



Aalborg Universitet

**AALBORG UNIVERSITY**  
DENMARK

## **A Novel Single Pass Authenticated Encryption Stream Cipher for Software Defined Radios**

Khajuria, Samant

*Publication date:*  
2012

*Document Version*  
Early version, also known as pre-print

[Link to publication from Aalborg University](#)

*Citation for published version (APA):*  
Khajuria, S. (2012). *A Novel Single Pass Authenticated Encryption Stream Cipher for Software Defined Radios*.

### **General rights**

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal -

### **Take down policy**

If you believe that this document breaches copyright please contact us at [vbn@aub.aau.dk](mailto:vbn@aub.aau.dk) providing details, and we will remove access to the work immediately and investigate your claim.

# A Novel Single Pass Authenticated Encryption Stream Cipher for Software Defined Radios

---

A DISSERTATION  
SUBMITTED TO THE DEPARTMENT OF  
ELECTRONIC SYSTEMS  
OF  
AALBORG UNIVERSITY  
IN PARTIAL FULFILMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF  
DOCTOR OF PHILOSOPHY

Samant Khajuria  
Center for Wireless Systems and Applications  
Center for TeleInfrastruktur – Copenhagen  
Dept. of Electronic Systems  
AAU

**Supervisor:**

Professor Ramjee Prasad, CTIF, Aalborg University, Denmark

Professor Birger Andersen, CWSA-CTIF, Copenhagen University College of Engineering,  
Denmark

**The examination committee:****Moderator:****Date of defense:**

**ISSN:** \*\*\* \_ \*\*\*\*

**ISBN:** \*\* \_ \*\*\*\* \_ \*\*\* \_ \*

**Copyright © 2012 by Samant Khajuria**

All rights reserved. No part of the material protected by this copyright notice may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage and retrieval system, without written permission from the author.

# Abstract

With the rapid growth of new wireless communication standards, a solution that is able to provide a seamless shift between existing wireless protocols and high flexibility as well as capacity is crucial. Software Defined Radio (SDR) technology offers this flexibility. It gives the possibility of adapting the radio to users' preferences and the operating environment and supporting multiple standards without requiring separate hardware for each standard. In order to avail this enabling technology that is applicable across a wide range of areas within the wireless infrastructure, these radios have to propose cryptographic services such as confidentiality, integrity and authentication. Therefore, integration of security services into SDR devices is essential.

Authenticated Encryption schemes donate the class of cryptographic algorithms that are designed for protecting both message confidentiality and its authenticity. Traditionally, authenticated encryption was achieved by using two independent algorithms for encryption and authentication. For past few years, new modes of operation of block cipher have been developed that allow us to use one algorithm for encryption as well as authentication. This makes authenticated encryption very attractive for low-cost low-power hardware implementations, as it allows for the substantial decrease in the circuit area and power consumed compared to the traditional schemes.

In this thesis, an authenticated encryption scheme is proposed with the focus of achieving high throughput and low overhead for SDRs. The thesis is divided into two research topics. One topic is the design of a 1-pass authenticated encryption scheme that can accomplish both message secrecy and authenticity in a single cryptographic primitive. The other topic is the implementation of this design on re-configurable hardware in SDRs by closely observing the trade-off between area/throughput performance parameters.

For test and performance evaluation the design has been implemented in Xilinx Spartan – 3 sxc3s700an FPGA. The resulting implementation consumes moderate number of slices on FPGA and achieves throughput in the range of 0.8 Gbps which can be suitably used for SDR applications. Comparing with traditional two pass approaches, the presented design demonstrates high throughput and small area to performance ratio.



# Dansk Resume

Med den hurtige vækst i nye trådløse kommunikationsstandarder, er en løsning, der er i stand til at levere et problemfrit skifte mellem eksisterende trådløse protokoller og høj fleksibilitet samt kapacitet afgørende. Software Defined Radio (SDR) teknologi tilbyder denne fleksibilitet. Det giver mulighed for at tilpasse radioen til brugernes præferencer og driftsmiljøet og understøtte flere standarder uden at kræve separat hardware for hver standard. For at kunne benytte denne teknologi, der kan anvendes på tværs af en lang række områder indenfor trådløse infrastruktur, er disse radioer nødt til at tilbyde kryptografiske tjenester såsom fortrolighed, integritet og autentificering. Derfor er integration af sikkerhedstjenester i SDR-enheder af afgørende betydning.

Autentificeret krypteringsmetoder tilhører klassen af kryptografiske algoritmer, der er designet til at beskytte både besked fortrolighed og dens ægthed (autensitet). Traditionelt blev autentificeret kryptering opnået ved hjælp af to uafhængige algoritmer til kryptering og autentificering. De sidste få år er nye former for blokalgoritmer blevet udviklet, som giver os mulighed for at bruge samme algoritme til kryptering samt autentificering. Dette gør autentificeret kryptering meget attraktivt for billige energibesparende hardware implementeringer, da det giver mulighed for betydelig reduktion af kredsløbets omfang og strømforbruget sammenlignet med traditionelle metoder.

I denne afhandling er foreslået en autentificeret krypteringsmetode med fokus på at opnå høj kapacitet og lavt overhead for SDR. Afhandlingen er opdelt i to forskningsområder. Det ene er udformningen af en 1-passage autentificeret krypteringsmetode, der kan benyttes til både hemmeligholdelse og autenticitet vha. et enkelt kryptografisk primitiv. Det andet område er implementering af dette design på rekonfigurerbar hardware i SDR ved nøje at observere trade-off mellem kredsløbets omfang og kapacitets præstationsparametre.

For test og evaluering er designet blevet implementeret i Xilinx Spartan - 3 sxc3s700an FPGA. Den resulterende implementering forbruger moderat antal slices på FPGA og opnår kapacitet i området ca. 0,8 Gbps, som kan være passende for SDR applikationer. I sammenligning med traditionelle to passage metoder demonstrerer det præsenterede design høj kapacitet og lille kredsløbsomfang som ydelsesforhold.



# Acknowledgements

I would like to take the opportunity to thank the people who supported and accompanied me during my PhD studies.

First and foremost I would like to thank my supervisors Prof. Ramjee Prasad and Prof. Birger Andersen for all the support and feedback, without which I would not have been able to complete my research work and write my thesis. I would also like to thank Prof. Rajarathnam Chandramouli and Goce Jakimoski for giving me the opportunity to work under their guidance in which they gave me their expertise and timely advices during my research at Stevens Institute of Technology.

I would like to thank Center for Wireless Systems and Applications (CWSA) for sponsoring first year of my research. Next, I would like to thank my colleagues, the current once and the ones that have already gone from CWSA. I have always appreciated their precious feedback and long discussions with them about my research. I would also like to thank John Kryger Sørensen for his support and guidance.

I would also like to thank Susanne Nørrevang and Inga Hauge for their help during my travels.

Last, I would like to thank my friends and family for all their unconditionally support during my PhD

Thank you all





# Table of Contents

<b>List of Figures .....</b>	<b>xii</b>
<b>List of Tables .....</b>	<b>xiv</b>
<b>List of Acronyms .....</b>	<b>xv</b>

<b>1 Introduction .....</b>	<b>1</b>
<b>1.1 Software Defined Radio .....</b>	<b>1</b>
<b>1.1.1 Cognitive Radio .....</b>	<b>3</b>
<b>1.1.2 Field Programmable Gate Arrays / System on Chip .....</b>	<b>5</b>
<b>1.2 Need for Security .....</b>	<b>5</b>
<b>1.2.1 FPGAs for Cryptographic Application .....</b>	<b>6</b>
<b>1.2.2 Attacks on FPGA .....</b>	<b>8</b>
<b>1.3 Confidentiality and Authenticity .....</b>	<b>10</b>
<b>1.4 Motivation .....</b>	<b>11</b>
<b>1.5 Problem Definition .....</b>	<b>12</b>
<b>1.6 Limitations .....</b>	<b>13</b>
<b>1.7 Contribution .....</b>	<b>13</b>
<b>1.8 Organization .....</b>	<b>14</b>
<b>References .....</b>	<b>17</b>

## Part I – Algorithm Analysis & Development

<b>2 Cryptography .....</b>	<b>21</b>
<b>2.1 Symmetric Encryption .....</b>	<b>22</b>
<b>2.2 Block Ciphers .....</b>	<b>24</b>
<b>2.2.1 Advanced Encryption Standard .....</b>	<b>28</b>
<b>2.3 Stream Ciphers .....</b>	<b>31</b>
<b>2.3.1 Synchronous Stream Cipher .....</b>	<b>33</b>
<b>2.3.2 Self-Synchronizing Stream Cipher .....</b>	<b>34</b>

<b>2.4 Authentication</b>	<b>34</b>
<b>2.4.1 Cryptographic Hash Functions</b>	<b>35</b>
<b>2.4.2 Message Authentication Codes (MAC)</b>	<b>37</b>
<b>2.4.3 Dedicated Hash functions</b>	<b>38</b>
<b>2.4.4 Secure Hash Algorithm</b>	<b>40</b>
<b>2.4.5 Keyed-Hash Message Authentication Code (HMAC)</b>	<b>42</b>
<b>2.4.6 CBC-MAC</b>	<b>44</b>
<b>2.5 Conclusions</b>	<b>46</b>
<b>References</b>	<b>47</b>

### **3 Authenticated Encryption .....51**

<b>3.1 Generic Composition</b>	<b>52</b>
<b>3.1.1 Encrypt – and – MAC</b>	<b>53</b>
<b>3.1.2 MAC – then – Encrypt</b>	<b>54</b>
<b>3.1.3 Encrypt – then – MAC</b>	<b>55</b>
<b>3.2 Two Pass Combined mode</b>	<b>56</b>
<b>3.2.1 CCM Mode</b>	<b>57</b>
<b>3.2.2 EAX Mode</b>	<b>58</b>
<b>3.3 Single Pass Combined Mode</b>	<b>59</b>
<b>3.3.1 IAPM</b>	<b>60</b>
<b>3.3.2 XCBC</b>	<b>61</b>
<b>3.3.3 OCB</b>	<b>62</b>
<b>3.4 AE Stream Ciphers</b>	<b>63</b>
<b>3.4.1 Helix</b>	<b>63</b>
<b>3.4.2 Sober-128</b>	<b>64</b>
<b>3.5 ASC-1 : An Authenticated encryption Stream Cipher</b>	<b>65</b>
<b>3.5.1 LEX Stream Cipher</b>	<b>65</b>
<b>3.5.2 ASC-1 Specification</b>	<b>68</b>
<b>3.6 Security Considerations</b>	<b>71</b>
<b>3.6.1 Security Measurements</b>	<b>71</b>
<b>3.7 Preliminaries</b>	<b>74</b>
<b>3.7.1 Classical Attacks of Cheating</b>	<b>77</b>
<b>3.8 Security in ASC-1</b>	<b>78</b>
<b>3.8.1 The Information-Theoretic Case</b>	<b>79</b>
<b>3.8.2 Computational Security Analysis of ASC-1</b>	<b>85</b>
<b>3.9 Conclusions</b>	<b>87</b>
<b>References</b>	<b>88</b>

## Part II – Implementation & Results

<b>4</b>	<b>Field programmable Gate Arrays (FPGAs)</b>	<b>95</b>
4.1	FPGA Architecture	96
4.1.1	FPGA Implementation Flow	97
4.1.2	Xilinx Spartan 3AN	98
4.2	Role of FPGAs in SDR	100
4.2.1	Cross Layer Architecture of SDR	101
4.2.2	GNU Radio and USRP	103
4.3	Generic SDR Structure	106
4.3.1	SOC in SDR	108
4.3.2	Secure Communication	109
4.4	Implementation of ASC-1	110
4.4.1	Initial phase Generation	111
4.4.2	Encryption Process	111
4.4.3	Proposed ASC-1 Architecture	113
4.5	Conclusions	118
	References	119
<b>5</b>	<b>Hardware Implementation &amp; Results</b>	<b>121</b>
5.1	Parameters of Hardware Implementation	122
5.2	Block Cipher Modes of Operation	122
5.2.1	Hardware Architecture for Feedback cipher modes	122
5.2.2	Hardware Architecture in Non-feedback cipher mode	124
5.3	Performance of ASC-1 Crypto core	127
5.4	Frame Delay	133
5.5	Payload Length on Effective Throughput	135
5.6	LTE and WiMAX	137
5.6.1	OFDM in LTE and WiMAX	138
5.6.2	Confidentiality and Integrity in LTE and WiMAX	139
5.7	Conclusions	142
	References	143
<b>6</b>	<b>Summary &amp; Future Scope</b>	<b>147</b>
6.1	Contribution of This Thesis	147
6.2	Open Problems	149

References .....150

**Appendix I - Test Vectors .....153**

A.1 ASC-1 Preprocessing/ Initial Phase .....153  
A.2 Key Expansion .....154  
A.3 ASC-1 Encryption .....155

**Appendix II - Xilinx Sample Code and Waveforms .....163**

B.1 Advanced Encryption Standard (AES) – 128 .....163  
B.2 Key Expansion .....169  
B.3 ASC-1 Encryption .....173

# List of Figures

1.1 Generic SDR transceiver .....	2
1.2 Thesis Outline .....	15
2.1 Electronic Codebook .....	25
2.2 Cipher Block Chaining .....	26
2.3 Cipher Feedback .....	27
2.4 Output Feedback .....	28
2.5 128 bits Advanced Encryption Standard .....	31
2.6 Stream Cipher .....	32
2.7 Synchronous Stream Cipher .....	33
2.8 Self-Synchronizing Stream Cipher .....	34
2.9 Classification of cryptographic hash functions .....	36
2.10 HMAC Construction .....	43
2.11 CBC-based MAC algorithm .....	44
3.1 EAX mode .....	59
3.2 Integrity Aware Parallelizable Mode scheme .....	61
3.3 Extended Cipher Mode Chaining Encryption mode scheme .....	62
3.4 Offset CodeBook mode scheme .....	63
3.5 LEX Stream Cipher .....	66
3.6 Leak Positions in odd and even rounds .....	66
3.7 The Encryption and decryption algorithms of ASC-1 .....	69
3.8 The 4R-AES transformation .....	70
3.9 An authenticated encryption scheme construction based on a LAXU hash function family on a CFB-like mode .....	76
3.10 A two round SPN structure with a leak .....	79
3.11 A composition of a transformation .....	82
3.12 The first $s$ output of a non-linear function $F$ .....	83
4.1 Programmable-Logic-Arrays .....	95

4.2 Field Programmable Gate Array .....	96
4.3Xilinx implementation Flow .....	98
4.4 Xilinx Spartan 3AN .....	99
4.5 Software Defined Radio .....	100
4.6 SDR Layers in OSI .....	102
4.7 Block diagram of GNU Radio Components .....	104
4.8 Block Diagram of the USRP .....	105
4.9 Practical SDR receiver .....	107
4.10 SCA Structure, showing the Red and Black FPGAs .....	109
4.11 Frame Structure .....	110
4.12 Initialization vector and Key Generation .....	111
4.13 ASC-1 Flowchart .....	113
4.14 Block diagram of ASC-1.....	114
4.15 Fully parallel pipelined structure .....	116
4.16 AES-128 & ASC-1 encryption core block .....	117
5.1 Iterative Architecture of Block cipher .....	123
5.2 Hardware Architecture .....	124
5.3 Hardware architecture for non-feedback cipher modes .....	125
5.4 Optimal Hardware Architecture for non-feedback cipher modes .....	126
5.5 S-Box organized as 8 banks of 256 x 8 dual port ROMs .....	128
5.6 Key Logic Unit .....	129
5.7 Proposed AES-128 Hardware Architectures .....	131
5.8 Proposed ASC-1 Iterative Hardware Architectures .....	133
5.9 Crypto Architecture .....	134
5.10 Throughput vs. Payload length at a bit error rate of $10^{-4}$ in a channel .....	135
5.11 Throughput vs. Payload length at a bit error rate of $10^{-5}$ in a channel .....	136
5.12 Throughput vs. Payload length at a bit error rate of $10^{-4}$ in a channel with no latency and with latency of 248ns .....	137
5.13 Confidentiality Algorithm $f_8$ .....	140
5.14 Integrity Algorithm $f_9$ .....	141





# List of Tables

2.1 Hash Functions and MAC Algorithms.....	45
3.1 Security Results in different composite authenticated encryption schemes .....	56
5.1 Performance of Basic Operations in AES block cipher .....	128
5.2 Performance of AES-128 & AES-256 Key Expansion in Parallel and Iterative Architecture .....	130
5.3 Performance of AES-128 Encryption in Parallel and Iterative Architecture .....	130
5.4 Performance of ASC-1 Encryption core Iterative Architecture .....	132
5.5 Physical Layer Parameter for LTE and WiMAX .....	139

# List of Acronyms

<b>A2D</b>	Analog-to-Digital
<b>ASIC</b>	Application Specific Integrated Circuits
<b>AES</b>	Advanced Encryption Standard
<b>ASC-1</b>	Authentication Stream Cipher One
<b>AEAD</b>	Authenticated encryption with associated-data
<b>AE</b>	Authenticated encryption
<b>AH</b>	Authentication header
<b>AU</b>	Almost Universal hash function
<b>AXU</b>	Almost XOR Universal hash function
<b>BER</b>	Bit Error Rate
<b>CR</b>	Cognitive radio
<b>CFB</b>	Cipher feedback mode
<b>CA</b>	Certificate Authority
<b>CBC</b>	Cipher Block Chaining
<b>CFB</b>	Cipher Feedback
<b>CRHF</b>	Collision Resistant Hash Functions
<b>CPLD</b>	Complex Programmable Logic Device
<b>DSP</b>	Digital Signal Processor
<b>DSA</b>	Dynamic Spectrum Access
<b>DES</b>	Data Encryption Standard
<b>DoS</b>	Denial of Service
<b>DPA</b>	Differential Power Analysis
<b>DDC</b>	Digital Down-convertors
<b>DoD</b>	Department of Defense
<b>D2A</b>	Digital-to-Analog
<b>ECB</b>	Electronic Code Book
<b>ESP</b>	Encapsulating security payload
<b>FAS</b>	Frame Alignment Signal
<b>FPGA</b>	Field Programmable Gate Array
<b>HDL</b>	Hardware Description Language
<b>IF</b>	Intermediate Frequency

<b>INT-PTXT</b>	Integrity of the plaintexts
<b>INT-CTXT</b>	Integrity of the ciphertexts
<b>IND-CPA</b>	Indistinguishability under a chosen plaintext attack
<b>IND-CCA</b>	Indistinguishability under a chosen ciphertext attack
<b>IACBC</b>	Integrity Aware CBC
<b>IAPM</b>	Integrity Aware Parallelizable Mode
<b>JTRS</b>	Joint Tactical Radio Software Program
<b>LAXU</b>	Leak-safe XOR Universal hash function
<b>MAC</b>	Message Authentication Codes
<b>MDC</b>	Modification Detection Codes
<b>NIST</b>	National Institute of Standards and Technology
<b>NM-CPA</b>	Non-malleability under a chosen plaintext attack
<b>NM-CCA</b>	Non-malleability under chosen ciphertext attack
<b>OFB</b>	Output Feedback
<b>OWHF</b>	One-Way Hash Functions
<b>OCB</b>	Offset CodeBook mode
<b>PKC</b>	Public-Key Cryptography
<b>PROM</b>	Programmable Read-Only Memory
<b>PLA</b>	Programmable logic array
<b>QoS</b>	Quality of Service
<b>RF</b>	Radio Frequency
<b>RTL</b>	Register transfer level
<b>SDR</b>	Software Defined Radio
<b>SHS</b>	Secure Standard
<b>SHA</b>	Secure Hash Algorithm
<b>SCA</b>	Software Communication Architecture
<b>SS</b>	Spectrum Sensing
<b>SD</b>	Spectrum Decision
<b>SM</b>	Spectrum Mobility
<b>SSH</b>	Spectrum Sharing
<b>SoC</b>	System-on-Chip
<b>SPA</b>	Simple Power Analysis
<b>SSL</b>	Secure Socket Layer
<b>XCBC</b>	Extended Cipher Block Chaining Encryption





# 1

## Introduction

Software-defined radio (SDR) has been recognized as one of the most important technologies for wireless communications. It offers a flexible mechanism to change transmitter and receiver characteristics such as modulation type, radiated power, and air interfaces by making software changes. This gives rise to the possibility of adapting the radio to users' preferences and the operating environment and of supporting multiple standards without requiring separate hardware for each standard. However lots of security concerns are raised in terms of reconfiguring the radio through software (downloading new radio functionality), platform integrity, key management, spectrum management, and integrity and confidentiality of data.

The concept of SDR appeared in 1970's in USA and Europe and the term was coined by Joseph Mitola III in 1991[JMI1]. The idea was to signal the shift from digital radio to multiband multimode software-defined radios where 80% of the functionality is provided in software. Mitola presented these radios as an intelligent agent able to track radio resources and related computer-to-computer communications and able to detect user communications needs as a function of use context, and to provide radio resources and the wireless services most appropriate to those needs. In 1992 United States Department of Defense (DoD) began the development of SDR technology through the SPEAKeasy research project, where the idea was to bring all the discrete military radios under one umbrella using software radio technology [RJD]. In 1997, the United State government launched the Joint Tactical Radio Software Program (JTRS) with the mission to develop standardized software architecture in order to improve software component portability, know as *Software Communication Architecture (SCA)* [JTR].

Initially SCA was mainly developed for military purposes. In 1999 SDR Forum adopted the SCA standard and promoted SDR technologies with applications for commercial cellular, Personal Communication Systems (PCS), and third generation (3G) and emerging fourth generation (4G) cellular services.

### 1.1 Software Defined Radio

The typical hardware radio system consists of a variety of analogue elements like filters, converters, modulators and demodulators. These hardware devices are expensive and offer low compatibility with other components. This prompted the idea of SDR, where the user could use

SDR technology to realize many applications without a lot of efforts in integration of different components by moving from analogue to digital technology. Different software modules can be implemented to support different modulators and demodulators in the SDR platform. Also with the exponential increase in digital technology in terms of performance and productivity will continue to move closer to the antenna and replace much of the analogue front end.

Figure 1.1 shows a generic SDR transceiver, where the main components are the radio front-end and baseband processor. The front-end part consists of analog hardware modules. As the receiver side, the RF front-end part is responsible for RF amplification, down conversion from radio frequency (RF) to intermediate frequency (IF) signal and finally convert the signal from Analog-to-Digital (A2D). On the transmit path, the signal is converted from Digital-to-Analog (D2A) then analog up conversion and RF power amplification. The baseband processing unit performs the baseband operations, these operations are quite different based on the type of communication technologies. However based on their role, it can be categorized into five function blocks: Channel coding/decoding, block interleaving/ de-interleaving, modulation/ demodulation, channel estimation, and pulse shaping [TUL].

