and accepted as RFC 7253 [57]. The tweakable block cipher structure used by OCB makes the power analysis attack difficult compared to other block ciphers. The plaintext is processed in 128-bit blocks and produces 128-bit ciphertext, and tags with lengths of 64, 96, and 128 bits. This cipher supports key lengths of 128, 192, and 256

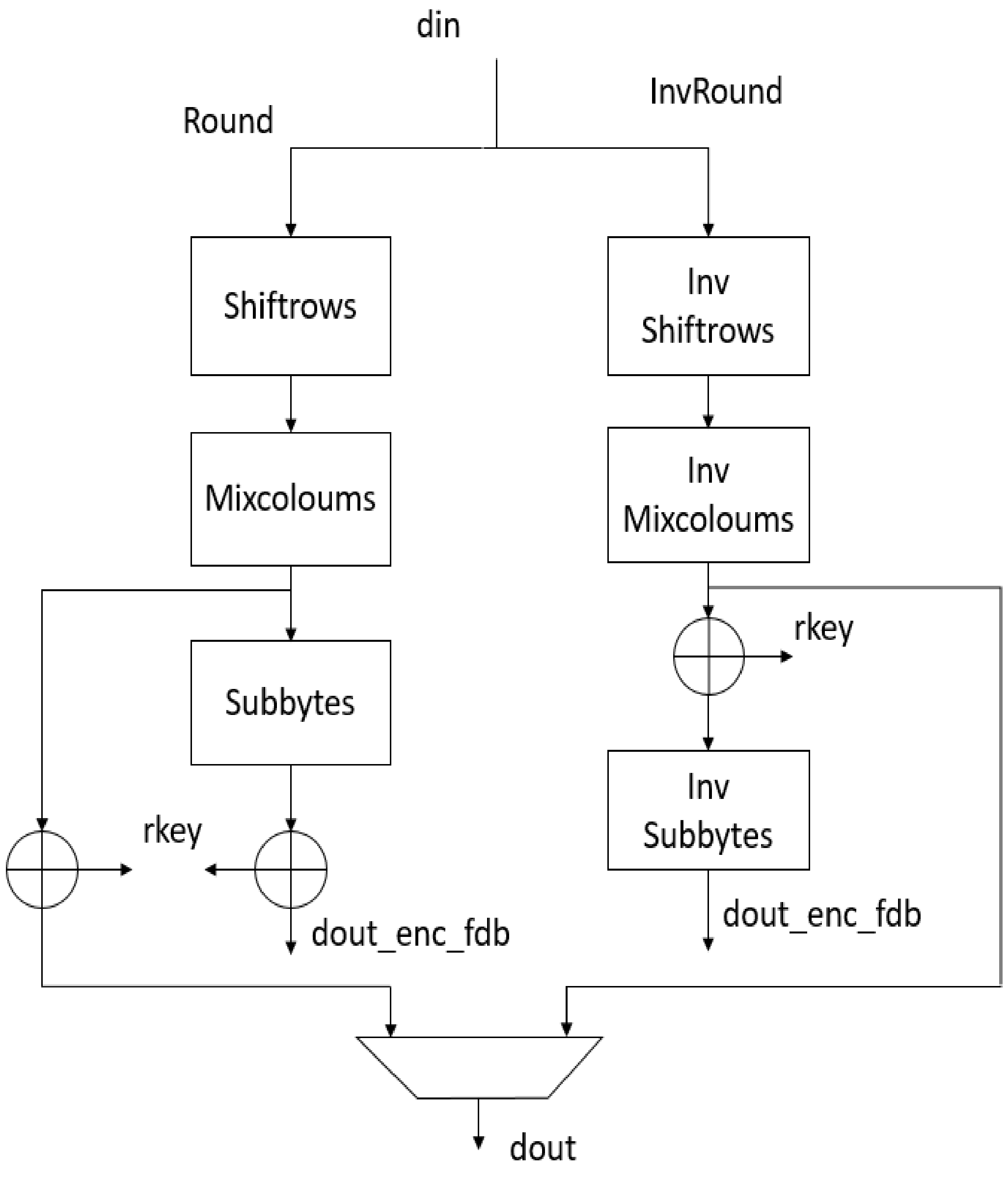


Figure 29 COLM, OCB Round Function

In the top section of Figure 30, the calculation of masking values (Δ) is shown. The nonce N is 96 bits which use 10*-padding to make the 128-bit block. For each encryption, the new values for the nonce are generated. If the nonce is used with the same key, the confidentiality and authenticity of the scheme will be endangered .