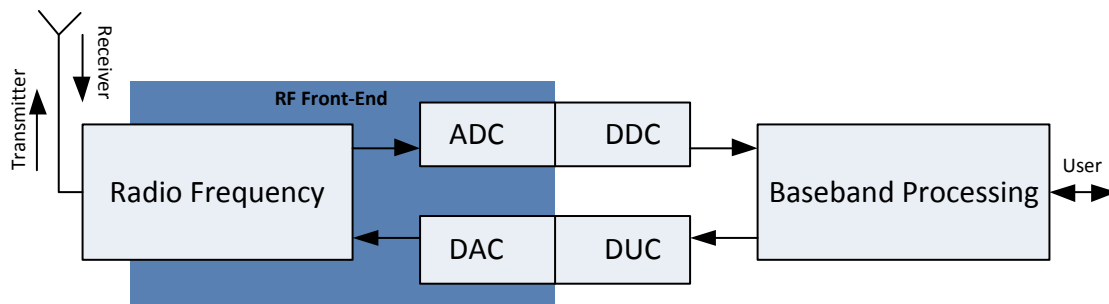


Figure 1.1: Generic SDR transceiver

The main feature of SDR is its ability to dynamically adapt according to the radio environment through the re-configurability of its components. More precisely, the re-configurability is the ability of adjusting operational parameters for the transmission on-the-fly without any modifications on the hardware components. This feature gives SDR systems the ability to support a variety of mobile radio standards. Unlike implementing radio functional blocks on inflexible Application Specific Integrated Circuits (ASIC) in the past, the technologies such as Field programmable Gate Array (FPGA), Digital Signal Processor (DSP) and General-Purpose Processor (GPP) are used to build software radio blocks. These components have reconfigurable capability and deliver flexibility of programmable architectures with power efficiency and performance. In

[TUL] one identifies four different opportunities for reconfiguration in software-defined radio:

- *Pre-deployment* – Late changes are made in the design process based on the target architecture just before the device is deployed.
- *In-field upgrade* – Device software / firmware is updated to support a new standard or feature that was not included at deployment.
- *Reconfiguration per call or session* – Device is reconfigured at the start of the session i.e., a voice or data transmission. This could be to select the most efficient or cheapest service available at the point of time.
- *Reconfiguration during a call or session* – Device is reconfigured during voice or data transmission e.g. to hand over from one service to another.

### 1.1.1 Cognitive Radio

The idea of cognitive radio (CR) was first described by Mitola [JMI2], as the natural evolution of the SDR. The CR by Mitola was presented as an intelligent fully reconfigurable wireless transceiver that is able to detect user communication needs and automatically adapts its communication parameters most appropriate to those needs. Later the focus of the research was shifted towards the intelligent and opportunistic use of the radio resources, a technique known as Dynamic Spectrum Access (DSA).

Based on this, the concept of CR was more emphasized towards DSA, which led to a new CR definition[SHY]: *an intelligent wireless communication system, aware of its surrounding environment that uses the methodology of understanding-by-building to learn from the environment and to adapt its internal states to statistical variations in the incoming RF stimuli by making corresponding changes in certain operating parameters (e.g., transmit-power, carrier-frequency, and modulation strategy) in real-time, with two primary objectives in mind: highly reliable communications whenever and wherever needed and efficient use of the radio spectrum.*

One of the main capabilities of the CR is its ability to reconfigure, which is enabled by the SDR platform, upon which the CR is built. The key enabling technologies of CR are the functions that provide the capabilities to share the spectrum in an opportunistic manner. Most of the spectrum is already assigned but not used all the time. Here the challenge is the optimal sharing of the



spectrum with other existing networks without interfering with their transmission. In order to do so the CRs makes uses of temporarily unused spectrum known as Spectrum Holes or white spaces. By definition in [SHY] a spectrum hole is a band of frequencies assigned to a primary user, but at a particular time and specific geographic location, the band is not being used by that user. However if CR is using a particular spectrum hole and another user also starts using this space then the CR either move to another spectrum hole or continue using it by altering its transmission power level or modulation scheme to minimize interference. Therefore CR is a perfect example for using the available spectrum in an efficient and opportunistic manner and through it, it is possible to:

- Sense the available spectrum;
- Decide the best available channel;
- Coordinate with other users;
- Vacate the channel either due to primary user or when the channel conditions worsen.

**Cognitive Radio Functions** - The main goal of CR is to enable networks to use the available spectrum band according to network users Quality of Service (QoS) requirements. However CR networks impose unique challenges due to the presence of primary network. Thus new spectrum management functions for CR networks are taken into consideration in order to avoid critical design challenges like – Interface avoidance, QoS awareness and Seamless communication. These functions are as follows:

- *Spectrum sensing (SS)*: The purpose of SS is to allocate only the unused portions of the spectrum. The radio monitors the available spectrum bands and look for spectrum holes.
- *Spectrum decision (SD)*: Based on the available spectrum bands and associated potential channel estimated channel capacity, the CR user can allocate the channel. The decision is not only based on the availability of the channel but also QoS requirements such as data rate, error rate, mode of transmission, bandwidth etc.
- *Spectrum mobility (SM)*: SM is an important function which allows a network to use the spectrum in a dynamic manner. If the specific portion of the spectrum in use is required by the primary user, the communication must be continued in another vacant portion of the spectrum.
- *Spectrum Sharing (SSH)*: Due to the shared nature of the wireless channel there may be multiple CR users trying to access the spectrum, access to the network should be coordinated in order to prevent multiple users colliding in overlapping portions of the spectrum.

### 1.1.2 Field Programmable Gate Arrays/ System on Chip

For past few years FPGAs have become an increasingly important resource for SDR. A successful SDR design is build by designing powerful processing blocks and connecting these blocks to form a flowgraph. These blocks are either implemented in software or in soft hardware with an FPGA. However with the development in the field of programmable logic, FPGA has ushered in an era of rapid prototyping for digital systems. The FPGAs are cost efficient and supplies abundant resources too for the system designers. FPGAs are programmed using Hardware Description Language (HDL) commonly known as VHDL. Same VHDL code could run on any FPGA that has enough logic cells, this characteristic give further flexibility and dynamic upgradeability to SDR device. FPGAs reprogrammable nature makes it ideal for SDRs, so any upgrades or changes in the operational parameters can be easily uploaded to the device without any hardware reconfigurations.

***Partial Reconfiguration*** – A shared resources model is referred as more efficient architecture for SDRs. As compared to dedicated resource model, shared resources are capable of supporting ex., multiple waveforms across a single set of processing resources; this allows for much more efficient usage of the resources. The technology that facilitates this model is partial reconfiguration of the FGPA. Partial reconfiguration allows the replacement of one or multiple functional blocks with a different implementation while other portions are either being used by other applications or going unused. Without partial reconfiguration, it would be necessary to reconfigure entire FPGA. However, using partially reconfigurable platform FPGAs as an SDR system-on-chip (SoC) will substantially decrease the component count of the SDR device and reduce power consumption while still providing the necessary functionality [PKG].

## 1.2 Need for Security

Communication between two or more devices over insecure channel, flexibility of implementing radio functions such as modulation/demodulation, signal generation etc, on software and reconfiguration of radios to upgrade or adapt to user preferences, and regional regulations may lead to serious radio security concerns. While reconfiguring the radios have many benefits, the ability to reconfigure radio functionalities with software may lead to many security problems such as unauthorized use of application and network services, unauthorized modification of software and manipulation of radio sets. For example, malicious software can be uploaded into the device that changes its radio frequency so that the device will no longer function within the regulated

constraints. This could lead to the Denial of Service (DoS) attacks. Additionally, transmission of unencrypted data over insecure channel could compromise the confidentiality and integrity of the data.

The concept of transmission security is broadly divided at three different levels of security [TAJ], as follows:

- *Detection* – If an unauthorized user is able to distinguish between transmission signal and noise, a signal is said to be detected.
- *Interception* – After transmission signal being detected if an unauthorized user is able to identify the specific class of a signal and distinguish from signals belonging to other classes, a signal is said to be intercepted.
- *Exploitation* – If an unauthorized user is able to recover any useful information such as the message contents or the origin of the message, then a signal is said to be exploited.

Security has always being a hot topic in SDRs weather it is for military or commercial applications. However the subject of security for SDR systems is quite broad and covers many issues. In the past SDR security has followed few general directions ex., the first direction mentioned in [SS1][SS2][HRM] covers the list of security requirements for the underlying hardware, integrity of the platform, downloading upgrades, key management issues, and content protection and possible threats on SDR. The second issue explores the essential issue of secure downloading of new radios parameters. A framework for establishing secure download using a tamper-proof hardware module is proposed [RFL][LMI][ABR]. The third direction focuses on spectrum management and policy enforcement of SDR [PFL][KSA][ATO].

### **1.2.1 FPGAs for Cryptographic Application**

FPGAs are generic semiconductor devices contain programmable logic components called “logic blocks” that can be programmed and reprogrammed, as per user-defined logic functions. In recent years, FPGAs are designed using the latest technologies to be as competitive as possible with ASICs in terms of performance, power and space. FPGAs compete by being reconfigurable and combine the advantages of software and hardware implementation. Additionally, FPGA reconfigurable hardware gives advantage in cryptographic applications [ACR][TWC]. Following shows the benefits of implementing Crypto solutions on FPGAs:

*Algorithm Flexibility* - The term flexibility in algorithms refers to the switching of cryptographic

algorithms during operation of the targeted application. Based on the degree / level of the security various encryption algorithms could be programmed or reprogrammed on the fly. Majority of security protocols such as IPSec are algorithm independent and allow for multiple encryption algorithms. These algorithms are negotiated on a per-session basis and wide variety may be required.

*Algorithm Upload* - It is to be recognized that the devices need to be upgraded at some point with newer or securer cryptosystem. The reason could be compatibility concerns with new applications. Algorithm upload is necessary because a current algorithm for ex., AES (Advanced Encryption Standard) uses only 128-bit encryption keys and to increase the level of security it needs to be replaced by AES 256-bit. This could be achieved under the assumption that there is some kind of connection to a network, where FPGA-equipped encryption devices can upload the new configuration code. As compared to FPGA-implemented this upgrade is practically infeasible to ASIC-implemented algorithms.

*Architecture Modification* - Most of the encryption algorithms used today are standardized. But in certain cases hardware architecture can be much more efficient if it is designed for specific set of performance parameters like implementation area, throughput and latency. Based on the type of cryptosystem and resources, and area available these algorithms could be implemented on sequential or parallel hardware architectures. For example as shown in chapter 5 section 5.3, hardware implementation of AES in fully parallel hardware architecture can achieve maximum throughput of 32 Gbits/s on Xilinx Vertix -5VLX50T and 51% of area is consumed. Whereas sequential architecture uses 20% of area and can achieve maximum throughput of 0.876 Gbits/s.

*Resource Efficiency* - Today most of the security protocols used for communication between two or more devices are hybrid protocols. This means, that a public-key algorithm is used for secure key exchange and private key algorithm for encryption of data over channel. Once the keys are exchanged and devices are authenticated, the public-key algorithm is not used for the session. Since the algorithms are not used simultaneously, the same FPGA device can be utilized for both through run-time reconfiguration. This is an important factor in many implementations where resources are limited.

*Throughput* - Comparing FPGAs to general purpose CPU and ASICs, FPGA implementations have the potential of running substantially faster than software implementations but slower than ASIC implementation. General purpose CPUs are not optimized for fast execution especially in the case of public-key algorithms. That happens mainly because they lack those instructions for modular arithmetic operations on long operands. AES block cipher for example reaches the data rate of 112.3 Mbit/s and 718.4 Mbit/s on a DSP TI TMS320C6201 and Pentium III, respectively [ACR]. As compared to FPGA implementation of AES on Virtex XCV-1000BG560-6 achieved 12 Gbit/s using 12,600 slices and ASIC, the Amphenon CS5240TK can reach 25.6 Gbit/s at 200 Mhz [TWC].

### 1.2.2 Attacks on FPGA

For past few years, the industrial market is more inclined towards FPGAs due to their benefits of re-programmability. With FPGAs being used in a variety of military and commercial applications that require security features and as these designs have become more valuable, attackers look for possible vulnerabilities and developers for defenses.

In case of an implementation of cryptographic algorithm, the main objective of an attacker is to recover a secret key that could be a symmetric key or the private key of an asymmetric encryption algorithm. Most of the cryptographic algorithms used in commercial applications are publicly known and recovering the key facilitates the attacker to decrypt all the future and past communication.

Rest of the section summarizes security issues produced by attacks against given FPGA implementation.

*Cloning* – Due to the generic nature of FPGAs, the image generated for one FPGA device could also be used in any other device for same family and size. In such case, attackers can clone image and use them in cheaper devices that can compete with the originals. This security vulnerability is usually common in volatile FPGA, where the configuration data is stored externally in a nonvolatile memory and is transmitted to the FPGA at power-up. An adversary could easily eavesdrop on the transmission and get the configuration file. There are many different concerns with regards to cloning. For original system developers, the clone system can hurt the bottom line after significant development investments and it could affect the reputation of the original product if the clone or poor quality products are marketed as originals. Another threat is cloning /copying of a cryptographic algorithm together with its key. In few cases it is possible to run the cloned application in decryption mode to decrypt communication. An attacker can also launch man-in-the-middle attack and masquerade as the attacked communication party.

*Black Box Attack* – Black box attack is a classical method to reverse engineer a chip. The attacker launches a variant of known-plaintext attack, where an attacker inputs all possible combinations, while saving the corresponding outputs. This attack can be used to reveal secret information such as secret keys and code books. However this attack is only successful if a small FPGA with specific inputs and outputs are attacked and a lot of processing power is available. With today's design complexity and the size of state-of-the-art FPGAs this attack is not really a threat nowadays.

*Readback Attack* – It is a feature in FPGAs, where the snapshot of the FPGAs current state could be retrieved while it is still in operation. It is used for read a configuration out of the FPGA for easy debugging. The image (snapshot) is different from original bit stream by missing the header, footer and initialization commands etc. But an attacker can easily readback the design, add the missing

static header and footer and use it in other device. The feature can be prevented by adding few security bit provided by the manufactures. Xilinx provides security bit for disabling readback, but it can be easily found. When these security bit are used, multiple, majority-voted, disabling registers are activated with the FPGA to prevent readback [SAD] [JMA]. Despite of countermeasures, an adversary is still able to attack FPGAs with fault injection. This kind of attack was first introduced in [DBR], where it was shown to break public-key algorithms by exploiting hardware faults. There has been many publications presenting different techniques to insert faults e.g., electromagnetic radiation [JJQ], infrared laser [CAJ], and flash light [SSR].

*Reverse engineering the image/bitstream* - Once the attacker is in possession of the image, assuming the case where image is unencrypted. The attacker will be able to transform bitstream into a functionality equivalent description of the original design. Image could be reversed fully or partially. Partial image can be used for the extraction of data for the image such as secret keys or Block Ram/LUT content without gaining the full functionality of the design. In full reverse image one can gain the full functionality of the design and reproduce another bitstream completely different from the original one such that it would be hard to prove in fingerprint.

*Side Channel Attacks* – An attack based on the information gained due to the unwanted leak in the physical implementation of a cryptosystem instead of theoretical weakness in the algorithm are known as side-channel attacks. Example of side channels include in particular- power consumption, timing behavior and electromagnetic radiation. Today while implementing an encryption system on FPGA, additional input / output are used which are not the plaintext or ciphertext. The devices produce timing information, radiations, power consumption statistics and lot more. Additionally these devices also have additional “unintentional” inputs such as voltage that can be tampered with to cause predictable outcomes. Side channel attacks make use of all this information with other know techniques to retrieve keys. In terms of power analysis side channel side channel attacks are further classified as Simple Power Analysis (SPA) and Differential Power Analysis (DPA), where the attacker studies the power consumption of an encryption device. SPA is based on the visual representation of the power consumption of a device will encryption is in process. The attacker simply observes system power consumption, as the amount of power consumed varies depending on the instruction performed. DPA consists not only of visual but also statistical analysis and error-correction statistical methods, to obtain information about keys.

## 1.3 Confidentiality and Authentication

Communication between two parties over an insecure channel often concerns with two main security objectives: confidentiality and authenticity of the data. The objective of confidentiality is to keep the contents of the information secure and no one but the sender and authorized receivers are able to read the data. Authentication of message data verifies the origin and improper or unauthorized modification of data.

In the past, confidentiality of the data was the main and probably the only issue that was considered. This was mainly because no other security objectives such as authentication or integrity prevented to have access to the information. Only message encryption can protect data from eavesdroppers. However encryption of messages provides some sort of authentication but as compared to present authentication techniques it is weak and cannot be relied upon. In addition to confidentiality, authentication services have been implemented but as add on feature to provide extra information security. Encryption algorithms are used to ensure confidentiality while Message Authentication Codes (MAC) can be used to provide authentication. In past few years, techniques have been invented which can combine encryption and authentication into a single algorithm. By combining these two security features and performing single pass operation might possibly provide following advantage for hardware implementation:

- The rapid growth of portable electronic devices with limited area has opened a vast scope for compact circuit design opportunities. Implementation of single algorithm instead of two separate algorithms with definitely has less area requirements. Reduction in area requirements on chip is directly proportional to the reduction in cost.
- Small and compact designs tend to consume less power as compared to bulky designs. This is an attractive feature for low-power devices like Cellular phone, PDAs and smartcards.
- Even though separate keys are used for encryption and authentication for better security of the system. But both the keys are usually derived from the same master key. This will have a slight advantage with regards to the key storage issues over separate algorithms.
- Most of the new designs target performance goals like throughput and throughput-area trade-off. In many cases, combined schemes are based on block ciphers, and designers have tried to be efficient with the number of block cipher calls required for getting both confidentiality and authentication from the algorithm. Based on the mode of the operations some of these combined schemes can run in parallel and achieve much higher speed than older techniques.

From above, the cryptographic schemes that provide both confidentiality and authentication are called authenticated encryption schemes. The scheme is designed in such a way that the sender produces the ciphertext as well as an authentication tag which is verified by the receiver.

## **1.4 Motivation**

The main focus of the thesis is to propose a single pass authenticated encryption scheme to achieve faster encryption and message authentication for reconfigurable chips such as FPGAs in Software Defined radios. The motivation to focus only on this issue comes from the fact that the information security is one of the key relevant aspects of SDR, whether it is for data transmission or downloading of radio parameters or upgrades. Additionally, proposing a crypto solution for reconfigurable chips instead of ASICs gives us the possibility to improve and correct hardware components. This will reduce the vulnerabilities in the device and improves radio interoperability and upgradeability.

The future of communication systems is expected to be based on SDR principals. SDR provides an efficient and comparatively inexpensive solution to the problem of building multi-mode, multi-band, multi-functional wireless devices that can be enhanced using software upgrades. As such SDR can be considered an enabling technology that is applicable across a wide range of areas within the wireless infrastructure, both military and commercial. The main issue in these communication systems is security, whereby an adversary can take control of a radio system to their advantage. Additionally, if a third party modifies a stream at lower layers then higher layers can be caused damage upon i.e. application level of a mobile phone. As a result, documents and traditional applications of a device can be changed. Thus, various security functions such as authenticity, integrity and confidentiality need to be supported in SDR transceivers.

The goal of this research is to find an encryption scheme that can accomplish both message secrecy and authenticity in a single cryptographic primitive with the focus to achieve high throughput and minimal overhead for these radios. As, it is expected that the speed of the network will increase beyond Giga-bit so the need for fast cryptographic solution will be increased in the nearest future to take advantage of the future fast speed network.

So the above gives the motivation for the design and careful implementation of an Authenticated Encryption Stream Cipher for optimal results in terms of security and speed. Different hardware architectures are also taken into consideration to explore trade-off between area utilization and throughput in the implementation.



## 1.5 Problem Definition

The problem tackled in this thesis is on how to develop a cryptosystem that could cover security features like confidentiality and authentication in single pass as opposed to traditional two pass approach. Additionally, keeping the security of the design at par with traditional systems, another problem was how to downsize the system and increase the efficiency so the design is ideal for implementation of low-cost, low-power SDR device. With this in mind this research gives answers to the following questions:

- What important parameters need to be kept under consideration while designing a single pass authenticated encryption scheme?
  - Parameters like – two separate keys for encryption and authentication, Initialization vector, underlying block cipher. For hash functions, security parameters like – pre-image resistance, second pre-image resistance and collision resistance.
- What is the right trade-off between the security and the performance of the cipher?
  - Design of block ciphers are mostly built from a round function in an iterative way. Using signal round function for encryption and its repeated use leaves patterns. This could be used to break the cipher. To add additional security extra rounds can be added but it also increases the complexity of the cipher. Thus, there should be right trade-off between the number of rounds and performance of the cipher.
- What are the optimum choices of hardware architecture for cryptosystem?
  - In this thesis we explore different hardware architectures for implementation. Hardware architectures for cryptosystem are based on feedback and non-feedback modes of operation. Such as basic iterative architecture, partial /full loop unrolling architecture and pipelined architecture.
- What is the trade-off between minimum area, minimum power consumption, maximum throughput, maximum throughput to area ratio, etc?
  - Hardware architectures mentioned above have their own pros and cons, ex: in a basic iterative architecture only single block cipher is implemented as a combinational logic where as in partial loop N rounds are implemented as a combinational logic.

In this thesis each one of these questions have been analyzed and solved.

## **1.6 Limitations**

Many issues need to be covered in order to make SDR systems secure. The security of SDRs has followed many general directions such as:

- Security Threats on the devices;
- Secure download of new radio parameters;
- Security concerned with spectrum management and policies;

In this thesis we have limited our research on the authenticity and privacy of the data transmission. This could also be applied for the secure upgrade and download of new radio parameters.

Another limitation is based on our proposed Authenticated Encryption (AE) cipher. ASC-1 operates in a cipher feedback mode (CFB) in order to compute an authentication tag, which means the encryption core of the scheme is limited to run in feedback modes hardware architectures, i.e., basic iterative or partial loop unrolling. This will certainly limit the throughput of the design as compared to the designs in non-feedback mode hardware architectures. However in such kind of encryption modes, the encryption and decryption operations are similar so there is no need for inverse block ciphers. From implementation point of view this is considered as an advantage.

As mentioned this research is mainly focused on two fundamental information security goals: confidentiality and message authentication. Issues like key exchange between communicating parties are not considered. We assume that the recipient has knowledge of the secret key that is used to derive the Initialization vector and keys for encryption and authentication. Additionally we also assume that the secret / master key is securely placed in a crypto module inside the FPGA.

## **1.7 Contribution**

The contributions given throughout this thesis are the following:

- An Authenticated encryption scheme ASC-1 (Authentication Stream Cipher One) that is designed using a stream cipher approach instead of a block cipher mode approach;
- Proving the security of ASC-1, by showing that it is secure if one cannot distinguish the case when the round keys are uniformly random from the case when the round keys are derived by the key scheduling algorithm of ASC-1;
- Design and implementation of ASC-1 and Introducing a new crypto unit for reconfigurable chips;
- Analysis of performance parameters such as Area utilization, throughput and latency by releasing different hardware architectures.

The Publications performed during the Ph.D work:

- Samant Khajuria and Henrik Tange, *Implementation of Diffie-Hellman Key Exchange on Wireless Sensor Using Elliptic Curve Cryptography : Wireless Communication Society, Vehicular Technology, Information Theory and Aerospace and Electronic Systems Technology, Wireless VITAE 2011*
- Goce Jakimoski and Samant Khajuria, *ASC-1 : An Authenticated Encryption Stream Cipher : Selected Areas of Cryptography (SAC), LNCS 7118, Springer, pp. 356-372, 2012.*
- Samant Khajuria and Birger Andersen, *Authenticated Encryption for Low-Power Reconfigurable Wireless Devices: Accepted for Journal of Cyber Security and Mobility, 2012.*
- Samant Khajuria, Comparative Analysis of the Hardware Implementation of ASC-1 : *In Pipeline.*

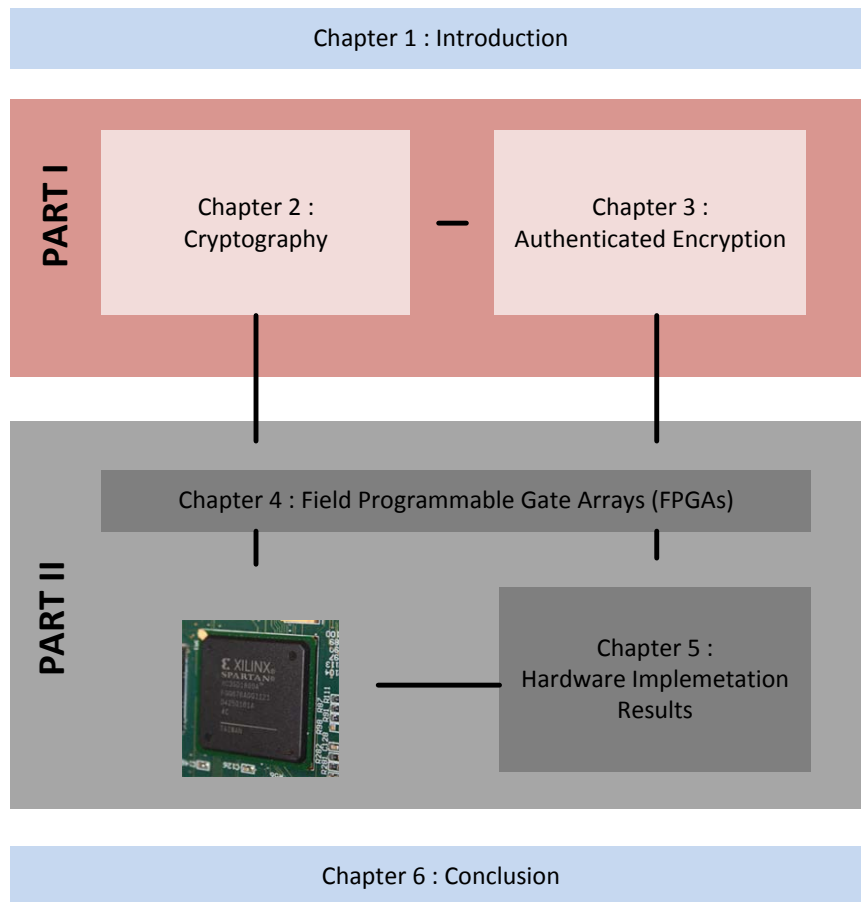
## 1.8 Organization

Figure 1.2 shows the outline of this thesis, as shown below the thesis is divided into two parts. Part I covers the cryptography part of the thesis and Part II is based on the design and implementation for the cryptographic schemes.

**Chapter 1** provides the overview on Software Defined Radios and the benefits of re-configurable architecture, together with the need of security for these architectures and the motivation for the

work developed and described in the thesis.

**Chapter 2** gives the general introduction of two prime security requirements – Confidentiality and Authentication of data. We introduce the cryptographic concept such as symmetric-key encryption and cryptographic primitives known as block and stream ciphers with some examples. Under Authentication we also give some background information on cryptographic hash functions, where both un-keyed and keyed hash functions are covered.



*Figure 1.2: Thesis Outline*

**Chapter 3** is based on Authenticated Encryption (AE) schemes, where we initially discuss the solution to the problem of privacy and authentication in a traditional manner known as “generic composition”. Then we look into the advantages and disadvantages of 2-pass and 1-pass combined approach with few well known AE schemes. Based on this finally we present “ASC-1 : An Authentication Encryption Stream Cipher ” and proof of security of ASC-1 by discussing The

information-theoretic case and computational security analysis.

**Chapter 4** presents the design and implementation of ASC-1 for FPGAs. At the start of the chapter we discuss the architecture and Implementation flow of the FPGAs followed by the role of FPGA in SDR with an example of Gnu radio framework and USRP hardware. This chapter also discusses the generic SDR structure and System-on-chip (SOC). Finally we present an implementation of ASC-1 for Xilinx Spartan-3 xc3s700an FPGA. For better performance and efficient resource allocation the design is divided into two parts – Initial phase generation and encryption process.

**Chapter 5** summarizes the results for hardware implementation of ASC-1. Performance parameters like area utilization, throughput and latency are also discussed and analyzed for different hardware architectures for feedback and non-feedback cipher modes. The results also show the optimal payload lengths of maximum throughput corresponding to the Bit Error Rate (BER) for different data rates. In the end we discuss OFDM modulation in LTE and WiMAX.

**Chapter 6** gives the final conclusion and outlook for the future work.

## References

- [JMI1] J.Mitola. Software Radios: Survey, Critical Evaluation and Future Directions, *In Proceedings of the National Telesystems Conference*, NY: IEEE Press, May 1992.
- [JMI2] J.Mitola and G.Q Maguire Jr. Cognitive radio: making software radios more personal. *IEEE Personal Communications*, 13-18, Aug.1999.
- [RJD] R. J. Lackey and D. W. Upmal. Speakeasy: The military software radio, *IEEE Commun. Mag.*, vol. 33, pp. 56–61, May 1995.
- [JTR] Software Communications Architecture Specification, JTRS Std. 2.2.2, Rev. FINAL, May 2006.
- [TUL] T. Ulversø. Software defined radio: Challenges and opportunities, *Communications Surveys Tutorials*, IEEE, vol. 12, no. 4, pp. 531 –550, 2010.
- [SHY] S. Haykin. Cognitive radio: brain-empowered wireless communications. *IEEE Journal on Selected Areas in Communications*, 23(2):201-220February 2005.
- [PKG] P.K Gopalakrishnan. Hardware Platforms for Software Defined Radio, KPIT Cummins Infosystems Ltd, V1.2 Sept-2011.
- [TAJ] Taj A. Strunman. Enabling Chaotic Waveform Diversity, *Waveform Diversity and Design Conference*, IEEE Conference publications, pp. 160-167, 2009.
- [HRM] R.Hill Myagmar and S. Campbell. Threat Analysis of GNU Software Radio, *In Proceedings of the WWC'05*. Palo Alto, CA, 2005
- [SS1] SDR system security, *SDRF-02-A-0006*, Tech. Rep., 2002.
- [SS2] A structure for software defined radio security, *SDRF-03-I-0010*,Tech. Rep., 2003.
- [SS3] SDR wireless security, *SDRF-04-I-0023*, Tech. Rep., 2004.
- [RFL] R. Falk, J. F. Esfahani, and M. Dillinger. Reconfigurable radio terminals - threats and security objectives *SDRF-02-I-0056*, Tech. Rep., 2002.
- [LMI] L. Michael, M. Mihaljevic, S. Haruyama, and R. Kohno. A framework for secure download for software-defined radio, *IEEE Communications Magazine*, vol. 40, no. 7, pp. 88–96, July 2002.
- [ABR] A. Brawerman, D. Blough, and B. Bing. Securing the download of radio configuration files for software defined radio devices. *In Proceedings of the International Workshop on Mobility Management & Wireless Access Protocols*, pp. 98–105,Sept. 2004.
- [PFL] P. Flanigan, V. Welch, and M. Pant. Dynamic policy enforcement for software defined radio. *In Proceedings of Software Defined Radio Technical Conference and Product Exposition*, Nov. 2005.

- [KSA] K. Sakaguchi, C. Fung Lam, T. Doan, M. Togooch, J. Takada, and K. Araki. ACU and RSM based radio spectrum management for realization of flexible software defined radio world, *IEICE Trans. Communications E Series B*, vol. 86, no. 12, pp. 3417–3424, Dec. 2003.
- [ATO] A. Tonmukayakul and M. Weiss. Secondary use of radio spectrum: A feasibility analysis, In *Proceeding of the Telecommunications Policy Research Conference*, Oct. 2004.
- [ACR] Alexandru Coman and Radu Fratila. Cryptographic Applications using FPGA Technology, *Journal of Mobile, Embedded and Distributed Systems*, vol. III, no.1, 2011
- [SAD] S. Drimer. Volatile FPGA design security – a survey, Computer Laboratory, University of Cambridge, IEEE Computer Society Annual Volume, 2008.
- [TWC] T. Wollinger and C. Paar. How secure are FPGAs in cryptographic applications?, In *Proceedings. of the 13th International Conference on Field-Programmable Logic and its Applications(FPL)*, pages 91–100, 2003.
- [JMA] J. M. Aplan, D. D. Eaton, and A. K. Chan. Security Antifuse that Prevents Readout of some but not other Information from a Programmed Field Programmable Gate Array. United States Patent, No. 5898776, April 27 1999.
- [DBR] D. Boneh, R. A. DeMillo, and R. J. Lipton. On the Importance of Checking Cryptographic Protocols for Faults. In *Advances in Cryptology — EUROCRYPT '97*, pages 37–51. Springer-Verlag, 1997. LNCS 1233.
- [JJQ] J.-J. Quisquater and D. Samyde. Electro Magnetic Analysis (EMA): Measures and Countermeasures for Smart Cards. In *International Conference on Research in Smart Cards, E-smart 2001*, pages 200 – 210, Cannes, France, September 2001
- [CAJ]C. Ajluni. Two New Imaging Techniques to Improve IC Defect Identification. *Electronic Design*, 43(14):37–38, July 1995.
- [SSR]S. Skorobogatov and R.J. Anderson. Optical Fault Induction Attacks. In *Workshop on Cryptographic Hardware and Embedded Systems — CHES 2002*, pages 2–12. Springer-Verlag, LNCS 2523. 2002

# **PART I – Algorithm Analysis & Development**





# 2

## Cryptography

The name Cryptography comes from the Greek words “kryptos” which means “hidden secret” and “graph” which means “writing” and is the art of hiding information. This definition may be historically accurate where the focus was only on the problem of secret communication and until 20<sup>th</sup> century, cryptography was an art. By late 20 century, a rich theory emerged enabling the rigorous study of cryptography as a science and today Cryptography is much more than secret information and art. For example, in addition to confidentiality it deals with the problems of data integrity, entity authentication, data origin authentication and much more.

As classical Cryptography is basically the process of encrypting and decrypting message, encryption is the process of converting normal message or communication (plaintext) into unintelligible text (ciphertext). Decryption is the reverse, moving from unintelligible ciphertext to plaintext. A cipher is a pair of algorithms which perform this encryption and decryption. This operation of a cipher is controlled both by the algorithms and the keys. In communication, cryptography is necessary when communicating over any unsecure medium, which includes particularly wireless communication. Within the context of any application-to-application communication, there are some specific security requirements including [MOV]:

*Confidentiality* - Ensuring that no one except the intended receiver should be able to read the message when transmitting a message over insecure channel. An unauthorized eavesdropper should not get any information about the contents of the message. Also stored data should be protected against unauthorized access.

*Authentication* - The process of proving one's identity. This function applies to both entities and information itself. Two users trying to communicate should identify each other and/or information delivered over a communication channel should be authenticated. For the above mentioned reasons, this aspect of cryptography is usually subdivided into two major cases: *entity authentication* and *data origin authentication*. Entity authentication assures the identity of the entity involved and data origin authentication ensures the integrity of the data, if the message is modified and the source had changed.

*Data integrity* - Assuring the receiver that the received message (neither accidentally nor on purpose) has not been altered in any way from the original. To assure this, one must be able to

detect manipulation by unauthorized parties.

*Non-Repudiation* – A mechanism to prove that the sender really sent the message. This prevents an entity from denying previous actions.

A fundamental goal of cryptography is to adequately address these four areas in both theory and practice. For secure communication there are two types of cryptographic schemes typically used, Symmetric Encryption (Secret-key cryptography) and Asymmetric Encryption (Public-key cryptography). In this thesis we focus on symmetric encryption schemes and Hash functions which are supposed to provide confidentiality and message authentication.

## 2.1 Symmetric Encryption

Two parties want to communicate over an insecure channel without allowing an eavesdropper to obtain or modify any information about their conversation. Another problem is the key management issue, where we assume that both the parties have somehow exchanged a secret key over a secure channel. The main purpose of these encryption algorithms is to protect the secrecy of the message transmitted over an insecure channel.

Generally an encryption scheme consists of two mathematical transformations: an encryption function  $E$  and a decryption function  $D = E^{-1}$ . An encryption function takes the plaintext ( $P$ ) as input and transforms into the corresponding ciphertext  $C = E(P)$ , and a decryption function inverts the encryption function  $D(C) = P$ . In order for this encryption scheme to work,  $E$  should be designed in such a way that an eavesdropper cannot extract any information about the plaintext after intercepting ciphertext and the decryption function should be kept secret. By looking into the practical issues, keeping the whole algorithm secret is never a good idea because it means that several algorithms are needed for different partners [CAB]. The idea to overcome this problem is to construct a parameterized encryption algorithm. In this scheme both encryption and decryption function uses an additional parameter also called as secret key ( $k$ ),  $D_k, (E_k(P))$  for  $k' \neq k$  does not reveal any information about the plaintext  $P$ . This secret key is usually a bit string and depending on encryption scheme the length of this key could vary from few bits to couple of hundred bits. A single key is used from both encryption and decryption functions. The sender uses the key  $k$  to encrypt the plaintext  $p$  and sends the ciphertext  $C = E_k(P)$  to the receiver. The receiver applies the same key  $k$  to decrypt the message and recover the plaintext  $P = D_k(C)$ .

**Definition 2.1** [STN]: A symmetric cryptosystem, also called a symmetric encryption scheme, is a

five-tuple  $(P, C, K, E, D)$  where  $P$  is the finite field set of plaintexts,  $C$  is the finite set of ciphertexts and  $K$  is the key space. For each key  $k \in K$  there is an encryption function  $E_k \in E$  with respect to  $k$ ,

$$E_k : P \rightarrow C$$

and a corresponding decryption function  $D_k \in D$ ,

$$D_k : C \rightarrow P$$

Such that  $D_k(E_k(p)) = p$  for all plaintexts  $p \in P$ .

From the above, it is clear that the key must be known to both the sender and the receiver. The biggest difficulty with this approach is the distribution of the key; exchange of secret key over a secure channel beforehand. In 1970s, Diffie and Hellman proposed *trapdoor one-way* functions where the secrecy of the encryption function was not necessary. These are functions which are easy to evaluate, but cannot be efficiently inverted, unless some extra information (the trapdoor) is given. This insight gave birth to Public-Key Cryptography (PKC). Two keys are used in public-key cryptosystem; a public and a private key, where the public defines the encryption function and the private key is the trapdoor information needed to invert the encryption function. When user  $A$  wants to send a message to user  $B$ , user  $A$  then encrypts the message using  $B$ 's public key. Public is usually obtained by a direct exchange over insecure channel or from a trusted source, often referred to as Certificate Authority (CA). Afterwards, user  $B$  is able to decrypt the message using its own private key. Public-key cryptography has huge advantage as both the parties do not have to exchange any information over secure channel before they start exchanging messages. However, symmetric cryptography still plays a vital role in practical systems because symmetric encryption is an order of magnitude more efficient than public key encryption. In the remainder of this thesis we only consider symmetric encryption schemes.

Symmetric encryption algorithms are traditionally divided into two categories: *stream ciphers* and *block ciphers*; A stream cipher operates on a single bit, byte, and word at a time and implements some form of feedback mechanism so that the key is constantly changing. As the name says in a block cipher, the scheme encrypts one block of data at a time using the same key on each block. In general, the same plaintext block will always encrypt to the same ciphertext when using the same key in a block cipher whereas the same plaintext will encrypt to different ciphertext in a stream cipher.

## 2.2 Block Cipher

Block ciphers can be either symmetric-key or asymmetric-key. In this thesis our main focus is symmetric-key block ciphers. Symmetric-key block ciphers play an important role in many cryptographic systems. Their flexibility allows construction of pseudorandom number generators, stream ciphers, Messages Authentication codes and hash function.

As the name suggest block ciphers operate on fixed lengths of plaintext blocks, cipher function maps  $n$ -bit plaintext to  $n$ -bit ciphertext blocks where  $n$  is called the blocklength of the cipher. The mapping is controlled by the second input – the secret key. It is generally assumed that the key is chosen at random. For  $n$ -bit plaintext and ciphertext blocks and a fixed key, the encryption function is a bijection, defining a permutation and substitution on  $n$  -bit vectors. Each key potentially defines a different bijection. As mentioned two important parameters of a block cipher are the block size which are normally 64 or 128 bits and key size generally ranges from 56 to 256 bits.

Most Block ciphers follow the same general approach. It simply consists of sequence of operations called *round functions* repeatedly applied  $r$  times on the input. Round function takes the output of the previous round and output the input of the next round. Addition to the intermediate output from the previous round, round function also uses a *subkey* (round key) which is usually derived from the secret key by a key schedule algorithm and are called *key schedule*. The receiver should be able to uniquely decrypt the ciphertext, so for the fixed key the round function is by definition bijective.

Majority of block ciphers are either realized as Feistel ciphers or Substitution-Permutation (SP) networks. In Feistel cipher, the round function operates only on one half of the block while the other half remains unchanged where as in SP networks, the round function combines layers of invertible functions such as substitutions and permutations. The substitution functions, also called as S-boxes (Substitution boxes) are usually implemented as look-up tables. The S-boxes are non-linear and introduce local confusion. The permutation layer is an affine transformation that operates on the complete block and introduces diffusion.

**Definition 2.2** [MOV] - An  $n$  -bit block cipher is a function  $E : V_n \times K \rightarrow V_n$ , such that for each key  $k \in K$ ,  $E(P, k)$  is an invertible mapping (the encryption function for  $k$ ) from  $V_n$  to  $V_n$  , written  $E_k(P)$ . The inverse mapping is the decryption function, denoted  $D_k(C)$ .  $C = E_k(P)$  denotes that ciphertext  $C$  results from encrypting plaintext  $P$  under  $k$ .

Block ciphers generally process plaintext in relatively large block. For plaintext messages exceeding

one block in length, various modes of operations for block ciphers are used.

**Electronic CodeBook (ECB)** [MOV] [BCM] mode, as shown in figure 2.1, it is the simplest and most obvious application: the secret key, together with the encryption (forward cipher) function is applied directly and independently to each block of the plaintext resulting in the sequence of ciphertext blocks. Similarly on the decryption side, decryption (inverse cipher) function is applied to the ciphertext resulting in the sequence of plaintext blocks. The drawback of this method is that two identical plaintext blocks (under the same key) will always generate the same ciphertext block. Thus it does not hide patterns well and is prone to a variety of brute-force attacks.

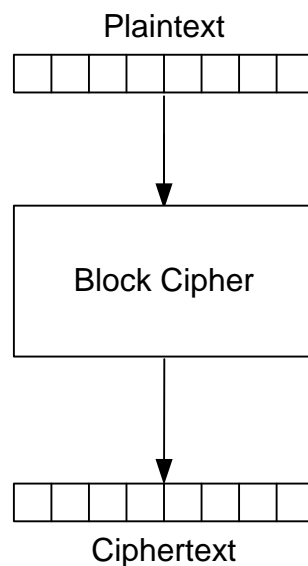


Figure 2.1: Electronic Codebook

**Cipher Block Chaining (CBC)**[MOV,BCM] mode adds a feedback mechanism to the encryption scheme. As shown in figure 2.2, an initialization vector is exclusively – ORed (XOR-ed) with the first plaintext block. An encryption function is applied to the first input block resulting in the output ciphertext block. The output block is XOR-ed with the second plaintext block to form next input block. Each plaintext block is XOR-ed with the previous output block to produce a new input block. Similarly in CBC decryption, the decryption function is applied to the first ciphertext block and the resulting output is XOR-ed with the initialization vector to recover the first plaintext block. For second block, ciphertext from previous block is applied instead of IV to recover plaintext blocks. In CBC decryption all the ciphertext blocks are immediately available, so multiple decryption operations can be performed in parallel. Identical plaintext result in identical ciphers under same key and Initialization vector, by changing the IV to the counter mode will result in different Ciphertext. Another drawback in CBC mode is error propagation; a single bit error in the  $n^{th}$  cipher block  $c_n$  will effect decryption of  $c_n$  and adjacent block  $c_{n+1}$ . Block  $p'_{n+1}$  recovered after

decryption of  $c_{n+1}$  will have bit errors precisely where  $c_n$  did. Thus an attacker can get an opportunity to cause predictable bit changes in  $p_{n+1}$  by altering corresponding bits to  $c_n$ .

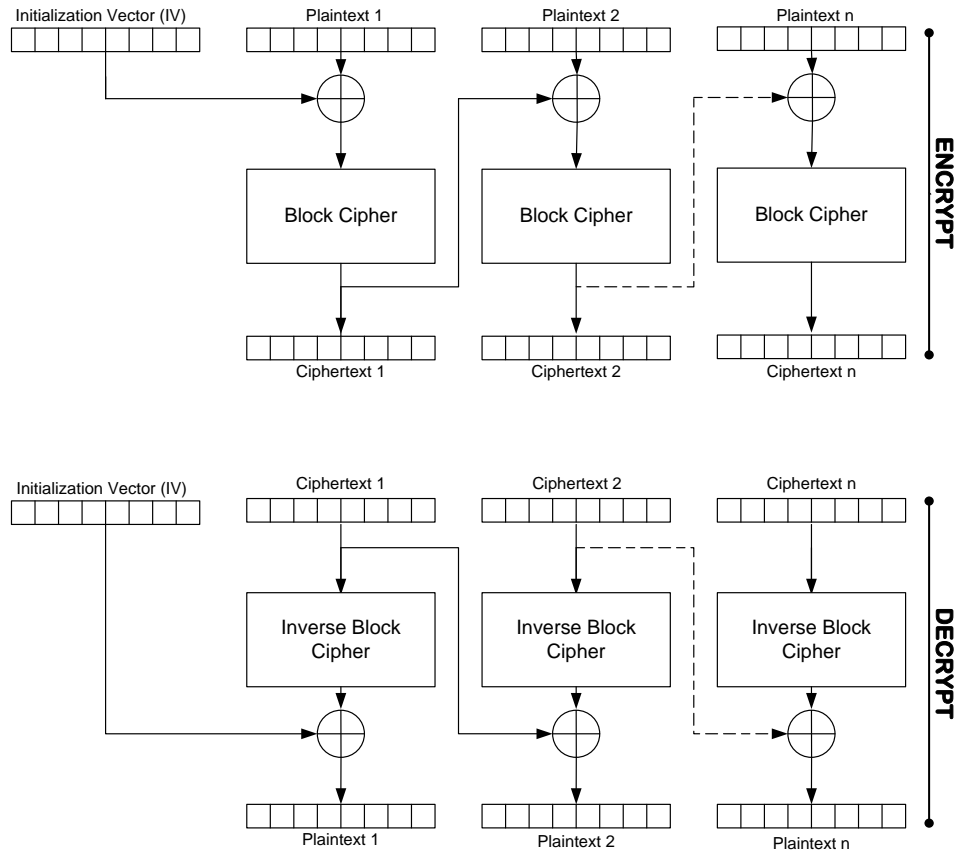


Figure 2.2: Cipher Block Chaining

**Cipher Feedback (CFB)** [MOV,BCM] mode is a block cipher implementation as a self synchronizing stream cipher. In CFB mode, encryption function is applied to the Initialization vector to produce the first output block as shown in figure 2.3. First ciphertext block is produced by XOR-ing plaintext block with  $S$  most significant bits of the output block; where  $S$  is an integer parameter and sometimes incorporated into the name of the mode, e.g., 1-bit CFB mode, 8-bit CFB mode, 64-bit CFB mode etc. For second input block, bits of the first input block circularly shift  $S$  positions to the left, and then the ciphertext block replaces the  $S$  least significant bits of the result. Whereas in CFB decryption, similar encryption function is applied to the input block, the IV is the first input block and plaintext is recovered by XOR-ing ciphertext with the  $S$  most significant bits of the output block. Operations in CFB decryption can be performed in parallel if the input blocks are first constructed from the IV and the ciphertext. One flip of bit in a  $S$ -bit ciphertext block, will affect the

decryption of that block and the next  $\lceil n/S \rceil$  ciphertext blocks i.e., until entire block has shifted entirely out of the shift register. The decrypted message  $p'_n$  will differ from  $p_n$  precisely in the bit positions where  $c_n$  was in error. Thus an attacker gets an opportunity to cause predictable bit changes in  $p_n$  by altering corresponding bits to  $c_n$ .