The ciphertext is calculated $C[i] = E_k\big(M[i] \oplus Z(i)\big) \oplus Z[i]$ for each 128-bit block of M[i] plaintext. AD is processed in the bottom part of Fig. 3 and used to calculate the final tag.

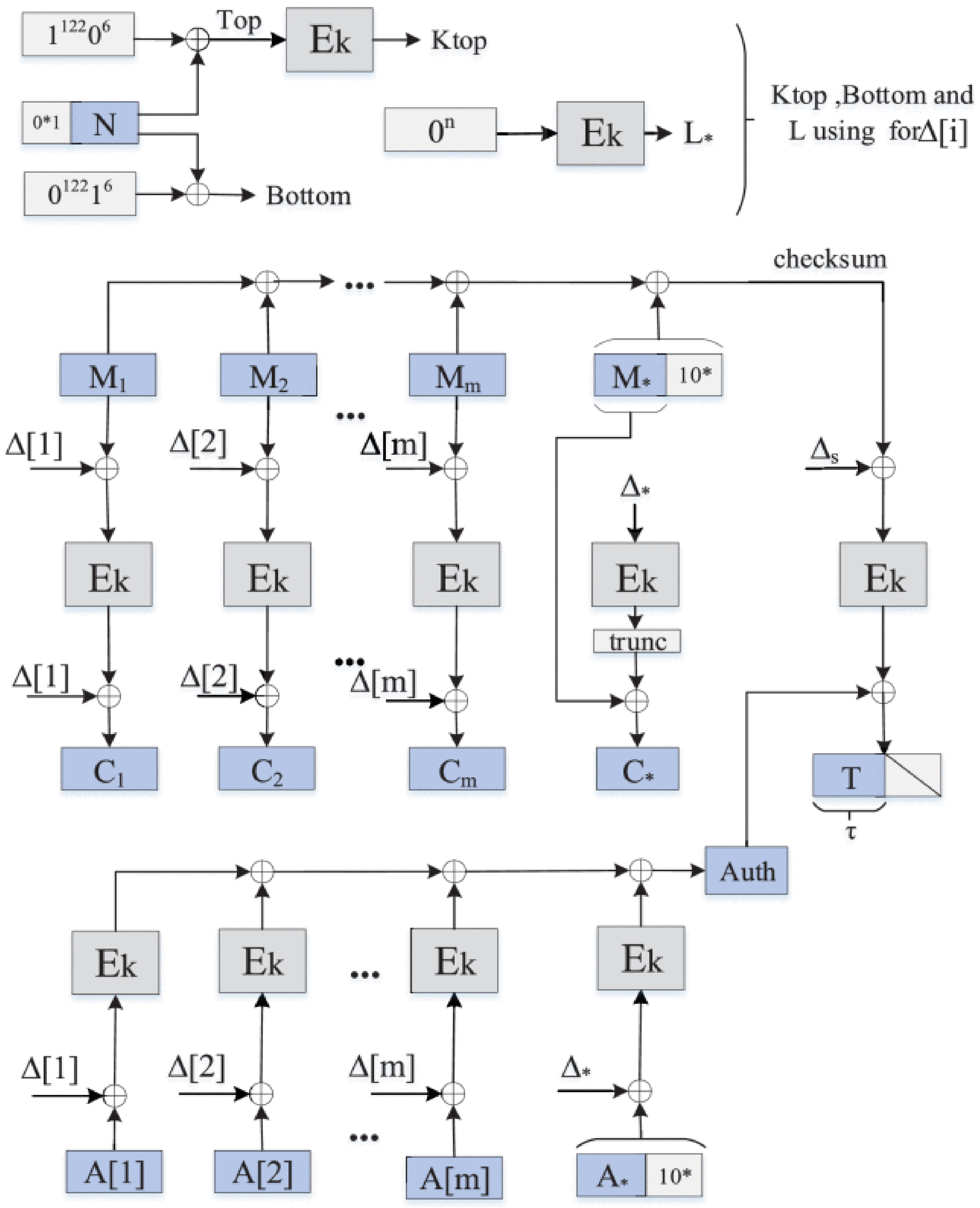


Figure 30 Illustration of OCB authenticated cipher [56] (N: nonce, Auth: authenticator for AD, trunc: truncate the least significant bits, τ : tag length)

4.6.1. Proposed System Overview

The proposed use the DPR is to alternate the operation mode between encryption and decryption based on one control signal the control the processor either to perform encryption or decryption operation.

4.6.1.1. COLM Reconfigurable Crypto processor

PDR is applied to the AES core in the COLM Cipher, COLM has 2 AES cores as discussed in the previous chapter. Figure 31. Describe the design architecture, the top module consists of two parts: a dynamic and a static part. The static part is Cipher controller, and Cipher Data path. The dynamic part is the AES Cores, here we have 2 RPs for each AES core, and each RP has two RMs (encryption and decryption).

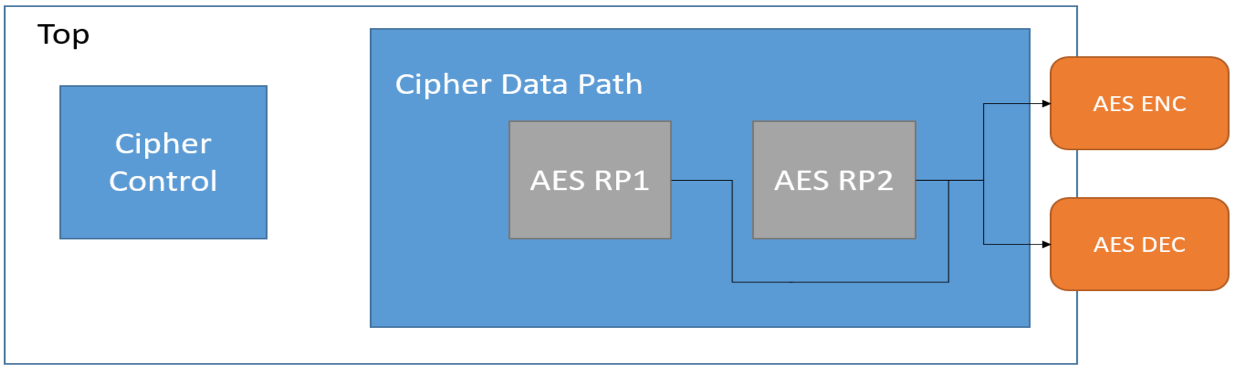


Figure 31 COLM Reconfigurable Crypto processor

4.6.1.2. OCB Reconfigurable Crypto processor

Similar to COLM, PDR is applied to the AES core in the OCB Cipher, but COLM has only one AES core. Figure 32. Similar to COLM. The static part is Cipher controller, and Cipher Data path. The dynamic part is the AES Core, here we have one RP, which has two RMs (encryption and decryption).

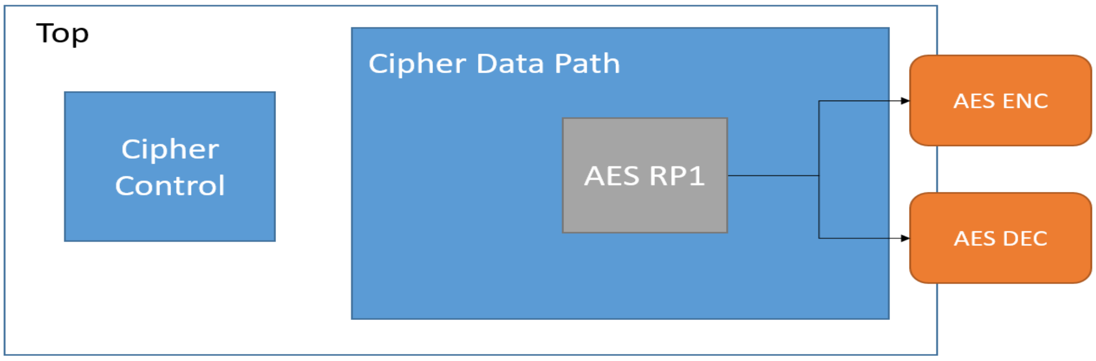


Figure 32 OCB Reconfigurable Crypto processor

4.6.2. Reconfiguration Time

The bit size assigned to the RP determine the reconfiguration time of RPs depends on the bit size assigned to the RP. The bit size for each RP is about 0.3 MB. The reconfiguration times for the one RP is around 0.75 msec. This makes the reconfiguration time for COLM 1.5 msec as it has 2 RMs and 0.75 for OCB as it has only one RM.

The reconfiguration take place on switching between encryption and decryption, and on changing the Key during decryption as it requires the Round function which exist in the encryption module. So the throughput for encryption is not affected, and for decryption it will be affected only in case the key changed during decryption, if the key changed during encryption the decryption throughput will not be affected. Therefore, we can assume the decryption throughput will not change as the key change will not take place frequently. Hence the reconfiguration time is not a concern here.

4.6.3. Resource Utilization

Table 5, 6 shows the resource utilization of the OCB and COLM proposed reconfigurable crypto processor against the corresponding publicly-available HS implementations [4]. The resource utilization decreases from 3617 LUTs to 2324 LUTs by 35% for OCB, decreased from 6827 LUTs to 4146 LUTs by 40% for COLM. The reduction in area for COLM is larger than OCB because COLM has 2 RPs.