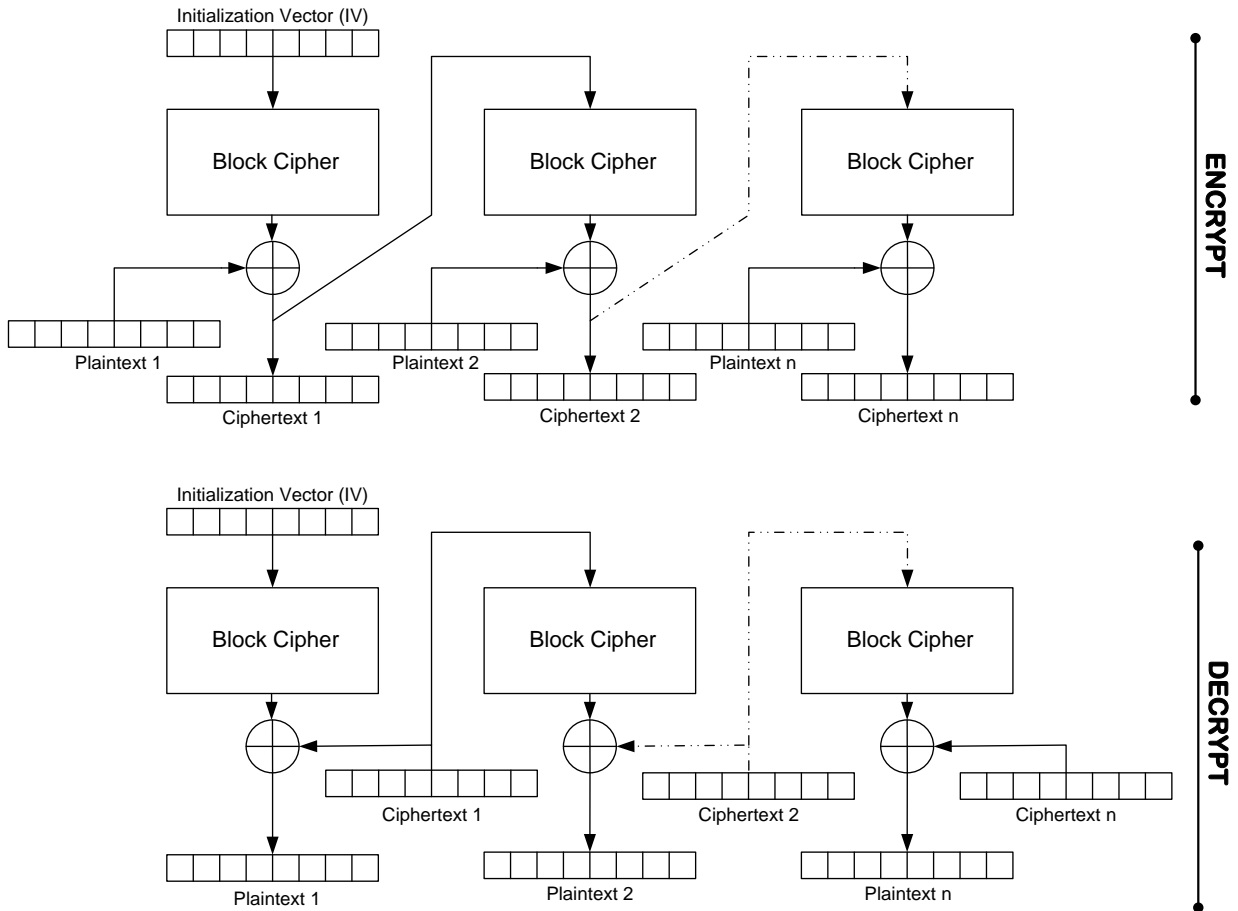


Figure 2.3: Cipher Feedback

**Output Feedback (OFB)** [MOV,BCM] mode is a block cipher implementation, conceptually similar to a synchronous stream cipher. In OFB mode, initialization vector is transformed by the encryption function to generate first block output. As shown in figure 2.4, the output is then XOR-ed with the plaintext block to produce the first ciphertext block. Encryption function is then invoked to the first output block to produce the second output block; the block is then XOR-ed with next plaintext block to produce ciphertext. Similarly at the decryption side, same encryption function is used for producing output blocks. These output blocks are then XOR-ed with the ciphertext blocks to produce plaintext blocks. In both the cases (encryption and decryption), output of the encryption function depends on the results of the previous encryption functions;



therefore, multiple encryption functions cannot be performed in parallel. However if IV is known then output blocks can be generated in advance. Bit errors in the ciphertext will affect the decryption of only those specific bits where error occurred. However OFB mode recovers from the error bit, but cannot self-synchronize after loss of ciphertext bits. In this case, explicit re-synchronization is required.

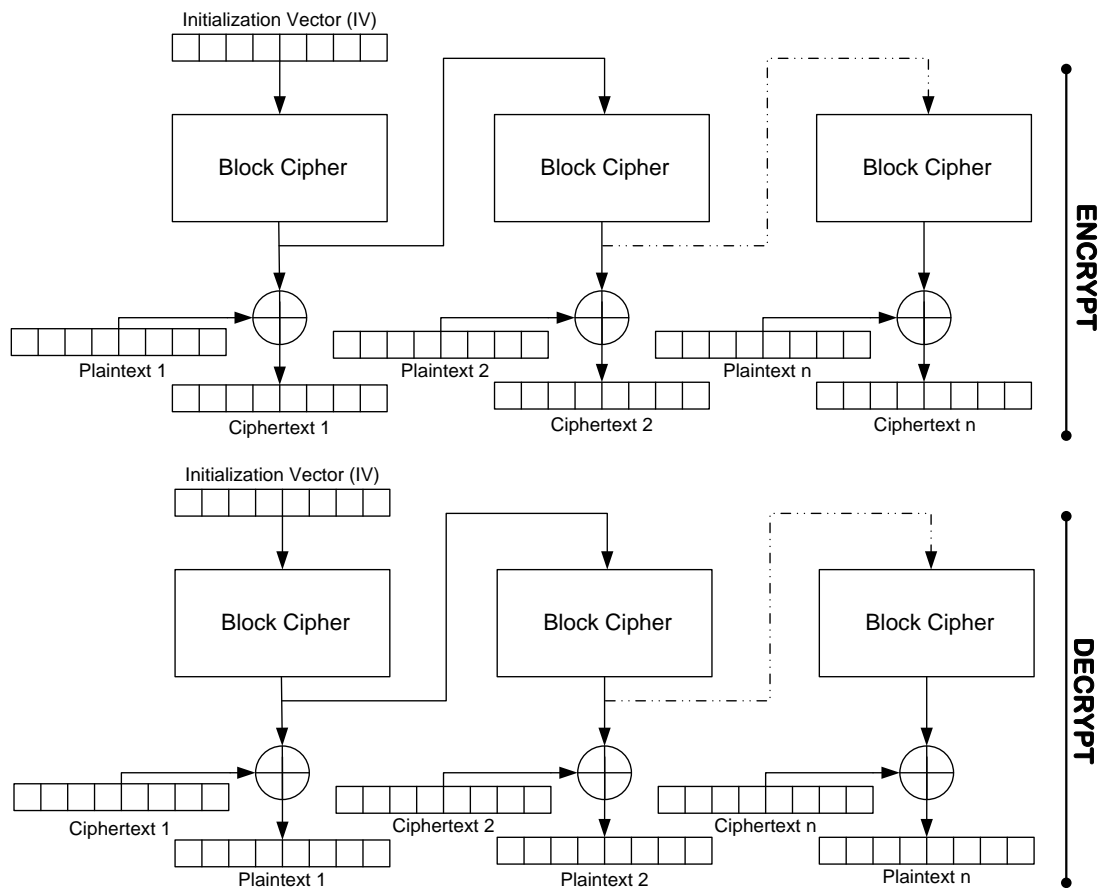


Figure 2.4: Output Feedback

### 2.2.1 Advanced Encryption Standard

In September 1997, the National Institute of Standards and Technology (NIST) issued a public call for proposals for a new block cipher to succeed the Data Encryption Standard (DES). Out of 15 submitted algorithms the Rijndael cipher by Daeman and Rijmen was chosen to become the new Advanced Encryption Standard (AES)[DRA] in 2001. Some of the requirements in the context held by NIST were that AES should have a block length of 128 bits and have a support for key lengths of

128, 192 and 256 bits. Security, efficient computation and straight forward implementation in software and hardware were the main focus point in the choice of Advanced Encryption standard. Implementation of AES included several steps shown below.

The Advanced Encryption standard is a block cipher with a fixed block length of 128 bits. It supports three different key lengths 128 bits, 192 bits and 256 bits. For encryption, data is divided into 128 bit blocks transforming it in  $n$  rounds into a 128-bit output block. Another crucial parameter is the key  $k$  used in each round  $r$  in  $n$  round. These key are derived from the AES key in so called key expansion. The number of  $n$  rounds depends on the length of the key:  $n = 10$  for 128 bit keys,  $n = 12$  for 192 bit keys, and  $n = 14$  for 256 bit keys.

*Key Expansion* – The AES algorithm takes the Cipher key,  $K$ , and performs a key Expansion Routine to generate a key schedule. The key Expansion generates a total of  $B$ (Block size) words: the algorithm requires an initial set of  $B$  words, and each of the  $R$  (number of rounds) rounds requires  $B$  words of key data. The resulting key schedule consists of a liner array of 4-byte words.

*AES encryption Specification* - As shown in figure 2.5 and described in algorithm 2.1, AES encryption of a 128-bit input block uses four basic operations, SubBytes, ShiftRows, MixColumns and AddRoundKey described in detail in the following.

*AddRoundKey* – The AddRoundKey operation inputs each byte of the state with the round key derived from the cipher key using key schedule and returns the bitwise *XOR* of the AES state.

*SubBytes* – The SubBytes operation substitutes each byte of the state to another byte, it's a non-linear substitution step where each byte is replaced with another according to a lookup table. The input to the SubBytes can be a 16 byte AES state type for encryption or 4 bytes from key expansion.

*ShiftRows* – The ShiftRows operation is a transposition step where each row of the state is shifted cyclically a certain number of steps.

$$\begin{pmatrix} b_{00} & b_{01} & b_{02} & b_{03} \\ b_{10} & b_{11} & b_{12} & b_{13} \\ b_{20} & b_{21} & b_{22} & b_{23} \\ b_{30} & b_{31} & b_{32} & b_{33} \end{pmatrix} \rightarrow \begin{pmatrix} b_{00} & b_{01} & b_{02} & b_{03} \\ b_{11} & b_{12} & b_{13} & b_{10} \\ b_{22} & b_{23} & b_{20} & b_{21} \\ b_{33} & b_{30} & b_{31} & b_{32} \end{pmatrix}$$

*MixColumns* – Unlike ShiftRows operation, MixColumns operate on the columns of the state, combining four bytes in each column. This operation considers the bytes of the AES state as elements of  $F_{2^8} = F_2[X]/(X^8 + X^4 + X^3 + X + 1)$ . It multiplies the matrix  $B$  with the fixed circulant matrix.

$$B \rightarrow \begin{pmatrix} X & X+1 & 1 & 1 \\ 1 & X & X+1 & 1 \\ 1 & 1 & X & X+1 \\ X+1 & 1 & 1 & X \end{pmatrix} \cdot B$$

---

#### Algorithm 2.1 AES-128 encryption

---

Input: 128- bit input block  $B$ , 128-bit round keys  $K_0, \dots, K_{10}$ .

Output: 128-bit block of encrypted output

$A \leftarrow \text{AddRoundKey}(A, K_0)$  -- [Initial state]

For  $i$  from 1 to 9 do

$A \leftarrow \text{SubBytes}(A)$

$A \leftarrow \text{ShiftRows}(A)$

$A \leftarrow \text{MixColumns}(A)$

$A \leftarrow \text{AddRoundKey}(A, K_i)$

End for

$A \leftarrow \text{SubBytes}(A)$

$A \leftarrow \text{ShiftRows}(A)$

$A \leftarrow \text{AddRoundKey}(A, K_{10})$

Return  $A$

---

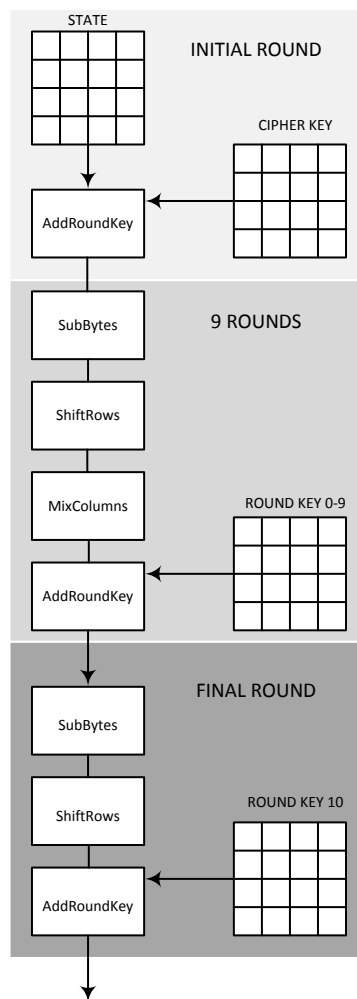


Figure 2.5: 128 bits Advanced Encryption Standard

## 2.3 Stream Ciphers

Stream ciphers are another major class of symmetric-key encryption schemes. Unlike block ciphers where blocks of plaintext data is transformed into ciphertext blocks, stream ciphers encrypt one bit (or a small chunk of bits) of a plaintext message one at a time using an encryption transformation which varies with time. Stream ciphers are in general faster than block cipher and less complex in hardware implementations [MOV]. When buffering is limited or if there is a need for bit to be individually processed as they are received for example in telecommunications, Stream ciphers are more appropriate. Another important advantage of stream cipher is that they have a very limited or no error propagation.

Even though various design principles for stream ciphers have been proposed and broadly

analyzed [CAB], block ciphers have been systematically replacing stream ciphers for example, Stream cipher A5/1 used in GSM standard has been replaced by Kasumi block cipher [EBD, AAD, EAG] and in wireless standard 802.11, RC4 stream cipher was replaced by newer standards based on AES block cipher. Another major disadvantage with the popularity of the stream cipher is most of ciphers used in practice tend to be patented; block ciphers on the other hand have been proposed, published and standardized. In most of the cases block ciphers are defined as memoryless; as same function is used to encrypt all the blocks. In contrast, stream ciphers are said to have some memory as the encryption function may vary as the plaintext is processed. The encryption function may vary not only because of the key and plaintext but also on the current state, these are known as state ciphers. By adding small amount of memory to the block cipher for example, as in the CBC mode results in a stream cipher with large blocks [MOV].

Much of the popularity of stream ciphers can undoubtedly be attributed to the work of Shannon in the analysis of the one-time pad, originally known as the Vernam cipher [VER]. Shannon proved that one-time pad provides perfect secrecy if the key is of same length as the message and chosen uniformly at random. The one-time pad uses a long string consists of bits that are chosen completely at random called key-stream. This key-stream is then combined with the plaintext on a bit by bit basis. The downside of this design is that the key and message length should be of same size and can only be used once. Let  $m (m_0 m_1 \dots m_{n-1})$  be the plaintext message and key-stream  $(k_0 k_1 \dots k_{n-1})$ , which is the same length as the message  $m$ . Then the ciphertext  $c = c_0 c_1 \dots c_{n-1}$  is defined by  $c_i = m_i \oplus k_i$  for  $0 \leq i \leq n - 1$  where  $\oplus$  denotes bitwise exclusive-or.

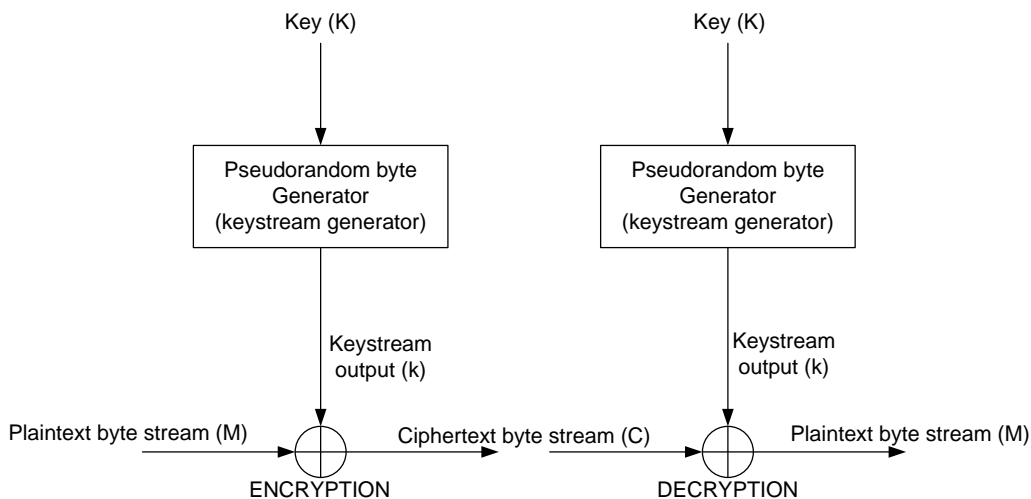
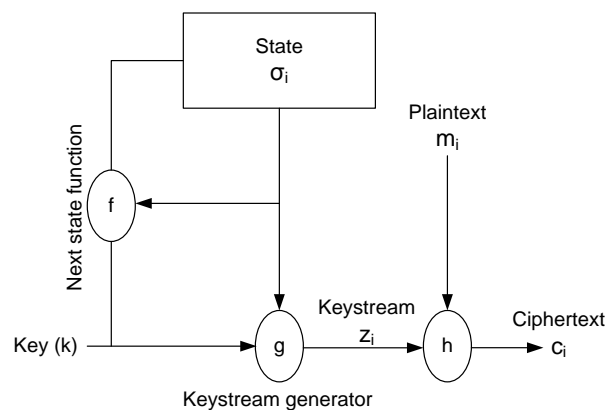


Figure 2.6: Stream Cipher

This is an obvious drawback in one-time pad as it increases the difficulty of key distribution and

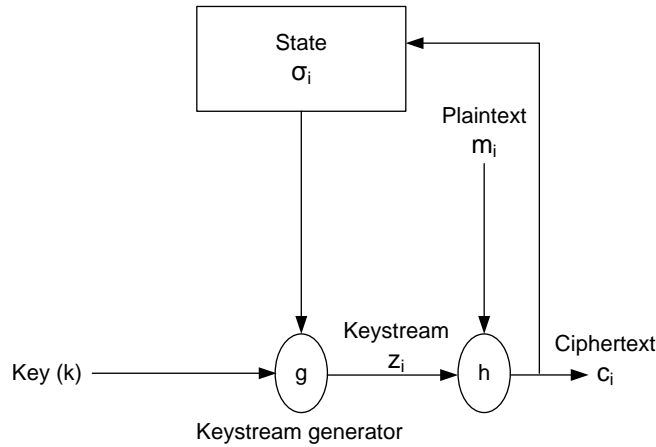
key management. The idea behind the stream cipher design is to use a short secret key to generate a long sequence of bits (key-stream), which appears to be random and this stream is combined with the plaintext to produce the ciphertext as shown in figure 2.6.

### 2.3.1 Synchronous Stream Cipher



### 2.3.2 Self-Synchronizing Stream Cipher

By definition, A self-synchronizing or asynchronous stream cipher is one in which the key-stream is generated as a function of the key and a fixed number of previous ciphertext digits [MOV]. This stream cipher solves the problem of synchronization and resumes correct decryption if the key-stream generated by the decrypting unit falls out of synchronization after a ciphertext bit was inserted or deleted. This is because the key-stream depends only on the fixed number of the preceding ciphertext bits. Also the error propagation is limited. Such ciphers are capable of re-establishing proper decryption automatically after loss of synchronization, with only a fixed number of plaintext characters unrecoverable. This makes such ciphers suitable for applications where synchronization is difficult to maintain.



*Figure 2.8: Self-Synchronizing Stream Cipher where  $\sigma_i$  is the state and  $k$  is the key. Key-stream  $z_i$  is generated by using key-stream generator function  $g$  i.e.,  $g(\sigma_i, k)$  and  $h$  is the output function which combines plaintext  $m_i$  and key stream  $z_i$  to produce ciphertext  $c_i$ .*

## 2.4 Authentication

Generally the term authentication is widely used in very broad sense. By itself, authentication is the process of determining whether someone or something is, in fact, who or what it is declared to be. Authentication is specific to different security objectives like access control, entity authentication, message authentication, data integrity, non-repudiation and key authentication. In this thesis we will be focusing on the authenticity and integrity of the messages.

Message Authentication also called as data-origin authentication plays a very important role in information security. It allows parties to send messages to each other over a channel in such a way that if the message is modified, then the receiving party should be able to detect this. Message authentication protects the integrity of messages and ensures that the each message arriving to the receiving party is in the same condition that it was sent out with no inserted, missing or modified bits. This property also validates in the case of communication between parties over a noisy channel. Data-origin authentication and integrity goes hand-in-hand, if the data is altered and source cannot be determined then the question of alteration cannot be settled.

### 2.4.1 Cryptographic Hash Functions

Hash functions were introduced in the domain of cryptography in the late seventies to protect the authenticity of information. Even conventional hash functions used in non-crypto computer application help mapping larger domains into smaller ranges. In this section we will only be focusing on cryptographic hash functions and their use in authenticity and integrity of the messages.

Hash functions take a message as input and produce an output referred to as a hash-code, hash-result, hash-value or just hash. In other words, hash function is a function  $h$  which takes input  $p$  of arbitrary finite length and converts it into an output  $h(p)$  of fixed length  $n$ . The basic idea of cryptographic hash functions is that a hash-value serves as a compact representative image (sometimes called an *imprint*, *digital fingerprint*, or *message digest*) of an input string, and can be used as if it was uniquely identifiable with that string.

Hash functions are used for data integrity in conjunction with digital signature schemes, where for several reasons a message is typically hashed first, and then the hash-value, as a representative of the message, is signed in place of the original message.

Hash functions are generally classified into two classes unkeyed and keyed hash functions. In unkeyed hash functions, the input parameter is only a message whereas in keyed hash functions two distinct parameters are used, a message and a secret key. Hash functions are further sub-classified as Message Authentication codes (MACs) and Modification Detection Codes (MDCs) or manipulation detection codes, provides an image or hash of a message [MOV]. MDCs are a subclass of unkeyed hash functions, and further divided into One-Way Hash Functions (OWHF) and Collision Resistant Hash Functions (CRHF). The term collision resistant hash function is preferable over strong one-way hash function, as it explains more clearly the actual property that is satisfied. This term proposed by I. Damgård [DAM, DA1], can be sometimes misleading because collisions do exist but it is hard to find them. An alternate proposed by Y. Zheng [ZHE, Z01] is collision



intractable hash function. The term weak one-way hash function was proposed by R.Merkle [MER] in order to stress the difference with a strong or collision resistant hash function.