Table 5 OCB Resource Utilization

	OCB Crypto Processor[4]	OCB Proposed Crypto Processor	Reduction [%]
Number of LUTs	3617	2324	35%
Number of Registers	1116	988	11%

Table 6 COLM Resource Utilization

	COLM Crypto Processor[4]	COLM Proposed Crypto Processor	Reduction [%]
Number of LUTs	6827	4146	40%
Number of Registers	2302	1789	22%

4.6.4. Power Consumption and Energy Efficiency

Table 7, 8 show the dynamic power consumption for the OCB and COLM proposed reconfigurable crypto processor against the corresponding publicly-available HS implementations [4]. The dynamic power has decreased from 46 mW to 29 mW by 37% for OCB and decreased from 105 mW to 70.6 mW by 33% for COLM.

The throughput is not affected for encryption, decryption as discussed above. Hence the energy consumed by the proposed crypto processor will decrease. The energy consumed for OCB decreased from 43 pJ/b to 27 pJ/b and decreased from 105 pJ/b to 70.6 pJ/p for COLM which makes the proposed crypto processor more energy efficient and consume less power.

Table 7 OCB Energy Utilization

	OCB Crypto Processor[4]	OCB Proposed Crypto Processor	Reduction [%]
Dynamic Power [mW]	46	29	37%
TP [Gb/Sec]	1.07	1.07	-
Energy [pJ/b]	43	27	37%

Table 8 COLM Energy Utilization

	COLM Crypto Processor [4]	COLM Proposed Crypto Processor	Reduction [%]
Dynamic Power [mW]	122	82	33%
TP [Gb/Sec]	1.16	1.16	-
Energy [pJ/b]	105	70.6	33%

4.7. Energy Efficient Hardware Implementation for AEAD Ciphers Using PDR

Because of the overhead necessary for the initialization or finalization of the ciphers, the energy consumption per bit also depends on the length of a message. In [58] Author compared energy consumption for different algorithms Joltik, Ascon and Morus. Morus was found to have long initialization and finalization stage. If the aim is to encrypt short messages, the cipher will be a lot less energy efficient. Ascon-128a is more efficient than Morus-640 if the message is under five data blocks long (0.64 Kb)

In order to implement energy efficient hardware encryption, PDR would be used to switch between different algorithms depending on the message length. If the message length is less than 0.64 Kb the crypto processor is configured to be Ascon and if the message length is more than 0.64 Kb the crypto processor is reconfigured to be Morus.

The Proposed design will be energy efficient as it select the most energy efficient algorithm based on the message length, this will be gained on the cost of reconfiguration time that will be in range of milliseconds, which make this flow suitable for application operating in frequencies of range 1 KHz. The Proposed Design will have only one RP, with 2 RMs (one for Ascon and the other one of Morus).

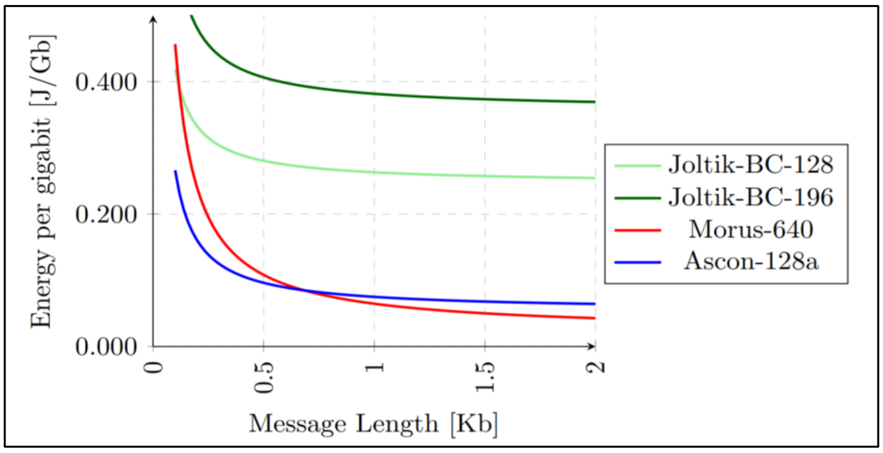


Figure 33 Energy Consumption [58]

4.8. Conclusion

In this chapter two design methodologies were proposed, the methodologies utilize PDR to have resource-efficient energy-efficient hardware encryption.

The first methodology utilize PDR to switch the crypto processor between decryption and encryption. The methodology was applied on two algorithms COLM and OCB. The methodology reduced the area for COLM by 40% and for OCB by 35%, in addition to reducing the energy for COLM by 33% and OCB by 37%.

The second methodology utilize PDR to switch the cipher based on the message length to achieve energy-efficient hard ware encryption. The switch is done between Morus and Ascon algorithms based on the message length.

Chapter 5 Conclusion and Future Work

5.1. Conclusion

The objective of the thesis is to optimize AEAD Ciphers for low power and low energy to fit in IoT low constrained devices, the target is met by

1. Optimizing five candidates of CAESAR for power and area reduction. The proposed optimization For FPGA flow a reduction in area with an average of 32% and a reduction in dynamic power with an average of 56% are achieved compared to their corresponding high-speed architectures.

2. Utilizing PDR to switch the crypto processor between decryption and encryption. The methodology was applied on two algorithms COLM and OCB. The methodology reduced the area for COLM by 40% and for OCB by 35%, in addition to reducing the energy for COLM by 33% and OCB by 37%.

3. Utilizing PDR to switch the cipher based on the message length to achieve energy-efficient hard ware encryption.

5.2. Future Work

1. Explore more CEASAR candidates for low area and low power optimization.

2. Combine the work proposed in chapter 3 and chapter 4 together, to achieve more power and area reduction.

References

1. Biryukov and L. Perrin, “State of the Art in Lightweight Symmetric Cryptography,” Cryptology ePrint Archive, Report 2017/511, 2017, https://eprint.iacr.org/2017/511.
2. Engels, O. Saarinen, P. Schweitzer and E. Smith, “The Hummingbird-2 Lightweight Authenticated Encryption Algorithm,” RFID. Security and Privacy: 7th International Workshop, RFIDSec ’11, Amherst, Massachusetts, USA,June 2011.
3. Bogdanov, F. Mendel, F. Regazzoni, V. Rijmen, and E. Tischhauser, “ALE: AES-Based Lightweight Authenticated Encryption,” 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013E. Homsirikamol, W. Diehl, A. Ferozpuri, F. Farahmand, and K.
4. Homsirikamol, E., Gaj, K.: An Alternative Approach to Hardware Benchmarking of CAESAR Candidates Based on the Use of High-Level Synthesis Tools. https://cryptography.gmu.edu/athena/presentations/GMU_DIAC_2016_HLS.pdf 2016.
5. Jalouli, Ons, “Chaos-based security under real-time and energy constraints for the Internet of Things”, 2017.
6. Homsirikamol, W. Diehl, A. Ferozpuri, F. Farahmand, and K.Gaj, “Implementers Guide to Hardware Implementations Compliant with the CAESAR Hardware API, v2.0,” [Online]. Available: https://cryptography.gmu.edu/athena/CAESAR HW API/CAESARHW Implementers Guide v2.0.pdf.
7. Mamidi, U, “LIGHTWEIGHT AUTHENTICATED ENCRYPTION FOR FPGAS”. Diss. Master’s thesis, George Mason University, 2016.
8. S. Babbage, C. D. Canniere, A. Canteaut, C. Cid, H. Gilbert, T. Johansson,M. Parker, B. Preneel, V. Rijmen, and M. Robshaw, “The estream portfolio(rev. 1).” ECRYPT Network of Excellence, 2008. http://www.ecrypt.eu.org/.
9. “Announcing request for candidate algorithm nominations for a new cryptographichash algorithm (sha-3) family.” National Institute of Standards and Technology,Docket No. 070911510-7512-01, 2007. https://www.gpo.gov/fdsys/pkg/FR-2007-11-02/html/E7-21581.htm/.
10. X. Wang and H. Yu, “How to break md5 and other hash functions.” EUROCRYPT’05 Proceedings of the 24th annual international conference on Theoryand Applications of Cryptographic Techniques, pp. 19-35, 2005.
11. V. Rijmen and E. Oswald, “Update on sha-1.” Cryptology
12. Homsirikamol, E., Farahmand, F., Diehl, W., Gaj, K.: Benchmarking of Round 3 CAESAR Candidates in Hardware: Methodology, Designs & Results. https://cryptography.gmu.edu/athena/presentations/CAESAR_R3_HW_Benchmarking.pdf (2017)

13. Homsirikamol, E., Gaj, K.: An Alternative Approach to Hardware Benchmarking of CAESAR Candidates Based on the Use of High-Level Synthesis Tools. https://cryptography.gmu.edu/athena/presentations/GMU_DIAC_2016_HLS.pdf (2016)

14. Homsirikamol, E., Diehl, W., Ferozpuri, A., Farahmand,F., Yalla, P., Kaps, J.P., Gaj, K.: CAESAR HardwareAPI. Cryptology ePrint Archive, Report 2016/626 (2016)

15. Kumar, Sachin et al. "A Comprehensive Performance Analysis of Hardware Implementations of CAESAR Candidates." *IACR Cryptol. ePrint Arch.* 2017 (2017).