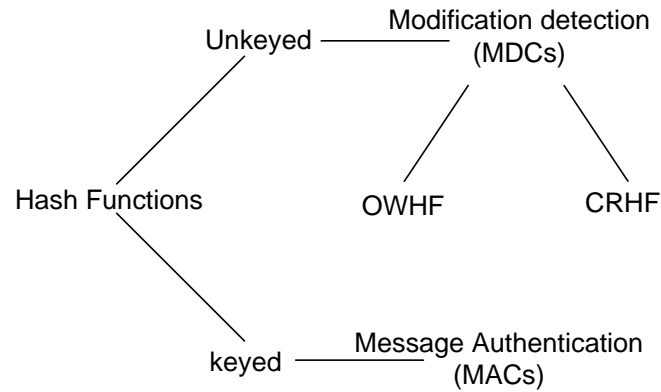


Figure 2.9: Classification of cryptographic hash functions

### **One-way hash function (OWHF)**

The first informal definition was given by R.Merkle [MER,M01] and M.Rabin [RAB]

**Definition 2.3** - A one-way hash function is a function  $h$  satisfying the following conditions:

1. The argument  $X$  can be of arbitrary length and the result  $h(X)$  has a fixed length of  $n$  bits (with  $n \geq 64$ ).
2. Given  $h$  and  $X$ , the computation of  $h(X)$  must be “easy”
3. The hash function must be one-way in the sense that given a  $Y$  in the image of  $h$ , it is “hard” to find a message  $X$  such that  $h(X) = Y$ , and given  $X$  and  $h(X)$  it is “hard” to find a message  $X' \neq X$  such that  $h(X') = h(X)$ .

The first part of the last condition corresponds to the intuitive concept of one-wayness, namely that it is “hard” to find a pre-image of a given value in the range. Pre-image – for all pre-specified outputs, it is computationally infeasible to find any input which hashes to that output. In the case of permutations or injective functions only this concept is relevant. The second part of this condition, is about finding a  $2^{\text{nd}}$  –pre-image should be hard, is a stronger condition that is relevant for most applications.  $2^{\text{nd}}$  – pre-image – it is computationally infeasible to find any second input which has the same output as any specified input. The terms “easy” and “hard” are intentionally left without formal definition; it is intended they should be interpreted relative to an understood frame of reference.

### ***Collision resistant hash functions (CRHF)***

The first formal definition of a CRHF was given by I.Damgård [DAM, D01]. An informal definition was given by R.Merkle [MER].

**Definition 2.4** - A collision resistant hash function is a function  $h$  satisfying the following conditions:

1. The argument  $X$  can be of arbitrary length and the result  $h(X)$  has a fixed length of  $n$  bits (with  $n \geq 128$  )
2. Given  $h$  and  $X$  , the computation of  $h(X)$  must be “easy”
3. The hash function must be one-way in the sense that given a  $Y$  in the image of  $h$ , it is “hard” to find a message  $X$  such that  $h(X) = Y$  and given  $X$  and  $h(X)$  it is “hard” to find a message  $X' \neq X$  such that  $h(X') = h(X)$ .
4. The hash function must be collision resistant: this means that it is “hard” to find two distinct messages that hash to the same result.

As mentioned in [DAM] the first part of one-way property follows from the collision resistant property. Collision resistance – it is computationally infeasible to find any two distinct inputs which hash same output. Formal definition of Collision resistant hash function can be obtained through insertion of a formal definition of terms “hard” and “easy” on combination with the introduction of a security parameter.

### **2.4.2 Message Authentication Codes (MAC)**

Message Authentication codes have been used for a long time for commercial purposes ex., in banking. These codes are used to provide authenticity: did the message received actually originate from the alleged sender? Algorithms used for message authentication allow two parties to send message to each other in such a way that if the message is modified in route to the receiving party then the receiver should be able to detect this. MAC algorithms may be viewed as hash functions which take two functionally distinct inputs, a message and a secret key, and produce a fixed-size (say  $n$  -bit) output, with the design intent that it must be infeasible in practice to produce the same output without knowledge of the key. Unlike digital signatures, MACs are computed and verified with a same key, so that they can only be verified by the intended

recipient.

**Definition 2.5** - [MOV, PRE] A MAC is a function satisfying the following conditions:

1. The argument  $X$  can be of arbitrary length and the result  $h(K, X)$  has a fixed length of  $n$  bits (with  $n \geq 32 \dots 64$ ).
2. Given  $h, X$  and, the computation of  $h(K, X)$  must be “easy”.
3. Given  $h$  and  $X$ , it is “hard” to determine  $h(K, X)$  with a probability of success “significantly higher” than  $1/2^n$ . Even when a large set of pairs  $\{X_i, h(K, X_i)\}$  are known, where the  $X_i$  have been selected by the opponent, it is “hard” to determine the key  $K$  or to compute  $h(K, X')$  for any  $X' \neq X_i$ . This last attack is called an adaptive chosen text attack.

The last condition implies that the MAC should be both pre-image and collision resistant for someone who does not know the secret key  $K$ . But it does not dictate whether MACs need be one-way and collision resistant for parties knowing the key  $K$ .

### 2.4.3 Dedicated Hash functions

In this section we will discuss hash functions specially designed for hashing operations. First part will give an overview of MDC proposals, while in second MAC proposals will be treated. The vast majority of dedicated MDC published up to date is more or less designed using ideas inspired by functions MD4 and MD5. Many functions like HAVAL [Z02], RIPEMD [P01] SHA-0[SHS] and SHA-1[SHS] all exhibit similar resemblance.

R. Rivest of RSA Data Security Inc. has designed a series of hash functions that were named MD for “message digest” followed by a number. MD1 is a proprietary algorithm. MD2 [KAL] was introduced in 1990, to replace BMAC [LIN]. MD3 was never published, and it seems to have been abandoned by its designer. MD4 was also announced in 1990 at Eurocrypt and was published in [RIV]. The four other algorithms MD5, SHA, RIPEMD, and HAVAL are variants on MD4 that were proposed in a later stage. However in 1991, weaknesses in MD4 were initially found and published [DEN] by Den Boer and Bosselaers and in 1995 Hans Dobbertin found the first full round collision attack [DOB].

**MD5 Algorithm** - After looking into the vulnerabilities in MD4 R. Rivest in 1991 proposed a strengthened version of MD4, namely MD5 [RIV01]. MD5 calculates a 128-bit digest for an arbitrary  $b$ -bit message and it consists of the following steps:

- Appending Padding bits: The  $b$ -bit message is padded so that a single 1-bit is added into the end of message. Then, 0-bits are added until the length of the message is congruent to 448, modulo 512.
- Appending length: A 64-bit representation of  $b$  is appended to the result of the padding. Thus, the resulted message is a multiple of 512 bits. This message is denoted here as  $M$ .
- Buffer Initialization: Four 32-bit registers are used in derivation of the 128-bit message digest. The registers are initialized to the following values: x"01234567" x"89abcdef" x"fedcba98" x"76543210".
- Processing of the message: Message  $M$  is divided into 512 bit blocks which are processed separately. The algorithm consists of four rounds, each of which comprises 16 steps. Hence 64 steps are performed in the algorithm. First the initial values  $(A, B, C, D)$  mentioned above are stored into temporary variables  $AA, BB, CC, DD$ . Then, the following operations are performed for  $i = 0$  to 63:

$$A = B + ((A + \text{Func}(B, C, D) + X_j[k] + T[i]) \lll s),$$

$$A \leftarrow D, B \leftarrow A, C \leftarrow B, D \leftarrow C.$$

Where  $X_j$  - denote the  $j^{\text{th}}$  block of  $M$ .  $X_j[k]$  - denote the  $k^{\text{th}}$  32-bit word of  $X_j$ ,  $T[i]$  - table of 64 32-bit constants,  $\lll s$  denote circular shift left by  $s$  bits and additions are addition of words i.e. additions modulo  $-2^{32}$ .

Each round employs one nonlinear round function, which is given below.

$$\begin{aligned} F(X, Y, Z) &= (X \wedge Y) \vee (\neg X \wedge Z), & 0 \leq i \leq 15, \\ F(X, Y, Z) &= (X \wedge Y) \vee (X \wedge \neg Z), & 16 \leq i \leq 31, \\ F(X, Y, Z) &= X \oplus Y \oplus Z, & 32 \leq i \leq 47, \\ F(X, Y, Z) &= Y \oplus (X \vee \neg Z), & 48 \leq i \leq 63, \end{aligned}$$

Where  $X, Y, Z$  are 32 bit words.  $\vee$  is a bitwise or-operation,  $\neg$  is a bitwise complement,  $\oplus$  is a bitwise exclusive-or-operation (xor) and  $\wedge$  is a bitwise and operation. Finally, the values of the temporary variables are added to the values obtained from the algorithm.

$$A = A + AA, B = B + BB, C = C + CC, D = D + DD.$$

However, Den Boer and Bosselaers in 1993 showed in their initial findings a "pseudo-collision" [D01] of the MD5 compression function, that is, two different initialization vectors which produce

an identical digest. This made MD5 not suitable for applications like SSL certificate or digital signatures. Finally in 1996, Dobbertin announced a collision [DOB1] of the compression function of MD5. While this was not an attack on full MD5 hash function, but close enough for cryptographers to recommend switching to the replacement SHA1.

#### 2.4.4 Secure Hash Algorithm

The first Secure Standard (SHS) was proposed in 1992 by U.S. National Institute of Standards and Technology (NIST) that contained description of the secure hash Algorithm (SHA). SHA-1, FIPS 180-3[SHS] was based on MD4, but with better strength. As compared to MD4, the hash value is 160 bits and five instead of four 32-bit chaining variables. The number of steps per round was increased to 20 (16 in case of MD4) and the number of rounds to 4 same as in MD5. By increasing the number of steps in the rounds implied that every word of the chaining variable is transformed 4 times per round. In the compression function, each 16-word message block was expanded to an 80 – word block by XOR-ing of 4 words from earlier positions in the expanded block. These 80 words are then input one-word-per-step to the 80 steps. In the core step, the only rote used is a constant 5-bit rotate; the fifth working variable is added into each step result. The Secure Hash Algorithm (SHA-1)[SHS], was one of the most popular hash functions. Unfortunately, the security level of this standard is limited to a level comparable to an 80-bit block cipher as compared to Advanced Encryption Standard (AES) block cipher which is specified in 128-, 192-, 256- bit keys. This demanded for a new SHA algorithm offering security comparable to the AES key strengths. Looking into the limitations of SHA-1, NIST announced the new Secure Hash Standard 2 (SHA-2), this introduced the specifications of three new Secure Hash Algorithms, SHA-2 (256, 384 and 512).

*SHA-2* – As mentioned above, Standard uses three hash functions SHA-2 (256), SHA-2 (384), SHA-2(512), for computing condensed representation of electronic data. The produced messages digest ranges in length from 256- to 512-bits, depending on the selected hash function. Each hash function operation can be divided into two stages:

*Preprocessing* – Before the computation begins the message is padded into the multiples of 512 or 1024, depending on the algorithm. Then parsing the padded message into  $N$  message blocks  $B^0, B^1, \dots, B^{n-1}$ , where block size is 512 or 1024 bits.

*Hash Computation* – After preprocessing each message block  $B^0, B^1, \dots, B^{n-1}$  is processed in order. For each message block  $B_i$ , starting from message schedule  $W_t$ , following steps (1 -4) are repeated to compute hash values  $H_0^i$  to  $H_7^i$  for the  $i$ th block.

*Step1:*  $W_t$  is computed by identical procedure for SHA-256, SHA-384 and SHA-512, only the logic

function  $\sigma_0$  and  $\sigma_1$  are different. Where  $W_t \in \{0,1\}^n$ ,  $n$  is 32 for SHA-256 and 64 for SHA-512.  $ROTR^k(x)$  – Right rotation of an  $n$ -bit string  $x$  by  $k$  bits and  $SHR^k(x)$  – Right shift of an  $n$ -bit string  $x$  by  $k$  bits.

#### For SHA-256:

Message schedule

$$W_t = B_t^i \quad 0 \leq t \leq 15$$

$$= \sigma_1^{256}(W_{t-2}) + W_{t-7} + \sigma_0^{256}(W_{t-15}) + W_{t-16} \quad 16 \leq t \leq 63$$

Where

$$\sigma_1^{256} = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x)$$

$$\sigma_0^{256} = ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x)$$

#### For SHA-384 & SHA-512:

Message schedule

$$W_t = B_t^i \quad 0 \leq t \leq 15$$

$$= \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16} \quad 16 \leq t \leq 79$$

Where

$$\sigma_1^{512} = ROTR^{19}(x) \oplus ROTR^{61}(x) \oplus SHR^6(x)$$

$$\sigma_0^{512} = ROTR^1(x) \oplus ROTR^8(x) \oplus SHR^7(x)$$

*Step 2:* The hash values  $H_0^i$  to  $H_7^i$ , are assigned to the variables  $a, b, c, d, e, f, g$  and  $h$ . The eight initial hash values are 32 or 64-bit words.

- A sequence of 64 constant 32-bit words,  $K_t^{256}$  or 80 constant 64-bit words,  $K_t^{512}$  is used by the hash processing unit.
- The processing unit uses four logical functions,  $Ch(x, y, z), Maj(x, y, z), \Sigma_0(x), \Sigma_1(x)$ . The logic functions  $Ch$  and  $Maj$  is identical for SHA-256, SHA-384 and SHA-512.

$$Ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z)$$

$$Maj(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$$

#### SHA-256:

$$\Sigma_0(x) = ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x)$$

$$\Sigma_1(x) = ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x)$$

#### SHA-384 & SHA-512:

$$\begin{aligned}\Sigma_0(x) &= ROTR^{28}(x) \oplus ROTR^{34}(x) \oplus ROTR^{39}(x) \\ \Sigma_1(x) &= ROTR^{14}(x) \oplus ROTR^{18}(x) \oplus ROTR^{41}(x)\end{aligned}$$

*Step 3:* The processing unit performs this step, 64 or 80 times on a 512 or 1024 bit block.

$$\begin{aligned}T_1 &= h + \Sigma_1(e) + ch(e, f, g) + K_t + W_t, \\ T_2 &= \Sigma_0(a) + Maj(b, c, d) \\ h &= g, g = f, f = e, e = d + T_1, d = c, c = b, b = a, a = T_1 + T_2\end{aligned}$$

*Step 4:* The  $i$ th intermediate hash value  $H_0^i$  to  $H_7^i$  be computed by modulo-32 or modulo-64 bit adders after the iterations.

$$\begin{aligned}H_0^i &= a + H_0^{i-1} & H_1^i &= b + H_1^{i-1} & H_2^i &= c + H_2^{i-1} & H_3^i &= d + H_3^{i-1} \\ H_4^i &= e + H_4^{i-1} & H_5^i &= f + H_5^{i-1} & H_6^i &= g + H_6^{i-1} & H_7^i &= h + H_7^{i-1}\end{aligned}$$

- The message digest is computed by  $P H_0^n \parallel H_1^n \parallel H_2^n \parallel H_3^n \parallel H_4^n \parallel H_5^n \parallel H_6^n \parallel H_7^n$  after processing all  $N$  blocks in the message.

### **2.4.5 Keyed-Hash Message Authentication Code (HMAC)**

The Keyed-Hash Message Authentication Code standard specifies a mechanism for message authentication using cryptographic hash functions. Hash functions were not originally designed for message authentication. Several constructions were proposed prior to HMAC, but they lacked a convincing security analysis. The HMAC construction intended to fill the gap [BEL]. The performance and the security of the system depend on the underlying hash function and this replaced by other hash functions. HMACs can be proven secure if the underlying hash function has some reasonable cryptographic strength.

As mentioned above HMAC can be used with any iterative hash function, in combination with a shared secret key ( $K$ ). For simplicity of description we may assume that the underlying hash function ( $H$ ) is SHA-1, where the input message is hashed by iterating a basic compression function on blocks of data. With  $B = 60$  is denoted the byte-length of blocks and  $L = 20$  be the

byte-length of the SHA-1 output.  $D$  donates the data to which the MAC function is to be applied. The key  $K$  should not be longer than the size of the hashing block, if shorter, zeros are appended to bring its length to the hashing block. In addition two fixed constants are specified:  $ipad$  and  $opad$ . In order to computer the HMAC over a data block ( $D$ ), the following function is applied:

$$HMAC(K, D) = H\{K \text{ XOR } opad\} \parallel H(K \text{ XOR } ipad \parallel D)\}$$

HMAC function takes the key  $K$  and Data block  $D$  and produces  $HMAC(K, D)$ . Figure 2.10 describes the operation in simple steps.

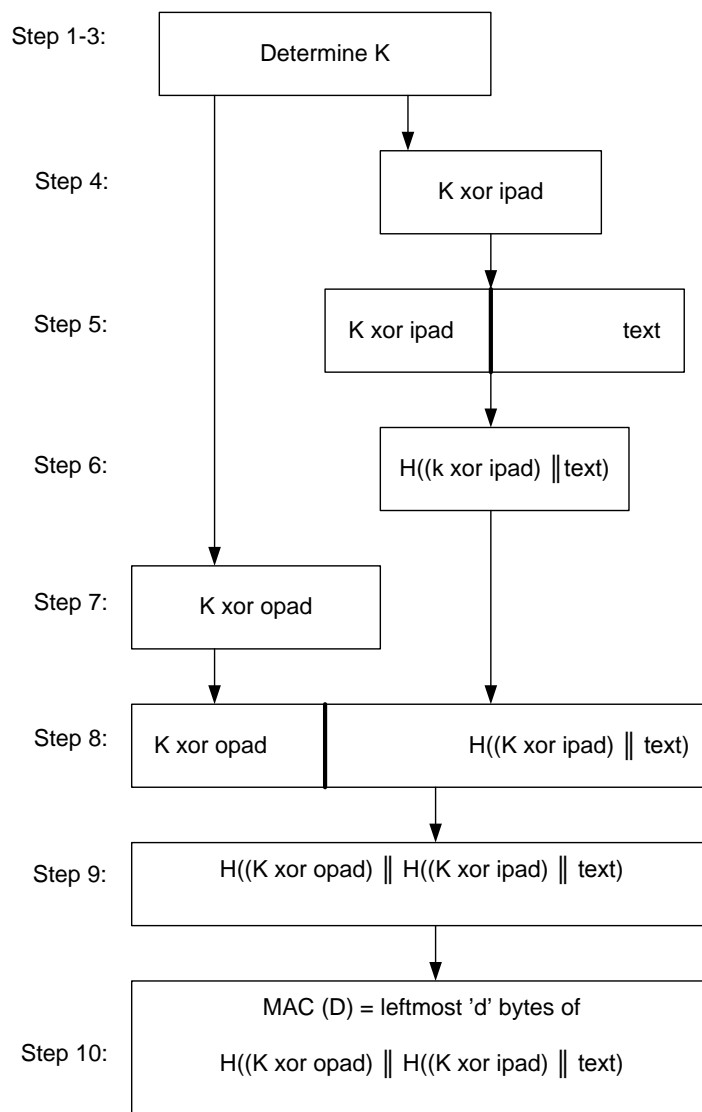


Figure 2.10: HMAC Construction



HMAC was chosen as the mandatory to implement authentication transform for the Internet security protocols being designed by the IPSEC working group of the IETF [BEL1].

## 2.4.6 CBC-MAC

The CBC-MAC [FIP13, ISO97] is an authentication code (MAC) based on a block cipher. Security of CBC-MAC was proved by Bellare, Kilian and Rogaway under the assumption that messages are of one fixed length,  $mn$  bits, where  $n$  is the block length of the underlying block cipher  $E$  [BEL2]. Which means when message lengths vary then CBC-MAC is not secure. Therefore, several variants of CBC-MAC have been proposed for variable lengths. However, first let us look into the basic construction of CBC-MAC with DES (Data Encryption Standard) as shown in figure 2.11, with an underlying block cipher  $E$  and  $n = 64$  and MAC key is a 56-bit DES key.

- Padding and blocking: First pad the data if needed and divide it into  $n$ -bit blocks donated as  $P_1 \dots P_t$
- CBC Processing: Let  $E_K$  donate encryption function using  $E$  with the key  $K$ , compute the hash  $H_t$  as follows:  $H_1 \leftarrow E_K(x_1)$ ;  $H_i \leftarrow (H_{i-1} \oplus x_i), 2 \leq i \leq t$ . Initialization vector  $IV = 0$ .
- Output: The Mac is the  $n$ -bit block  $H_t$

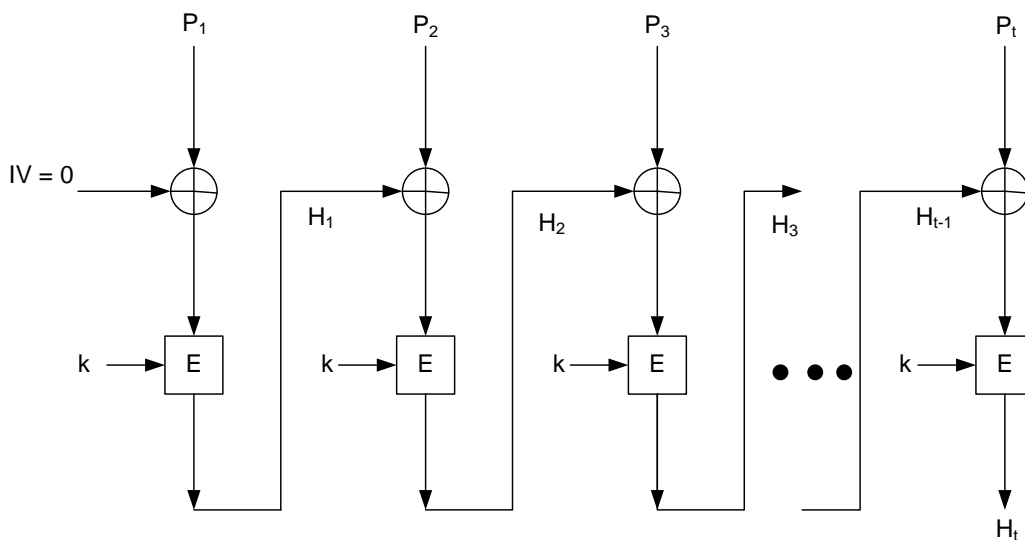


Figure 2.11: CBC-based MAC algorithm

To increase strength of the MAC, First Encrypted MAC (EMAC) was proposed. It is obtained by encrypting the CBC-MAC value by  $E$  again with a new key  $K_2$ .

$$EMAC_{K_1, K_2}(M) = E_{K_2} \left( CBC_{K_1}(M) \right),$$

Where  $M$  is the message,  $K_1$  is the key of the CBC MAC and  $CBC_{K_1}(M)$  is the CBC MAC value of  $M$ . However EMAC was proved to be secure only if the message length is a positive multiple of  $n$  [PET, VAU]. Other variants of CBC-MAC were also proposed for example XCBC by black and Rogaway which requires only one key scheduling of the underlying block cipher  $E$  [BLK]. XCBC takes three keys: one block cipher key and two  $n$ -bit keys. Again the drawback of XCBC is that it requires three keys. Two-key CBC MAC (TMAC) was proposed by Kurosawa and Iwata: a block cipher key  $K_1$  and an  $n$ -bit key  $K_2$ . TMAC [KUR] is obtained from XCBC by replacing  $(K_2, K_3)$  with  $(K_2 \cdot u, K_2)$ , where  $u$  is a non-zero constant and “ $\cdot$ ” denotes multiplication in  $GF(2^n)$ . Finally One-key CBC Mac (OMAC) was proposed by Kurosawa and Iwata [IWA]. This scheme operated in exactly the same way as XCBC except that it only uses a single key  $K$ . A key triple  $(K_1, K_2, K_3)$ , as used is XCBC, is then derived from  $(K, K')$  by setting  $L = e_k(0^n)$ ,  $K_1 = K$ ,  $K_2 = u \cdot L$  and  $K_3 = u^2 \cdot L$ , where  $0^n$  is the  $n$ -bit block of all zeros, and  $u$  is a constant.