16. F. Farahmand, W. Diehl, A. Abdulgadir, J. Kaps and K. Gaj, "Improved Lightweight Implementations of CAESAR Authenticated Ciphers," Proceedings of the 26th IEEE International Symposium on Field- Programmable Custom Computing Machines,FCCM 2018, Boulder, CO, USA, Jun 2018.

17. E. Homsirikamol,W. Diehl, A. Ferozpuri, F. Farahmand, and K. Gaj, "Implementers Guide to Hardware Implementations Compliant with the CAESAR Hardware API, v1.0," https://cryptography.gmu.edu/athena/CAESAR_HW_API/CAESAR_HW_Impleenters_Guide_v1.0.pdf

18. E. Homsirikamol,W. Diehl, A. Ferozpuri, F. Farahmand, and K. Gaj "Implementers Guide to Hardware Implementations Compliant with the CAESAR Hardware API, v2.0," https://cryptography.gmu.edu/athena/CAESAR_HW_API/CAESAR_HW_Implementers_Guide_v2.0.pdf.

19. H. Gro, E. Wenger, C. Dobraunig, and C. Ehrenhofer, Suit up! Made-to-Measure Hardware Implementations of ASCON, in 2015 Euromicro Conference on Digital System Design, Aug 2015,

20. D. Engels, O. Saarinen, P. Schweitzer and E. Smith, "The Hummingbird-2 Lightweight Authenticated Encryption Algorithm," RFID. Security and Privacy: 7th International Workshop, RFIDSec '11, Amherst, Massachusetts, USA,June 2011.

21. A. Bogdanov, F. Mendel, F. Regazzoni, V. Rijmen, and E. Tischhauser, "ALE: AES-Based Lightweight Authenticated Encryption," 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013E. Homsirikamol, W. Diehl, A. Ferozpuri, F. Farahmand, and K.

22. P. Yalla and J. P. Kaps, "Evaluation of the CAESAR Hardware API forLightweight Implementations," 2017 International Conference on ReCon-Figurable Computing

23. F. Farahmand, W. Diehl, A. Abdulgadir, J. Kaps and K. Gaj, "Improved Lightweight Implementations of CAESAR Authenticated Ciphers," Proceedings of the 26th IEEE International Symposium on Field- Programmable Custom Computing Machines,FCCM 2018, Boulder, CO, USA, Jun 2018

24. J. Aumasson, P. Jovanovic, and S. Neves, "NORX," https://competitions.cr.yp.to/round3/norxv30.pdf.

25. I. Nikolic, “TIAOXIN,” https://competitions.cr.yp.to/round1/tiaoxinv1.pdf.

26. E. Andreevam, A. Bogdanov, N. Datta, A. Luykx, B. Mennink, M. Nandi, and E. Tischhauser, “CLOM,”https://competitions.cr.yp.to/round2/colm.pdf.

27. T. Iwata, K. Minematsu, J. Guo, S. Morioka, and E. Kobayashi, “CLOC and SILC,” https://competitions.cr.yp.to/round3/clocsilcv3.pdf.

28. Wu H, Huang T. “The JAMBU Lightweight Authentication Encryption Mode (v2.1)”, CAESAR competition proposal, 2016.

29. D. Bhattacharjee and A. Chattopadhyay, “Efficient Hardware Accelerator for AEGIS-128 Authenticated Encryption,” 10th International Conference, Inscrypt 2014, Beijing, China, December 13-15, 2014.

30. Canright, David “A Very Compact Rijndael S-box”, 2005.

31. J. Jean, I. Nikolić and T. Peyrin, JOLTIK v1.3 (2015), https://competitions.cr.yp.to/round2/joltikv13.pdf.

32. P. S. Barreto, H. Y. Kim and V. Rijmen, “Toward secure public-key blockwise fragile authentication watermarking”, IEE Proc., Vis. Image Signal Process. 149 (2002) 57–62.

33. Michael Fivez, ”Energy Efficient Hardware Implementations of CAESAR Submissions”. Diss. Master’s thesis, KU Leuven, 2016.

34. A. Abbas, H. Mostafa, and A. N. Mohieldin, “Low Area and Low Power Implementation for Caesar Authenticated Ciphers”, IEEE International Conference on Next Generation Circuits and Systems (NGCAS 2018), Malta, 2018.

35. A. Abbas, H. Mostafa, and A. N. Mohieldin, “Low Area and Low Power Implementation for Competition for Authenticated Encryption, Security, Applicability, and Robustness Authenticated Ciphers. Journal of Low Power Electronics”, 2019,. 15. 104-114. 10.1166/jolpe.2019.1593.