Table 2.1 summarizes the above mentioned common hash functions and MAC algorithms.

Name	Year of the Standard	Example
<b>MD4</b>	<b>1990</b>	MD4 was used to compute NT LAN Manager password-derived key digests on Microsoft Windows. MD5 was used in many applications like GPG (GNU Privacy Guard), Kerberos, TLS/SSL etc.
<b>MD5</b>	<b>1992</b>	
<b>SHA-0</b> <b>SHA-1</b>	<b>1993</b>	SHA-1 was developed based on the security flaws of original SHA i.e., SHA-0. SHA-1 was used with the Digital Signature Algorithm (DSA) in electronic mail, bank transfers, software distribution, data storage and many other security applications and protocols, including IPsec, TLS/SSL. Soon after the retirement of SHA-1 because of certain weaknesses, SHA-2 hash functions took over for these applications.
<b>SHA-224</b>	<b>1995</b>	
<b>SHA-256</b>	<b>2004</b>	
<b>SHA-384</b>	<b>2002</b>	
<b>SHA-512</b>	<b>2002</b>	
<b>HMAC</b>	<b>1994</b>	A message authentication code based on above mentioned hash functions. HMAC was developed for the IPsec standard of the internet Engineering Task force (IETF). Currently HMAC is incorporated in SSL/TLS, SSH, Kerberos, IPsec etc.
<b>CBC-MAC</b>	<b>1994</b>	CBC-MAC authentication method is widely used in practice. The construction is secure if the messages are of one fixed length. However, variants of CBC MAC with variable lengths are suggested in the past ex., XCBC, TMAC and OMAC. Most common and widely used example is Counter Cipher Mode with Block Chaining Authentication Code protocol (CCMP), an

---

encryption protocol used for WiFi devices. CCMP protocol is based on AES using the counter mode with CBC-MAC (CCM) mode of operation.

---

*Table 2.1: Hash Functions and MAC Algorithms*

## 2.5 Conclusions

Modern Cryptography is much more than secret information and art. In addition to confidentiality it deals with the problem of data integrity, entity authentication, data origin authentication and much more. In this chapter, we have covered two prime security requirements i.e., confidentiality and authentication of data. It is assumed that the authentication of the source of data provides messages integrity also as a part of the service. We further introduce the concept of symmetric-key encryption and traditional classes of symmetric encryption algorithm – block ciphers and stream ciphers.

Block ciphers are an important primitive in cryptography. A secret key is used to transform a plaintext into a ciphertext in block cipher. The security of block cipher lies within the secrecy of the key, without any knowledge of the secret key, it should be computationally infeasible to recover the original plaintext. Block ciphers are well studied and standardized, and can be used for building other cryptographic primitives, including hash functions, message authentication codes and stream ciphers. Stream ciphers forms the second class of symmetric encryption algorithms. Unlike block ciphers, stream ciphers take a continuous stream of plaintext bits as an input and transform one bit (or a small chunk of bits) at a time. Stream ciphers are in general faster than block cipher and smaller in terms of hardware implementation, even though stream ciphers are broadly proposed and analyzed they have been systematically replaced by block ciphers ex., in 802.11, RC4 stream cipher was replaced by AES block cipher. Another major disadvantage is most of the stream cipher designs are proprietary.

Confidentiality deals with the secrecy of the information whereas authentication is about whether someone or something is, what it is declared to be. Protecting both the security requirements is important but encryption on its own does not provide authenticity. This leads to our second part of the chapter, where Authentication of data is discussed. Cryptographic hash functions are important tools for providing authentication of data and entities. Due to the wide range of applications, hash functions are considered as the “Swiss army knives” of cryptography. Defining the security requirements of the hash functions is not straightforward because they strongly depend on the application and sometimes are used in unexpected ways. For the purpose of data

integrity we have defined the concepts of preimage, second-preimage and collision resistance in section 2.4. We have further explained the classification of cryptographic hash functions as keyed and un-keyed hash functions. Several hash functions are analyzed in section 2.4 e.x., MAC, HMAC and CBC-MAC.

## References

- [MOV] A.Menezes, P.V.Oorschot, and S.Vanstone. Handbook of Applied Cryptography. CRC Press LLC, 1997.
- [CAB] C. Cannière, A. Biryukov, and B. Preneel. An introduction to block cipher cryptanalysis. *Proceedings of the IEEE*, vol 94, no 2, pp. 346 -356, Feb. 2006.
- [STN] D.Stinson. Cryptography – *Theory and Practice*, 3<sup>rd</sup> Chapman & hall / CRC, 2002.
- [BCM] M.Dworkin. "Recommendation for Block Cipher Modes of operation". *NIST Special Publication 800-38A*, 2001
- [DRA] Advanced Encryption Standard (AES), *FIPS Publication 197*, November 26, 2001.
- [EBD] E.Biham, and O. Dunkelman. Cryptanalysis of the a5/1 GSM stream cipher. *In Progress in Cryptology - Indocrypt '00*, B. Roy and E. Okamoto, Eds., vol. 1977 of Lecture Notes in Computer Science, Springer, pp. 43–51. 2000
- [AAD] A.Biryukov, A.Shamir, and D.Wagner. Real time cryptanalysis of a5/1 on a pc. *In Fast Software Encryption - FSE 2001*, M. Matsui, Ed., vol. 1978 of Lecture Notes in Computer Science, Springer, pp. 37 – 44. 2001
- [EAG] ETSI/SAGE. Specification of the A5/3 Encryption Algorithms for GSM and EDGE, and GEA3 Encryption Algorithm for GPRS, Document 1: A5/3 and GEA3 Specifications. ETSI/SAGE, May 2002.
- [VER] G.Vernam. Cipher printing telegraph systems for secret wire and radio telegraphic communications, *J. Am. Institute of Electrical Engineers* Vol. XLV, 109-115, 1926.
- [DAM]I.B. Damgård. Collision free hash functions and public key signature schemes, *Advances in Cryptology, Proc. Euocrypt'87*, LNCS 304, D. Chaum and W.L. Price, Eds., Springer-Verlag, pp. 203–216. 1988.
- [DA1] I.B. Damgård. The application of claw free functions in cryptography. PhD Thesis, Aarhus University, Mathematical Institute, 1988.
- [ZHE]Y. Zheng, T. Matsumoto, and H. Imai. Connections between several versions of one-way hash

functions. *Proc. SCIS90, The 1990 Symposium on Cryptography and Information Security, Nihondaira, Japan*, Jan. 31–Feb.2, 1990.

[Z01] Y. Zheng, T. Matsumoto, and H. Imai. Structural properties of one-way hash functions. *Advances in Cryptology, Proc. Crypto'90, LNCS 537*, S. Vanstone, Ed., Springer-Verlag, 1991, pp. 285–302.

[Z02] Y. Zheng, J. Pieprzyk, and J. Seberry, HAVAL — a one-way hashing algorithm with variable length output, *Advances in Cryptology, Proc. Auscrypt'92, LNCS*, Springer-Verlag, 1992

[MER] R. Merkle. One way hash functions and DES, *Advances in Cryptology, Proc. Crypto'89, LNCS 435*, G. Brassard, Ed., Springer-Verlag, pp. 428–446, 1990.

[M01] R. Merkle, *Secrecy, Authentication, and Public Key Systems*, UMI Research Press, 1979.

[RAB] M.O. Rabin. Digitalized signatures in *Foundations of Secure Computation*. R. Lipton and R. DeMillo, Eds., Academic Press, New York, pp. 155-166, 1978.

[PRE] B. PRENEEL. Analysis and Design of Cryptographic Hash Functions, Ph.D Thesis, Katholieke Universiteit Leuven, January 1993.

[P01] B. Preneel, D. Chaum, W. Fumy, C.J.A. Jansen, P. Landrock, and G. Roelofsen. Race Integrity Primitives Evaluation (RIPE): a status report. *Advances in Cryptology, Proc. Eurocrypt'91, LNCS 547*, D.W. Davies, Ed., Springer-Verlag, pp. 547–551, 1991,.

[SHS] Secure Hash Standard, *Federal Information Processing Standard (FIPS)*, Draft, National Institute of Standards and Technology, US Department of Commerce, Washington D.C., October, 2008

[KAL] B.S. Kaliski. The MD2 Message-Digest algorithm. *Request for Comments (RFC) 1319, Internet Activities Board*, Internet Privacy Task Force, April 1992.

[LIN] J. Linn. Privacy enhancement for Internet electronic mail, Part I: Message encipherment and authentication procedures. *Request for Comments (RFC) 1040, Internet Activities Board*, Internet Privacy Task Force, January 1988.

[RIV] R.L. Rivest. The MD4 message digest algorithm. *Advances in Cryptology, Proc. Crypto'90, LNCS 537*, S. Vanstone, Ed., Springer-Verlag, pp. 303– 311, 1991.

[RIV01] R.L. Rivest. The MD5 message digest algorithm. *RFC 1321*, 1991.

[DEN] B. den Boer and A. Bosselaers. An attack on the last two rounds of MD4. *Advances in Cryptology, Proc. Crypto'91, LNCS 576*, J. Feigenbaum, Ed., Springer- Verlag, pp. 194–203, 1992.

[D01] B. den. Boer and A. Bosselaers. Collisions for the compression function of MD5, *Advances in*

*Cryptology*, Eurocrypt'93 Proceedings, Springer-Verlag, 1994.

[DOB] H.Dobbertin. MD4 is not collision-free. Manuscript, September 1995.

[DOB1]H. Dobbertin. Cryptanalysis of MD5 compress, presented at the rump session of Eurocrypt'96, 1996.

[BEL]M. Bellare, R. Canetti, and H. Krawczyk. Keying Hash Functions for Message Authentication. In *Advances in Cryptology – CRYPTO '96*, edited by Neal Koblitz, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag, Berlin Germany, 1996.

[BEL1] M. Bellare, R. Canetti, and H. Krawczyk. Message Authentication using Hash Functions—The HMAC Construction. *RSA Laboratories CryptoBytes*, 2(1), Spring 1996.

[BEL2] M.Bellare, J.Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. *Advances in Cryptology – CRYPTO '94*, vol. 839 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 341–358, 1994.

[FIP13] FIPS 113. Computer data authentication. Federal Information Processing Standards Publication 113, U.S. Department of Commerce/National Bureau of Standards, National Technical Information Service, Springfield, Virginia, 1994.

[ISO97]Iso/lec 9797-1. Information technology – security techniques – data integrity mechanism using a cryptographic check function employing a block cipher algorithm. International Organization for Standards, Second edition, Geneva, Switzerland, 1999.

[PET] E. Petrank and C. Rackoff. CBC MAC for real-time data sources. *J.Cryptology*, vol. 13, no. 3, pp. 315–338, Springer-Verlag, 2000.

[VAU]S. Vaudenay. Decorrelation over infinite domains: The encrypted CBC-MAC case. *Communications in Information and Systems (CIS)*, vol. 1, pp. 75–85, 2001. Earlier version in *Selected Areas in Cryptography*, SAC 2000, LNCS 2012, pp. 57–71, Springer-Verlag, 2001.

[BLK] J.Black and P.Rogaway. CBC MACs for arbitrary-length messages: The three key construction. *Advances in Cryptology. CRYPTO 2000*, LNCS 1880, pp.197-215, springer-Verlag,2000.

[KUR]K. Kurosawa and T. Iwata. TMAC: Two-Key CBC MAC. *Cryptology ePrint Archive*, Report 2002/092,*RSA Conference 2003, CT-RSA 2003*.

[IWA]T. Iwata and K. Kurosawa. OMAC: One-key CBC MAC. In T. Johansson, editor, *Fast Software Encryption 2003*, volume 2887 of *Lecture Notes in Computer Science*, pages 129-153. Springer-Verlag, Berlin, Germany, Feb. 24-26, 2003.



## Authenticated Encryption

Symmetric key encryption plays vital role in the field of communication, it refers to the schemes and algorithms used to communicate data secretly over an insecure channel between parties sharing a secret key. Often when two parties communicate over a network, they have two main security goals: *confidentiality* and *authentication*. In fact, there is compelling evidence that one should never use encryption without also providing authentication [SBE, JBL]. Confidentiality addresses the issue of keeping the information secret from unauthorized users. Often, this is achieved by encrypting the data using a symmetric-key encryption scheme. Message authentication addresses the issues of source corroboration and improper or unauthorized modification of data. To protect the message authenticity, the sender usually appends an authenticated tag that is generated by the signing (tagging) algorithm of some message authentication scheme.

Although symmetric-key encryption and message authentication have been mainly studied in the separate context, there are many applications where both are needed. The cryptographic schemes that provide both confidentiality and authenticity are called *authenticated encryption schemes*. The authenticated encryption scheme consists of three algorithms: a key generation algorithm, an encryption algorithm and a decryption algorithm. The encryption algorithm takes a key, a plaintext and an initialization vector and it returns a ciphertext. Given the ciphertext and the secret key, the decryption algorithm returns plaintext when the ciphertext is authentic and invalid when the ciphertext is not authentic. The scheme is secure if it is both un-forgeable and secure encryption scheme [MBE]. When an attacker is not able to successfully produce a ciphertext  $C$ , a nonce  $N$ , and a tag  $\sigma$  (three parameters which maintains the integrity of the message) even if the attacker convinces the receiver to will believe that the sender was the originator then the scheme is *Unforgeable*. The term *secure* is related towards confidentiality of the scheme, where confidentiality means, that an attacker cannot understand the contents of the message  $M$ , even after knowing the ciphertext  $C$  and the nonce  $N$ . One way to achieve this is to make the encryption scheme indistinguishable from a random permutation; this is a standard definition that is used in many security proofs such as the security proofs of the modes of operation for block ciphers.



### 3.1 Generic Composition

The generic approach provides authenticated-encryption with associated-data (AEAD) as well as authenticated-encryption (AE). The first generic composition where an encryption scheme and MAC were used jointly (but securely) under independent keys was proposed by Bellare and Namprempré in 2000 [MEB]. However, this was not very efficient generic composition, the time it took to encrypt and authenticate made this block cipher based authenticated encryption twice as slow as either encryption or authentication. In the same year, Katz and Yung presented the single-pass RPC block cipher mode for authenticated encryption [JKA]. RPC could run almost twice as fast as generic authentication scheme but it depends on the size of the authentication tag. In 2001, two block modes of operation for authenticated encryption, IACBC for integrity aware cipher block chaining and the parallelizable mode called IAPM for integrity aware parallelizable mode, supported by a claim of provable security were proposed in [CSJ]. Other provably secure authenticated encryption schemes that use a block cipher as a building block were XCBC, XECB and OCB [VGL, VGL1, PRO]. These schemes were as fast as conventional encryption (without authenticity) i.e. twice as fast as the generic approach with minimal expansion i.e., the size of a ciphertext is same as the size of the plaintext and  $\sigma$  bits of authentication tag, where  $\sigma$  is independent of plaintext size and a constant. The duplex construction [GBE] iteratively applies a bijective transformation, and its main application is authenticated encryption. One can also incorporate some message authentication mechanisms in a stream cipher. The drawback of this approach is that one cannot reduce the security of the scheme to a well-known problem such as the indistinguishability of block ciphers from random permutations. However, this approach promises better efficiency. One such authenticated encryption scheme is Helix [NFE]. Another example of a heuristically designed authenticated encryption scheme is SOBER-128 [PHA].

As mentioned above, the goal of Authenticated encryption is to provide privacy and integrity. Two possible notations are used for the authenticity of AE, INT-PTXT (Integrity of the plaintexts) –  $M = D_K(C)$  was never encrypted by the sender, it is computationally infeasible to produce a ciphertext decrypting to a message that is never encrypted by the sender and INT-CTXT (Integrity of the ciphertexts) –  $C$  was never transmitted by the sender, it is computationally infeasible to produce a ciphertext not previously produced by a sender. Privacy goals for encryption schemes consists of indistinguishability (advantage of a reasonable adversary determining what message was sent,  $M$  or  $M'$ ) and non-malleability (advantage of a reasonable adversary being able to change the message to be meaningful), each of which are considered under either chosen-plaintext or chosen-ciphertext attack. This leads to two indistinguishability notations of security IND-CPA (indistinguishability under a chosen plaintext attack), IND-CCA (indistinguishability under a chosen ciphertext attack) and two non-malleability security notations, namely NM-CPA (non-malleability under a chosen plaintext attacks), NM-CCA (non-malleability under chosen ciphertext attack).

To analyze the security of the Authenticated Encryption Scheme three “generic composition”

methods are considered namely Encrypt-and-MAC, MAC-then-Encrypt, and Encrypt-then-MAC. In each case two different keys ( $K_1, K_2$ ) are used.

### 3.1.1 Encrypt-and-MAC (E&M)

The ciphertext ( $C$ ) is generated by encrypting the plaintext message ( $M$ ) and appending a MAC ( $\sigma$ ) of the plaintext. At decryption side, first ciphertext is decrypted and then authentication tag is verified.

An authenticated encryption scheme ( $AE$ ) = ( $K, E, D$ ) in E&M is defined by

*Encryption Algorithm:*  $E_{K_1 \parallel K_2}(M)$

$C' \leftarrow E'_{k_1}(M)$   
 $\sigma \leftarrow T_{K_2}(M)$   
 Return  $C' \parallel \sigma$

*Decryption Algorithm:*  $D_{K_1 \parallel K_2}(C' \parallel \sigma)$

$M \leftarrow D'_{k_1}(C')$   
 If ( $\sigma = T_{K_2}(M)$ ) then return  $M$   
 Else return  $\perp$

Secure shell (SSH) protocol uses the Encrypt-and-Mac composition [TYL]. SSH is a cryptographically secure replacement for standard Telnet, rlogin, rsh and rch commands. Because of its security properties SSH became incredibly popular as the secure mechanism for access to remote systems interactively. Other programs like rlogin and telnet transmit usernames and passwords in cleartext, sniffing a network were easy, whereas by beginning an encrypted session in SSH before the username and password are transmitted, confidentiality was guaranteed. SSH consists of both a client and a server that use public key cryptography to provide session encryption. The protocol is responsible for encrypting and authenticating all messages between two parties involved an SSH session. To establish a secure connection, client initiates communication by requesting an SSH session. Once the server receives the request, both perform handshake and agree upon a set of shared symmetric keys. The client and the server also agree upon which encryption and message authentication schemes they wish to use. All the recommended encryption and message authentication schemes are based on CBC mode encryption and HMAC [TYL].

Initial formal security analysis and attacks were proposed by Bellare et al. [MEB1] and Dai [WDA]. These attacks were primarily focused on Binary Packet Protocol (BPP) of SSH and assumed that keys had already been securely established. Bellare et al. focused on details such as the mode of operation (SSH's use of CBC mode) and use of initialization vectors (chained IVs to random IVs). Plaintext-recovering attacks were exposed by Albrecht et al. [MRA] based on Bellare's initial security analysis, where they neglected to consider the underlying data structure used by the SSH

BPP in their security proofs. The model was later improved by Paterson and Watson [KGP], to avoid the pitfalls exposed by Albrecht et al.

### 3.1.2 MAC-then-Encrypt (MtE)

The ciphertext ( $C$ ) is generated by appending a MAC ( $\sigma$ ) to the plaintext and then encrypting everything. At the receiving side, decryption is first performed to get the plaintext and the tag, and then tag is verified.

An authenticated encryption scheme  $(AE) = (K, E, D)$  in MtE is defined by

*Encryption Algorithm:*  $E_{K_1 \parallel K_2}(M)$

$\sigma \leftarrow T_{K_2}(M)$   
 $C \leftarrow E'_{K_1}(M \parallel \sigma)$   
 Return  $C$

*Decryption Algorithm :*  $D_{K_1 \parallel K_2}(C)$

$M \parallel \sigma \leftarrow D'_{K_1}(C)$   
 If  $(\sigma = T_{K_2}(M))$  then return  $M$   
 Else return  $\perp$

Secure Socket Layer (SSL) protocol uses Mac-then-Encrypt composition. SSL is a public key based protocol that was developed by Netscape and is the standard Internet protocol for secure communications [PKO]. The secure hypertext transfer protocol (HTTPS) is http using SSL is a communication protocol designed to transfer encrypted information between computers over the World Wide Web. SSL provide a connection between a client and server, over which any amount of data can be sent securely. This layer provides confidentiality, authenticity and replay protection over a connection-oriented reliable transport protocol such as TCP. The SSL protocol supports variety of different cryptographic algorithms to support authenticity between the client and server, confidentiality of the data and establishing session keys. In initial version of SSL (SSL 2.0), most commonly used cipher suite was RSA key exchange (developed by Rivest, Shamir and Adleman). Other algorithms like MD5[RON], SHA-1[SHS] were used for generating Mac and SKIPJACK, Triple-DES for generating ciphertext.

SSL 2.0 suffered from much vulnerability both in cryptographic security and in functionality so the protocol was upgraded to SSL 3.0 with significant enhancements. Security analysis and attacks on SSL 3.0 was proposed by Wagner et al.[DWA] Later in 2001, H Krawczyk [HKR] showed that the generic composition, Mac-then-encrypt method yields a totally insecure protocol when an encryption function that provides perfect secrecy but combined with any MAC function. But

Krawczyk, also showed that MtE could provide secure protocol under two very common forms of encryption: CBC Mode (with an underlying secure block cipher) and stream ciphers (that XOR the data with random or pseudorandom pad).

### 3.1.3 Encrypt-then-Mac (EtM)

The ciphertext ( $C$ ) is generated by encrypting the plaintext ( $M$ ) and then appending a MAC ( $\sigma$ ) of the encrypting plaintext. At the receiving side, authentication tag of the ciphertext is first verified and then decryption is performed to get the plaintext.

An authenticated encryption scheme ( $AE$ ) = ( $K, E, D$ ) in EtM is defined by

*Encryption Algorithm:*  $E_{K_1 \parallel K_2}(M)$

$C' \leftarrow E_{K_1}(M)$   
 $\sigma \leftarrow T_{K_2}(C')$   
 Return  $C' \parallel \sigma$

*Decryption Algorithm :*  $D_{K_1 \parallel K_2}(C' \parallel \sigma)$

$M \leftarrow D'_{K_1}(C')$   
 If ( $\sigma = T_{K_2}(C')$ ) then return  $M$   
 Else return  $\perp$

Out of all three above mentioned generic compositions, Encrypt-then-MAC is provably secure [MBE, JBL]. It guaranties INT-CTXT and IND-CCA which gives secure Encryption and MAC functions (IND-CPA and strongly unforgeable). Internet Protocol Security (IPSEC) uses Encrypt-then-MAC composition[SKE,SKE1]. IPsec was originally developed at the Naval Research Laboratory as part of a DARPA-sponsored research project. The protocol, as defined by IETF is “a framework of open standards for ensuring private, secure communications over Internet Protocol networks, through the use of cryptographic security services” IPsec secures communication by authenticating and encrypting each IP packet. IPsec is not bound to any particular authentication method of secure communications, which is why it is considered an “open standard” The IPsec protocols can be deployed in two basic modes of operation: transport mode and tunnel mode. In transport mode only the data (payload) is encrypted during communication, which gives the advantage of speed—since the IP headers are not encrypted, so the packets are smaller. The disadvantage of transport mode is that an adversary can sniff the network and gather information about end parties. In tunnel mode whole packet, payload and the header is encrypted and treated as a payload for new header called the outer header. The original or inner IP packet is said to be encapsulated within the outer IP header. The advantage is that neither the payload nor any information about end parties can be sniffed and the disadvantage is speed, since the size of the encrypted packet

increases. IPSec provides authenticity, integrity and confidentiality for network layer data through two separate security protocols – Authentication header (AH) and Encapsulating security payload (ESP). AH provides the authenticity and integrity of the data. AH authenticates the packets by signing them, signing packets also provides integrity, since the unique signature prevents the data from being modified. ESP also provides authenticity and integrity, but also adds the advantage of data confidentiality through encryption and usually makes use of block cipher algorithm operating in CBC mode [SKE1, SKE2].

Table 3.1 summarizes the security results of above mentioned composite authenticated encryption schemes.

Generic Composition	Privacy	Integrity	
		INT-PTXT	INT-CTXT
<i>Encrypt-and-MAC</i>	NO	YES	NO
<i>MAC-then-Encrypt</i>	YES	YES	NO
<i>Encrypt-then-MAC</i>	YES	YES	YES

Table 3.1: Security Results in different composite authenticated encryption schemes.

## 3.2 Two pass combined mode

In this section we will discuss two pass combined mode with one pass for encryption and another one for authentication. With the popularity of highly-efficient single pass Authenticated Encryption (discussed in next section), several patents were filed by the authors to cover such schemes. To avoid such patents, two-pass schemes were developed. The first such scheme was CCM (CBC MAC with Counter Mode) [DWH] by Whiting et al. followed by EAX [MEB2], CWC (Carter-Wegman with Counter mode encryption)[TKO].

### 3.2.1 CCM Mode

Counter with Cipher Block Chaining – Message Authentication Code, abbreviated as CCM is an authenticated-encryption mode of operation of block cipher. The underlying block cipher used in CCM is AES -128. CCM is invoked by providing 4 inputs: the key  $k$  used with AES, plaintext data  $M$ , both authenticated and encrypted, associated data  $H$ , authenticated but not encrypted and the nonce  $N$ . CCM processing consists of two pairs of related processes – generation-encryption and decryption-verification. These processes combine encryption using counter mode and authentication using CBC-MAC. Only the encryption function of the block cipher algorithm is used. CCM encryption processes two types of input blocks - Payload block, data must be encrypted and authenticated and associated data block, where data block must be authenticated but not encrypted.

CCM specification – Steps performing CCM processing:

- *Pre-processing of the Message:* Initially the data is arranged in blocks suitable for CCM processing. The first block uniquely determines the nonce  $N$  and the length of the payload. If associated data is present, the block after initial block will consist of associated data until there is no more associated data left.
- *Computing CBC-MAC:* Blocks are encrypted using CBC mode and the output of final encryption is the MAC which can be used for authentication.
- *Encryption using counter mode:* Data is encrypted using Counter (CTR) Mode of operation. The initial input value to the mode is an Initialization Vector (IV) and subsequent values are computed by incrementing the current counter value by one.
- *Processing Counter Mode Encryption:* Initial block generated by counter mode encryption is processed differently than the following blocks.
- *Computing Authentication Tag:* Authentication tag  $\sigma$  is generated by XORing the initial block with the computed CBC-MAC. The length of the tag can be set by the user.
- *Computing the ciphertext:* Finally the cipher is generated by XORing the output of counter mode encryptions with the plaintext blocks. Final ciphertext output is concatenated with the authentication tag i.e.  $C \parallel \sigma$ .

Several problems were encountered in CCM in terms of efficiency, parameterization, complexity, variable-tag-length etc. After examining shortcomings of CCM, Bellare et al. offered new 2-pass authenticated encryption called EAX mode of operation [MEB2]. EAX mode addressed several problems with CCM for example; CCM did not take advantage, if the associated data field was fixed from message to message. In CCM, message lengths must be known in advance because it is encoded into the first block before process begins.

### 3.2.2 EAX Mode

Like CCM, EAX is a combination of a type of CBC MAC and CTR mode encryption. However unlike CCM, EAX mode of operation supports streaming data input and does not require data to be in storage before applying EAX processing. EAX also does not impose any restrictions on the block size of the underlying block cipher and uses only encryption function of the block cipher. EAX processing consists of two invertible processes – Encryption plus authentication tag generation and Decryption plus authentication tag verification. Three input parameters are used for EAX mode of operation –Nonce, Header and Message. Security is related to indistinguishability from random bits and the inability of an attacker to produce a new but a valid triple i.e., {nonce, header, ciphertext}. EAX uses OMAC with an extra input called a “tweak” which allows them to essentially get several different MACs by using distinct values for this tweak input. As shown in figure 3.1, to invoke encryption and authentication tag generation in EAX, a nonce  $N$ , header  $H$  will be authenticated but not encrypted and the message  $M$  will be authenticated and encrypted. Then OMAC is used under key  $k$  (for the chosen block cipher) three times, each time with a different tweak,  $OMAC_k^0$ ,  $OMAC_k^1$  and  $OMAC_k^2$ . First counter value is obtained by computing nonce  $N$  for CTR mode encryption,  $ctr \leftarrow OMAC_k^0(N)$ . Then authentication tag is obtained by computing Header,  $\sigma_H \leftarrow OMAC_k^1(H)$ . Then Message  $M$  is encrypted and authenticated,  $C \leftarrow OMAC_k^2(CTR_k^{ctr}(M))$ . Finally the output is an authentication tag  $\sigma = (ctr \oplus C \oplus \sigma_H)$  and  $(N, H, C)$ . The decryption and verification process is quite similar and straightforward.

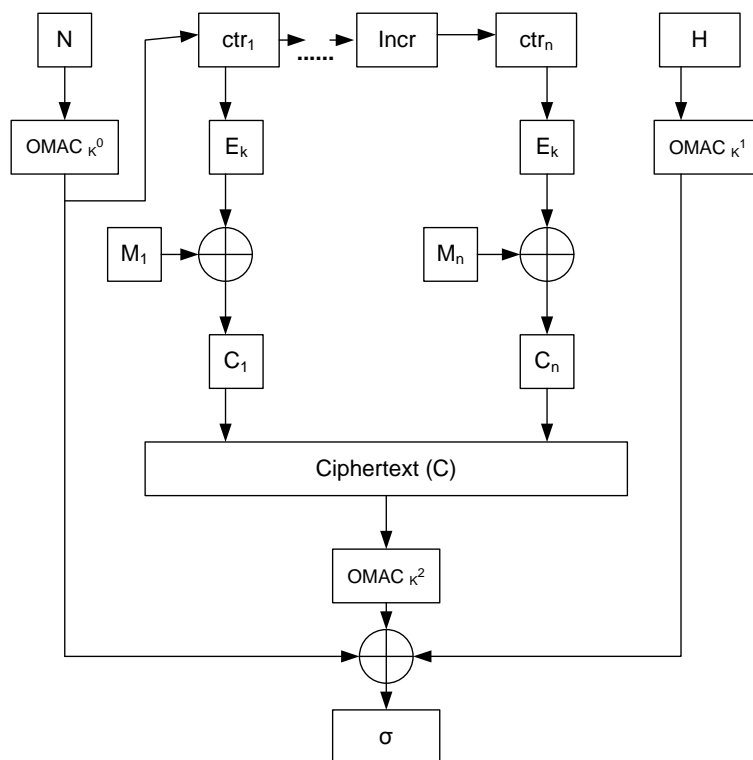


Figure 3.1: EAX mode

Despite of all the problems addressed by EAX mode of operation ex., complex parameterization, handling online data etc still proving security in EXA is difficult since the key  $k$  is re-used in several contexts which are not a safe practice.

### 3.3 Single Pass Combined Modes

The goal of single pass Authenticated Encryption is to achieve faster encryption and message authentication by performing both the encryption and message authentication in a single pass as opposed to the traditional encrypt-then-mac approach, which requires two passes. Several *single-pass* minimal-expanding Authenticated encryption schemes have been proposed: IAPM[CJU] was proposed by Julta in 2000, immediately after announcement of IACBC and IAPM, Gligor proposed two classes of schemes: XCBC and XECB, XCBC was similar to CBC mode encryption just as IACBC and XECB was similar to ECB mode just as IAPM method[VGL]. Rogaway et al. also announced their scheme: OCB, which was similar to IAPM but with the additional optimizations [RRO]. These combine minimal expansion with a close-to-optimal running time: for large messages, these schemes are almost as fast as conventional encryption (without authenticity), i.e. twice as fast as



the generic approach. Unfortunately, several patents cover the usage of the fast single-pass schemes.

### 3.3.1 IAPM

Two modes of encryption were introduced by Jutla of IBM in 2000, which were the first correct single-pass Authenticated Encryption modes [CJU]. These two modes were called IACBC (Integrity Aware CBC) and IAPM (Integrity Aware Parallelizable Mode). The first mode was lot more similar to CBC mode of encryption. However more interest was shown in second mode, IAPM because of its advantage over IACBC. As mentioned IACBC resembles CBC mode, where one cannot begin computation for the  $n$ th block-cipher until one has the results of previous  $n - 1$  block.

IAPM was the first provable secure mode for authenticated encryption. It requires two independent keys  $k_0, k_1$  with the same length as an underlying block cipher, Message  $M$  and nonce  $N$ , as shown in figure 3.2. The mode is divided into two main steps: offset-generation and encryption-tag generation. For offset-generation, a unique nonce  $N$  is used to generate an “offsets”, pair wise differentially uniform vectors (a sequence of uniformly distributed  $n$ -bit random numbers,  $S_1, S_2, \dots, S_m$ ). This generation requires a one block cipher invocation using a key  $k_0$ , an integer additions over the Galois field using  $n$ -bit prime  $p(GF_p)$  respectively only by the xor operations with a penalty of approximately  $\log_m$  extra block cipher invocations [CJU]. The nonce  $N$  is communicated as part of a ciphertext. For encryption-tag generation, each block of message  $M$  ( $M_1 \dots M_{m-1}$ ) is computed as  $C_i \leftarrow E_{k_1}(M \oplus S_i) \oplus S_i$  for  $1 \leq i \leq m - 1$ . The XOR-ing of  $S_i$  before and after the block-cipher invocation is a technique called “key-whitening” to increase the security of an iterated block cipher. To calculate authentication tag  $\sigma$ ,  $\sigma \leftarrow E_{k_1}(S_m \oplus \{M_1 \oplus M_2 \oplus M_3 \dots \oplus M_{m-1}\}) \oplus S_0$ . Finally output  $(N, C_1, \dots, C_{m-1}, \sigma)$  as the authenticated ciphertext. During the decryption process same offset are generated by using  $k_0$  and encryption process is simply reversed. After decrypting, authenticated tag is generated to ensure it matches with the received tag. If the match is positive then transmission is accepted, and if not transmission is considered as an attempt of forgery.

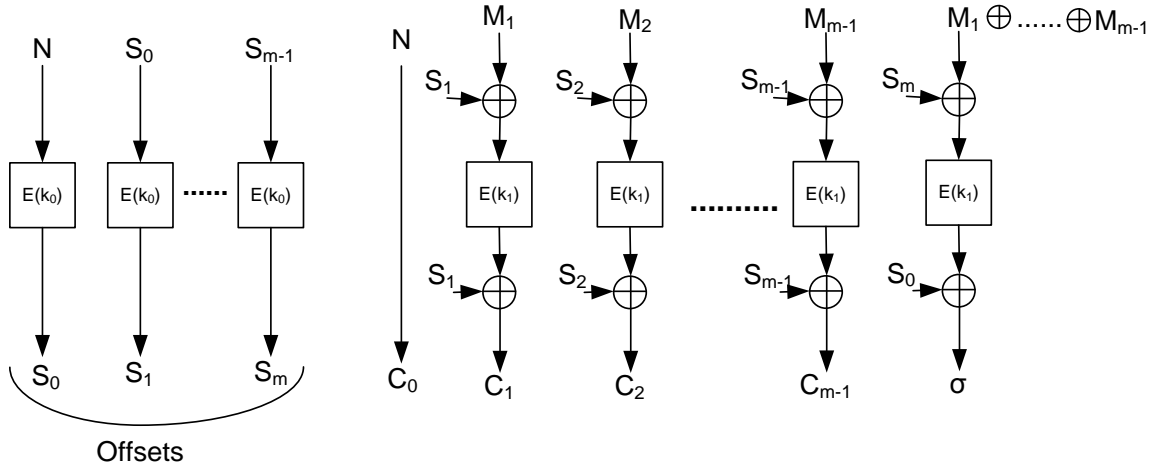


Figure 3.2: Integrity Aware Parallelizable Mode scheme.

### 3.3.2 XCBC

Soon after IACBC and IAPM, Gligor et al. presented two classes of schemes XCBC and XECB[VGL]. As mentioned above, XCBC (Extended Cipher Block Chaining Encryption) is similar to CBC mode encryption. XCBC is proposed in three versions: stateless, stateful-sender and stateful both. In this section we will be considering only stateful-sender version, because of having the same nonce requirement in our proposed system. Initially a set of keys  $(k_1, k_2)$  is used to calculate  $R$  and  $Z_0$  i.e.,  $R \leftarrow E_{k_1}(Ctr)$  and  $Z_0 \leftarrow E_{k_2}(E_{k_1}(Ctr))$ , where  $Z_0$  serves as an IV for CBC-like encryption,  $R$  is used to post-whitening each ciphertext block  $C_i$  by  $i * R$ . To calculate authentication tag  $\sigma$ ,  $\sigma \leftarrow E_{k_1}(Z_0 \oplus \{M_1 \oplus M_2 \oplus M_3 \dots \oplus M_m\}) \oplus (m + 1) * R$ . However XECB, similar to IAPM generates offset to each message block applied before and after a block cipher invocation. Offset are generated in a very efficient manner, using arithmetic mod  $2^n$ , which is very fast on most processors. Both XCBC and XECB is patented and provable secure, AE at a cost very close to that of encryption alone.

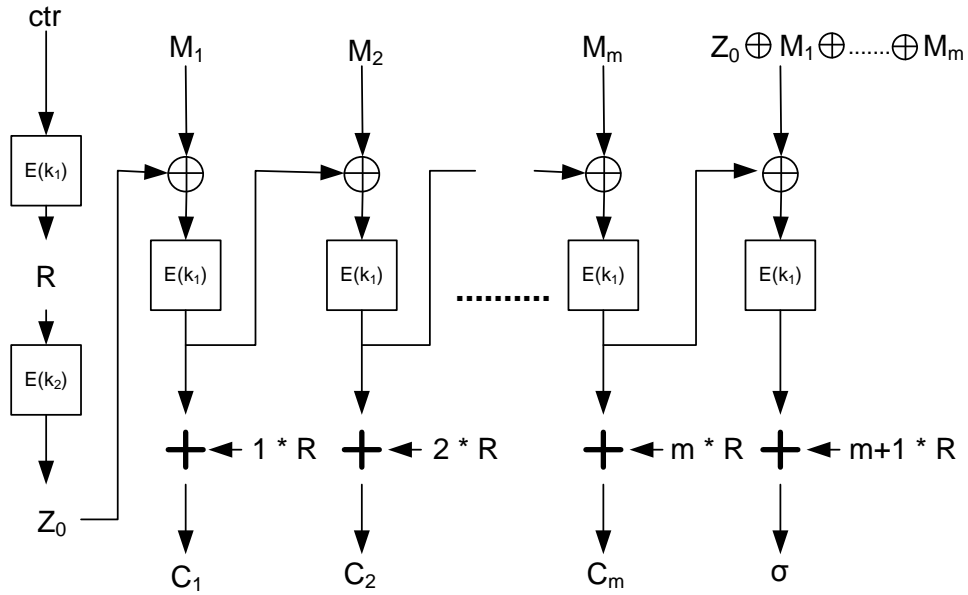


Figure 3.3: Extended Cipher Mode Chaining Encryption mode scheme

### 3.3.3 OCB

Offset Codebook mode (OCB) was presented by Rogaway et al [RRO]. This work was based on Julta's IAPM mode with some new improvements. As shown in figure 3.4, in OCB same key is used for "offset" calculations and encryption. An authentication tag is verified by using an optional tag length  $\tau$  (up to  $n$  bits), depending on an application needs. For every new message, non-repetitive nonce is used. To invoke OCB mode key  $k$  is used to calculate  $L \leftarrow E_k(0^n)$  and for each new nonce  $N$ , one block cipher invocation is used to create the intermediate value  $R \leftarrow E_k(N \oplus L)$ . Using the Gray codes  $\gamma, L$  and  $R$ , the "offsets"  $z_i$  are generated  $z_i \leftarrow \gamma_i \cdot L \oplus R$ . For encryption each block  $M_i$   $1 \leq i \leq m - 1$  is computed as  $C_i \leftarrow E_k(M_i \oplus Z_i) \oplus Z_i$  for  $1 \leq i \leq m - 1$ . To calculate authentication tag  $\sigma$ ,  $\leftarrow E_k(Z_m \oplus \{M_1 \oplus M_2 \oplus M_3 \dots \oplus M_{m-1} \oplus C_m \oplus Y_m\})$  [first  $T$  bits].

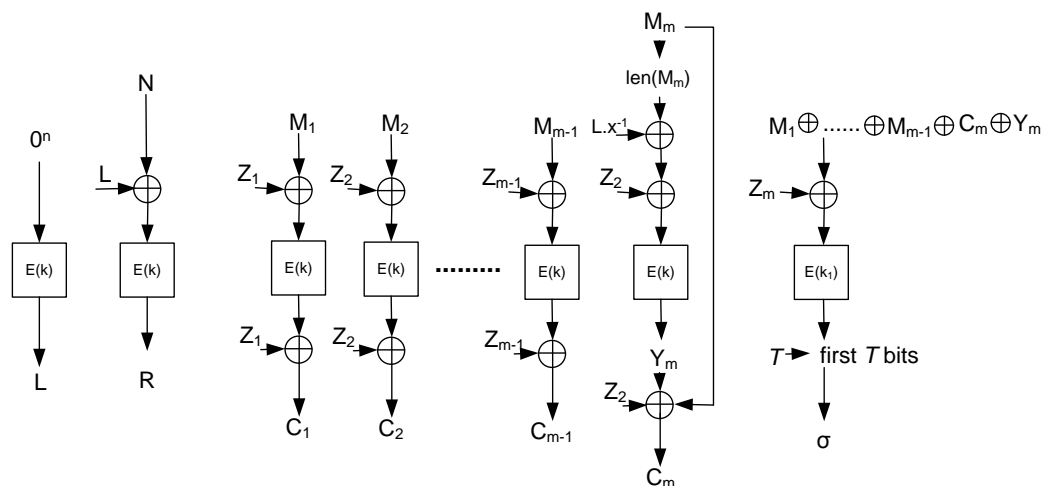


Figure 3.4: Offset Codebook mode scheme

OCB is patented, fully parallelizable, provable secure and very efficient with small requirements with AEAD feature. Performance tests indicate that OCB is about 6.4% slower than CBC mode encryption, and this is without exploiting the parallelism that OCB offers up.

### 3.4 AE Stream Ciphers

Until now, we have looked into the schemes with mode of operation and every mode has used a block cipher as its underlying primitive. In this section we will explore authenticated encryptions mechanisms in a stream ciphers which provide authentication in addition to privacy. The drawback of this approach is that one cannot reduce the security of the scheme to a well-known problem such as the indistinguishability of block ciphers from random permutations. However, this approach promises better efficiency. Two schemes are discussed in this section: Helix [NFE] and SOBER-128[PHA]. Both were designed by experienced cryptographers and close attention was paid towards security and efficiency to the ciphers.

### 3.4.1 Helix

Helix was proposed at FSE'03[NFE] by Ferguson et al. It is an asynchronous stream cipher based on a fast keystream generator. The goal was to produce a fast and patent-free stream cipher with integrity. Helix guarantees the integrity of the message for very little additional computation and without requiring a second pass. Helix is based on an iterated block function applied to an internal state of 160 bits. To invoke the function, input consists of a secret key  $k$  upto 256 bits, a nonce of 128 bits and a Message  $M$ . Before encryption, the internal state of the  $i$ -th word of Message  $M$  is represented in five 32-bit words.  $(Z_0^{(i)}, \dots, Z_4^{(i)})$ , which are initialized for  $i = 0$  using  $k$  and  $N$ . It

uses a block function  $F$  to update the internal state in function of the Message  $M$ , the key  $k$  and nonce  $N$ . More precisely, the  $i$ -th state of Helix emits one 32-bit word of key-stream  $S_i$ , which requires two 32-bit words from the  $k$  and  $N$ , also requires the  $i$ -th Message word  $M_i$ . Using a message stream to generate key-stream is highly-unusual for a stream cipher, but this allows Helix to generate key-stream and authentication tag. To produce ciphertext  $C_i$   $i$ -th key-stream  $S_i$  is XORed with Message  $M_i$ . The 5-word state resulting from block  $i$  is then fed into block  $i + 1$  and this process is repeated until all words of the Message have been encrypted. Finally a last step can generate a tag of 128 bits that constitute the MAC.

In 2004, Muller presented Differential attacks against Helix[MUL]. He showed that the key of Helix can be recovered faster than by brute force if the attacker can force the IV's to be used more than once. The attack requires  $2^{88}$  basic operations and processes only  $2^{12}$  words of chosen plaintext in order to recover the secret key for length upto 256 bits. Later Paul et al. reduced the number of adaptively chosen plaintext words by a factor of at least 3 [SPA].

### 3.4.2 SOBER-128

SOBER-128 was developed from SOBER [PHA], it was proposed by Hawkes and Rose. It is a software-oriented stream cipher based on a linear feedback shift-register (LFSR) over  $GF(2^{32})$ , a non-linear filtering function (NLF) consists of additions modulo  $2^{32}$ , XORing, circular shift and 8 X 32 bit substitution box (S-box). A nonlinear plaintext feedback function (PFF) is added when authentication and encryption are required. To invoke the cipher for authenticated encryption, initially it generates the key-stream and XORs with the Message  $M$  then uses a separate API call "maconly" to process the associated data. Like Helix, SOBER-128 also feedback plaintext into the key-stream generator.

Watanbe and Furnya from Hitachi presented differential cryptanalysis attacks on SOBER-128[DWA]. Their claim was that the MAC generation function in SOBER-128 is vulnerable against differential cryptanalysis and the success probability of this attack is estimated at  $2^{-6}$ .

### 3.5 ASC-1: An Authenticated Encryption Stream Cipher

As mentioned in the previous sections, the goal of a single pass Authenticated Encryption is to achieve faster encryption and message authentication by performing both the encryption and message authentication in a single pass as opposed to the traditional encrypt-then-mac approach, which require two passes. Several single-pass minimal expanding AE schemes have been proposed ex., IACBC and IAPM are two block cipher modes of operation for authenticated encryption supported by a claim of provable security [CSJ]. Other provably secure AE schemes that use a block cipher as a building block were also presented in [VGL] [PRO].