36. Wu H, Huang T. “The JAMBU Lightweight Authentication Encryption Mode (v2.1)”, CAESAR competition proposal, 2016.

37. Daemen, J., Rijmen, V. “The Design of Rijndael: AES - The Advanced Encryption Standard”. Information Security and Cryptography. Springer (2002), New York, United States.

38. Rogaway P, Bellare M, Black J. “OCB: A block-cipher mode of operation for efficient authenticated encryption”, ACM Transactions on Information and System Security (TISSEC), 2003, 6(3): 365-403

39. Minematsu K. “AES-OTR v3”, CAESAR competion proposal, 2016.

40. K. Papadimitriou, A. Anyfantis and A. Dollas, "An Effective Framework to Evaluate Dynamic Partial Reconfiguration in FPGA Systems," in *IEEE Transactions on Instrumentation and Measurement*, vol. 59, no. 6, pp. 1642-1651, June 2010.19

41. S. Liu, R. Pittman, A. Forin and J. Gaudiot, "Minimizing the runtime partial reconfiguration overheads in reconfigurable systems", The Journal of Supercomputing, vol. 61, no. 3, pp. 894-911, 2012.20

42. S. Bhandari *et al.*, "High Speed Dynamic Partial Reconfiguration for Real Time Multimedia Signal Processing," *2012 15th Euromicro Conference on Digital System Design*, Izmir, 2012, pp. 319-326.21
43. S. Di Carlo, P. Prinetto, P. Trotta and J. Andersson, "A portable open-source controller for safe Dynamic Partial Reconfiguration on Xilinx FPGAs," *2015 25th International Conference on Field Programmable Logic and Applications (FPL)*, London, 2015, pp. 1-4.22
44. Xilinx Inc. "AXI HWICAP PG134" v3.0, October 2016.40
45. A. Kamal, "Dynamic Partial Reconfiguration Techniques for Software Defined Radio Hardware Implementation", Master's thesis, Cairo University, 2017.
46. A. Kamaleldin *et al.*, "Design guidelines for the high-speed dynamic partial reconfiguration based software defined radio implementations on Xilinx Zynq FPGA," *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, Baltimore, MD, 2017, pp. 1-4, doi: 10.1109/ISCAS.2017.8050456.
47. Xilinx Inc., "Partial Reconfiguration Controller PG193," April 2016.
48. S. Bhandari et al., "High Speed Dynamic Partial Reconfiguration for Real Time Multimedia Signal Processing," 2012 15th Euromicro Conference on Digital System Design, Izmir, 2012, pp. 319-326.21
49. S. Di Carlo, P. Prinetto, P. Trotta and J. Andersson, "A portable open-source controller for safe Dynamic Partial Reconfiguration on Xilinx FPGAs," 2015 25th International Conference on Field Programmable Logic and Applications (FPL), London, 2015, pp. 1-4.22
50. M. Liu, W. Kuehn, Z. Lu and A. Jantsch, "Run-time Partial Reconfiguration speed investigation and architectural design space exploration," 2009 International Conference on Field Programmable Logic and Applications, Prague, 2009, pp. 498-502.23
51. K. Vipin and S. A. Fahmy, "A high speed open source controller for FPGA Partial Reconfiguration," Field-Programmable Technology (FPT), 2012 International Conference on, Seoul, 2012, pp. 61-66.24
52. K. Vipin and S. A. Fahmy, "ZyCAP: Efficient Partial Reconfiguration Management on the Xilinx Zynq," in IEEE Embedded Systems Letters, vol. 6, no. 3, pp. 41-44, Sept. 2014.2
53. C. Kohn "Partial Reconfiguration of a Hardware Accelerator on Zynq7000 All Programmable SoC Devices XAPP1159", Xilinx Inc. Application Notes, 2013.
54. M. Jahanbani, Z. Norozi and N. Bagheri, "DPA Protected Implementation of OCB and COLM Authenticated Ciphers," in *IEEE Access*, vol. 7, pp. 139815-139826, 2019, doi: 10.1109/ACCESS.2019.2942781.
55. W. Stallings, "The offset codebook (OCB) block cipher mode of operation for authenticated encryption", *Cryptologia*, vol. 42, no. 2, pp. 135-145, 2018.

56. P. Rogaway, M. Bellare, J. Black and T. Krovetz, "OCB: A block-cipher mode of operation for efficient authenticated encryption", *Proc. 8th Conf. Comput. Commun. Secur.*, pp. 196-205, 2001

57. T. Krovetz and P. Rogaway, The OCB Authenticated-Encryption Algorithm, 2014, [online] Available: https://tools.ietf.org/html/rfc7253.

58. Michael Fivez, “Energy Efficient Hardware Implementations of CAESAR Submissions”. Diss. Master’s thesis, KU Leuven, 2016.

Appendix A: List of Publications

1. A. Abbas, H. Mostafa, and A. N. Mohieldin, “Low Area and Low Power Implementation for Caesar Authenticated Ciphers”, IEEE International Conference on Next Generation Circuits and Systems (NGCAS 2018), Malta, 2018.
2. A. Abbas, H. Mostafa, and A. N. Mohieldin, “Low Area and Low Power Implementation for Competition for Authenticated Encryption, Security, Applicability, and Robustness Authenticated Ciphers. Journal of Low Power Electronics”, 2019. 15. 104-114. 10.1166/jolpe.2019.1593.

الملخص

يستخدم إنترنت الأشياء البيانات التي تم جمعها من أجهزة إنترنت الأشياء لتحسين المراقبة والسيطرة على العالم في مجالات مثل الخدمات اللوجستية ، وتجارة التجزئة والخدمات العسكرية ، والرعاية الصحية .أجهزة إنترنت الأشياء معرضة للهجوم ، نظرًا لأنها صغيرة جدًا ، وعادة ما تمتلك الزكاء الكافي لأداء وظيفة واحدة ، بحيث يمكنها لتحقيق أن تناسب أي مكان تقريبًا .في هذا البحث، نركز على أمان الأجهزة منخفضة الطاقة منخفضة الحجم ، وتم تحسين خمسة شفرات للتشفير من مسابقة (CAESAR) لتحقيق حجم اقل وطاقة اقل وهي JAMBU.و Tiaoxin و COLM و SILC و NORX .

علاوة على ذلك ، هناك منهجية تصميم جديدة تستخدم إعادة التكوين الديناميكي الجزئي لتبديل وظائف معالج التشفير بين التشفير وفك التشفير والتي تحقق أمانًا في الأجهزة يتميز بكفاءة الموارد.

مهندس:	عمرو محمد عباس دسوقى
تاريخ الميلاد:	6/4/1990
الجنسية:	مصرى
تاريخ التسجيل:	1/10/2013
تاريخ المنح:	
القسم:	هندسة الإلكترونيات والإتصالات الكهربية
الدرجة:	ماجستير العلوم
المشرفون:	د. أحمد نادر محي الدين
	د. حسن مصطفى حسن
الممتحنون:	

عنوان الرسالة:

أمان أجهزة انترنت الأشياء

الكلمات الدالة:

إعادة التشكيل الجزئي الديناميكى انترنت الأشياء ، مصفوفات البوابات المنطقية القابلة للبرمجة.

ملخص الرسالة:

أمان أجهزة انترنت الأشياء

اعداد

عمرو محمد عباس دسوقى

رسالة مقدمة إلى كلية الهندسة - جامعة القاهرة
كجزء من متطلبات الحصول على درجة **ماجستيرالعلوم**
في
هندسة الإلكترونيات والإتصالات الكهربية

يعتمد من لجنة الممتحنين:

الأستاذ الدكتور:	المشرف الرئيسى
الأستاذ الدكتور:	مشرف
الأستاذ الدكتور:	الممتحن الداخلي
الأستاذ الدكتور:	الممتحن الخارجي

كليــة الهندســة - جامعــة القاهــرة
الجيـزة - جمهوريـة مصـر العربيــة
2020

أمان أجهزة انترنت الأشياء

اعداد

عمرو محمد عباس دسوقى

رسالة مقدمة إلى كلية الهندسة - جامعة القاهرة
كجزء من متطلبات الحصول على درجة **ماجستيرالعلوم**
في
هندسة الالكترونيات والاتصالات الكهربية

تحت اشراف

أ.د. أحمد نادر محي الدين

أستاذ دكتورمساعد
بقسم الإلكترونيات والإتصالات الكهربية
بكلية الهندسة جامعة القاهرة

أ.د. حسن مصطفى حسن

أستاذ دكتور مساعد
بقسم الإلكترونيات والإتصالات الكهربية
بكلية الهندسة جامعة القاهرة

كليــة الهندســة - جامعــة القاهــرة
الجيـزة - جمهوريـة مصـر العربيــة
2020

أمان أجهزة انترنت الأشياء

اعداد

عمرو محمد عباس دسوقى

رسالة مقدمة إلى كلية الهندسة - جامعة القاهرة
كجزء من متطلبات الحصول على درجة **ماجستير العلوم**
في
هندسة الالكترونيات والاتصالات الكهربية

كليــة الهندســـة - جامعـــة القاهـــرة
الجيـزة - جمهوريـة مصـر العربيـــة
2020