In this thesis, we propose the single pass authenticated encryption scheme ASC-1[SKG]. The design of the scheme has roots in message authentication and encryption scheme that use four rounds of Advanced Encryption Standard (AES) as a building block such as the LEX [ABI] stream cipher, the ALRED[JDE,JDE1] MAC scheme and the MAC schemes proposed in [GJA,KMI]. However, unlike the previous constructions, this scheme uses a single cryptographic primitive to achieve both message secrecy and authenticity. To argue the security of the scheme, it shows that the scheme is secure if one cannot tell apart the case when the scheme uses random round keys from the case when the round keys are derived by a key scheduling algorithm.

For better understanding of ASC-1, we will first look into LEX stream cipher and attacks on LEX stream cipher.

#### 3.5.1 LEX Stream Cipher

Alex Biryukov presented a new methodology of stream cipher design, called leak extraction. The idea is to extract parts of the internal state at certain rounds and give them as the output key stream. The underlying block cipher for LEX is AES block cipher. LEX stream cipher was selected to phase 3 of the eSTREAM competition, the ECRYPT stream cipher project [ABI].

LEX is based on AES and it uses AES in a natural way. The key-stream bits are generated by extracting 4 bytes from the intermediate state of AES in a 128-bit Output Feedback (OFB) mode. As shown in the figure 3.5, in the initialization step, the publicly known 128-bit Initialization Vector (IV) is encrypted by AES under a secret key  $k$  to get  $S = AES_k(IV)$ . The  $S$  and subkeys are the output of the initialization process.

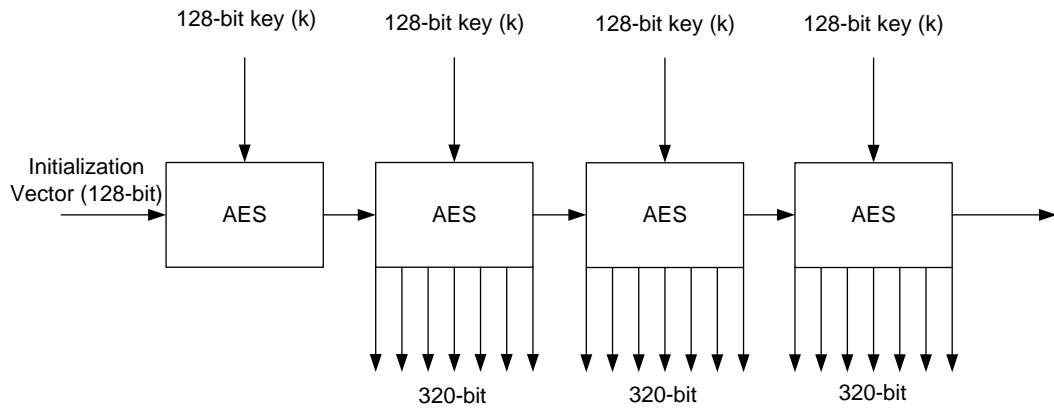


Figure 3.5: LEX Stream Cipher

$S$  is repeatedly encrypted in the OFB mode under  $K$ , where during the execution of each encryption, four bytes are leaked from each round. Another IV is chosen after every 500 encryptions and after  $2^{32}$  IVs, the secret key is replaced.

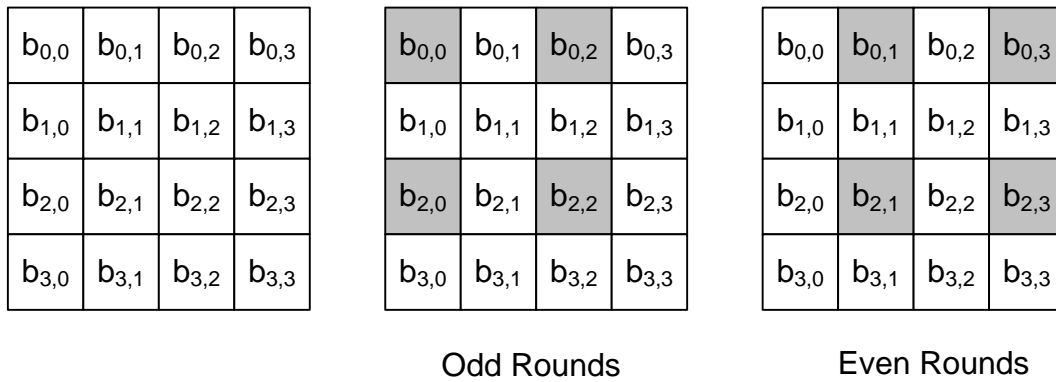


Figure 3.6: Leak Positions in odd and even rounds

Now the most crucial part is the extraction of bytes from the exact location and frequency of the outputs from the intermediate rounds. The designer of the system suggested to extract bytes  $b_{0,1}, b_{2,1}, b_{0,3}, b_{2,3}$  at every odd round and  $b_{0,0}, b_{2,0}, b_{0,2}, b_{2,2}$  at every even round, as shown in figure 3.6. LEX encryption round consists of:

```

Round (State, i)
{
    SubBytes (State);

    ShiftRows(State);

    MixColumns(State);

    AddRoundKey(State, ExpandedKey[i mod  $N_r$ ]);
}

```

$N_r$  is the number of rounds and is equal to 10 for 128-bit key AES. The full  $T$  iterations of LEX would then look like:

```

LEX (State, SecretKey)
{
    AESKeyExpansion(SecretKey, ExpandedKey);

    State = AESEncrypt(IV, ExpandedKey);

    AddroundKey (State, Expandedkey[0]);

    for (i = 1; i < T; i++)
    {
        Round (State, i);

        Output [i] = LeakExtract (State, i mod 2);
    }
}

```

The speed of the cipher is about 2.5 times faster than 128-bit AES. However a key recovery attack was presented by Dunkelman et al, where the attack required about  $2^{36.3}$  bytes of key-stream produced by the same key, and retrieves the secret key in times of  $2^{112}$  simple operations [OD'08]. The attack was divided into three main steps – identification of a special state, extracting information on the special state and finally guess-and-determine attack on the remaining unknown bytes, based on the known bytes an attacker can use this attack to retrieve the key.



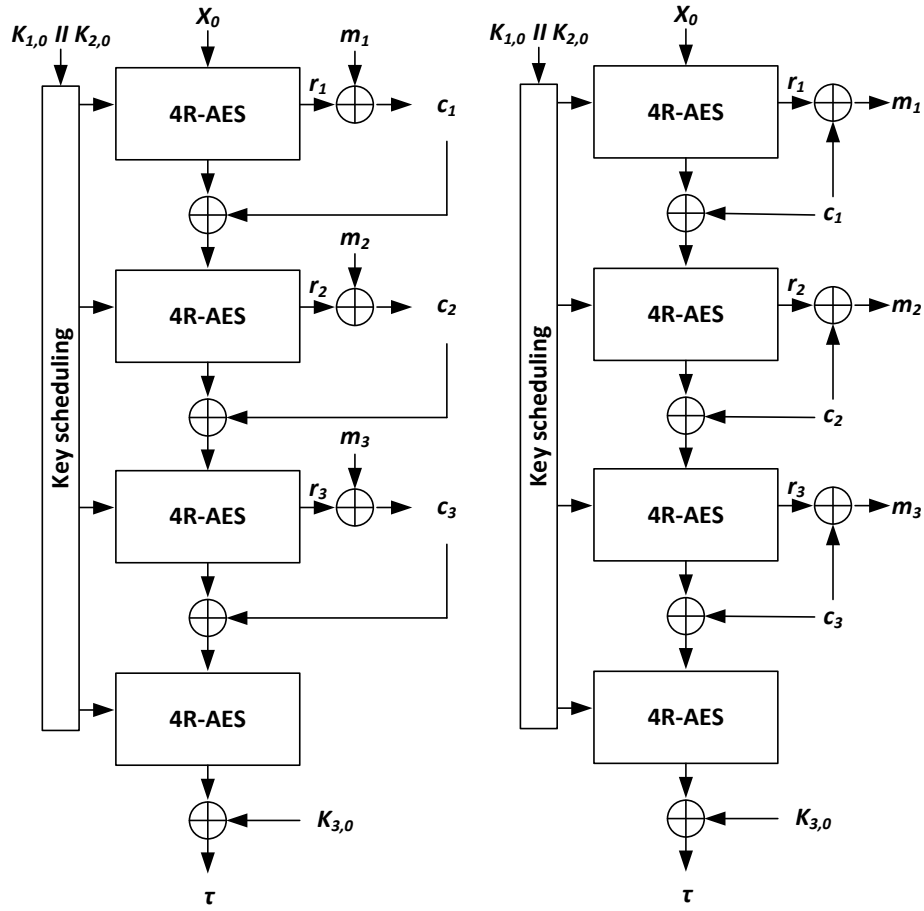
Another attack called “Resynchronization Attack” was presented by Hongjun Wu and Bart Preneel, where the resynchronization of LEX is vulnerable to the slide attacks [HWU]. The attack showed that if the key is used with about  $2^{60.8}$  random IVs and 20,000 keystream bytes are generated from each IV, then the key could easily be recovered.

In general, slide attacks are not dependent of the properties of the iterated round function and the number of rounds as compared to the generic cryptanalytic attacks – differential and linear analysis where for each extra round an exponential effort is required from an attacker. Typically a slide attack exploits the self similarity of a block cipher and sees the cipher as a product of identical transformations [SKM].

### 3.5.1 ASC-1 Specification

ASC-1 is an authenticated encryption scheme. Its key size can vary depending on the block cipher that is used. Block cipher suggestion for ASC-1 is AES with 128-bit key. The encryption and decryption algorithms for a message  $M = m_1 || m_2 || m_3$  consisting of three 128-bit blocks depicted in figure 3.7.

The scheme uses a 56-bit representation of a counter that provides a unique initialization vector for each encrypted message. The encryption algorithm derives an initial state  $X_0$  and three keys  $K_{1,0}$ ,  $K_{2,0}$  and  $K_{3,0}$  by applying block cipher to  $0^{70} || 00 || Cntr$ ,  $0^{70} || 01 || Cntr$ ,  $0^{70} || 10 || Cntr$  and  $l(M) || 00000011 || Cntr$  respectively, where  $l(M)$  is a 64-bit representation of the bit length of the message  $M$ . The message is then processed in a CFB-like mode using the 4R-AES transformation. The 4R-AES transformation takes as input a 128-bit input state and outputs a 128-bit “random” leak  $r_i$  and a 128-bit output state. The first leak  $r_1$  is used to encrypt the first message block  $m_1$ . The resulting ciphertext block  $c_1$  is XOR-ed with the output state to give the input state for the second 4R-AES transformation. This process is repeated for all message blocks. The leak from the last 4R-AES application is ignored, and its output  $h$  is encrypted by  $K_{3,0}$  to give the authentication tag. The ciphertext consists of the counter value, the ciphertext blocks and the authentication tag.



$$X_0 = E_K(0^{70}||00||Cntr), \quad K_{1,0} = E_K(0^{70}||01||Cntr), \quad K_{2,0} = E_K(0^{70}||10||Cntr), \\ K_{3,0} = E_K(l(M)||0^6||11||Cntr)$$

Figure 3.7: The encryption and decryption algorithms of ASC-1. The message consists of three blocks. The ciphertext consists of the counter value, three ciphertext block and an authentication tag. The receiver recovers the original message and verifies its validity by checking whether the re-computed authentication tag is equal to the received one.

The decryption algorithm uses the same secret key and the received counter value to compute  $X_0$ ,  $K_{1,0}$ ,  $K_{2,0}$  and  $K_{3,0}$ . The leak  $r_1$  derived by applying 4R-AES to  $X_0$  is used to decrypt  $c_1$  into the original message block  $m_1$ . The output of the first 4R-AES is XOR-ed with the first ciphertext block to give the next input state, and the process is repeated until all message blocks are recovered and an authentication tag of the message is computed. If the computed tag is same as the one that was received, then the decrypted message is accepted as valid.

Although, the scheme uses 64-bit and 56-bit representation for the message length and the counter, but both the maximum message length and maximum number of messages to be encrypted is  $2^{48}$ . The message length might not be a multiple of the block length. In this case, the

last message block  $m_n$  with length  $l_n < 128$  is padded with zeros to get a 128-bit block  $m'_n$ . A 128-bit ciphertext block  $c'_n$  is derived as  $c'_n = m'_n \oplus r_n$ , and it is XOR-ed with the  $n$ -th output state to give the  $(n+1)$ -st input state. However, the sender will not transmit  $c'_n$  but  $c_n$ , which consists of the first  $l_n$  bits of  $c'_n$ . This will enable the receiver to recover the message length.

The 4R-AES transformation is depicted in figure 3.8. Four AES rounds are applied to the initial state  $x = (x_1, \dots, x_{16})$  to give a 128-bit leak  $r = l_{1..4} \parallel l_{5..8} \parallel l_{9..12} \parallel l_{13..16}$  and an output state

$y = (y_1, \dots, y_{16})$ . Here, we assume that the key addition is the first operation of the AES rounds. Four bytes are leaked after the MixColumns transformation in each round. The leak positions are same as in LEX. However, unlike LEX, key whitening is added before each extracted byte. This gives additional security to the scheme.

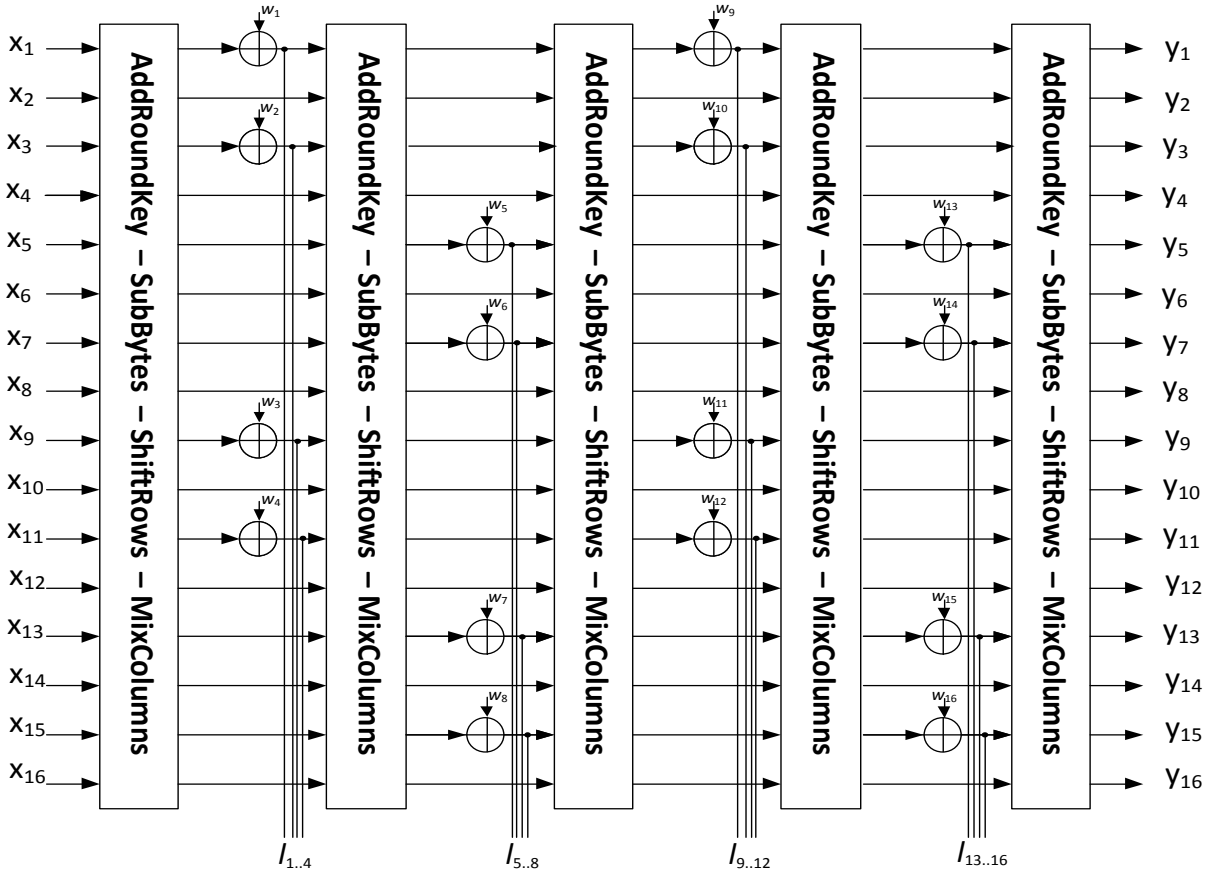


Figure 3.8: The 4R-AES transformation

The 4R-AES transformation uses five 128-bit keys: four round keys and one whitening key. These keys are derived from the 256-bit key  $K_{1,0} \parallel K_{2,0}$  as follows.

The AES-256 key scheduling algorithm is applied to  $K_{1,0} || K_{2,0}$  to derive 14 round keys  $K_1, K_2, \dots, K_{14}$ . The keys  $K_2, K_3, K_4$  and  $K_5$  are used as round keys in the first 4R-AES transformation. The keys  $K_7, K_8, K_9$  and  $K_{10}$  are used as round keys in the second 4R-AES transformation. The key  $K_1$  is used as whitening key in the second 4R-AES transformation and the key  $K_{11}$  is used as whitening key in the first 4R-AES transformation. The AES-256 key scheduling algorithm is again applied to  $K_{13} || K_{14}$  to derive 14 keys that are used by the third and the fourth 4R-AES transformation, and the process is repeated as long as we need new keys.

### 3.6 Security Considerations

In the previous section we have proposed a single pass authenticated encryption scheme ASC-1, with the goal to enable two parties to securely communicate over an insecure channel. The security of the ciphers is continuously evaluated by cryptanalysts all over the world in order to examine the resistance of the designs towards different kind of attacks. Defining the security of the system is quite difficult because an attacker may have different goals and abilities. Before trying to prove the security of the system, we will first list approaches to solve most problems comprising confidentiality and authenticity, and an overview of typical attack scenarios.

#### 3.6.1 Security Measurements

In modern cryptography, many approaches can be identified to evaluate the security of the cryptosystem in the literature. These approaches are based on different assumptions about the capabilities of an attacker. However we will consider two different approaches in this section: Information theoretic approach and Computational security

##### Information Theoretic Approach

The approach is based on information theory and it offers unconditional security. A cryptosystem is unconditionally secure if it cannot be broken even with infinite computational resources. Nevertheless, it should be stated that the unconditional security of the cryptosystem is only probabilistic; a system could be secure in one scenario but easy to break in another. In 1950's a classic definition was proposed by Claude Shannon [SH'49] for "perfect secrecy" of an encryption scheme.

**Definition 3.1** (Perfect secrecy)

A cryptosystem has perfect secrecy if

$$\Pr[P = p | C = c] = \Pr[P = p]$$

for all  $p \in P$  and  $c \in C$ .

In other words, the plaintext is independent of the ciphertext. Another interesting thing to note here is the definitions of unconditional security and perfect secrecy is not equivalent, a cryptosystem that is perfectly secure is unconditionally secure against a ciphertext only attack but not necessarily unconditionally secure against any other attacks. Perfect secrecy is a strong condition, Shannon proved that the perfect secrecy could only be achieved if the length of the secret key is same or exceeds the length of the plaintext. This is usually taken as evidence that unconditional security can never be practical but achievable. Shannon proved this condition by presenting one-time pad cipher.

**Definition 3.2** (One-time pad)

Let  $n > 1$  be the length of the message, then  $P = C = K = Z_2^n$ . For  $x = (x_1, \dots, x_n) \in P$ ,  $y = (y_1, \dots, y_n) \in C$  and  $k = (k_1, \dots, k_n) \in K$  the encryption  $E_k(x)$  and decryption  $D_k(y)$  is defined as bitwise exclusive-or.

$$E_k(x) = (x_1 \oplus k_1, \dots, x_n \oplus k_n)$$

and

$$D_k(y) = (y_1 \oplus k_1, \dots, y_n \oplus k_n)$$

It is essential for the security of one-time pad that the key  $k$  is chosen uniformly at random and no two messages are encrypted using the same key. In the section we saw that the requirements for unconditional security are rather impractical. So, in practice it is safe to assume that an attacker does not have infinite computational resources. This leads to our second approach of computational security.

**Computational Security**

Computational security is the modern approach and it is based on computational complexity. It discards the assumption that an attacker has unlimited computational resources and assumes that the attacker's computational power is limited in some reasonable way. In this approach the main question is not if there is plaintext information present on the ciphertext, but rather if the information can be efficiently extracted. The security in this approach is based on a gap between efficient algorithms guaranteed for the legitimate user versus the computational infeasibility of

retrieving information for an attacker [SGM].

For the sake of an argument, if one has at possibility to verify that the solution is correct, an exhaustive search method could be applied. In this method all the possible elements in the space are tried until the correct one is found.

**Definition 3.3** (Computational Security)

*A cryptosystem provides  $n$  bits of security if an attack requires a computational effort which is equivalent to an exhaustive search over  $2^n$  values.*

From the definition we can deduce that a cryptosystem is computationally secure if it provides  $n$  bit security where  $2^n$  operations are computationally infeasible with the present resources available or that will be in near future. The value of  $n$  is generally related to the length of the key used in the cryptosystem, this is because given the plaintext it is possible to exhaustively search all the possible  $2^n$  keys until the correct one is found. Hence the length of the key gives away the upper bound for the parameter  $n$  and in the ideal case also the lower bound, where no other attacks are faster than exhaustive key search.

Shannon identified two basic techniques for obscuring the redundancies in the plaintext messages for any cryptographic algorithm – Confusion and Diffusion

**Confusion** – obscures the relationship between the plaintext and ciphertext. According to Shannon “confusion is to make the relation between the simple statistics of ciphertext and the simple description of key a very complex and involved one”[SH’49]. This frustrates attempts to study the cipher text looking for redundancies and statistical patterns. The easiest to realize this is by substitutions such as S-boxes.

**Diffusion** – dissipates the redundancy of the plaintext by spreading it out over the cipher text. According to Shannon “The statistical structure of the plaintext which leads to its redundancy is dissipated into long range statistics, i.e., into statistical structure involving long combinations of letters in the cryptogram”[SH’49]. This means that a cryptanalyst looking for those redundancies will have a harder time finding them. Diffusion can easily be caused through transposition.

For designing an encryption scheme, the main goal of a designer is to maximize the complexity of the known attacks and minimize the complexity of the cryptosystem. In our proposed design, we have looked into the vulnerabilities and attacks presented on similar cryptosystems like LEX stream cipher and tried to overcome those attacks[OD’08][HB’06].

Looking from attackers prospective, the goal of an attacker is to find an attack with the complexity lower than the bound estimated by the designer. But not every successful attack faster than exhaustive search makes the cipher useless because it might still be computationally infeasible. On the other hand, every attack discloses some unknown vulnerabilities of the system and carries the risk of becoming a potential threat in the future.

## Attack Scenarios

One can view block ciphers as a family of permutations, indexed by a key. The strongest property from it is to be indistinguishable from completely random permutations, when the key is selected uniformly at random. For the classification of following attack scenarios we assume that the attacker have all the details of the cryptosystem except for the secret key. In addition to this we also assume that the attacker can access the full communication between the sender and receiver.

**Known plaintext attacks (KPA)** – The attacker have no control over the plaintexts but have access to plaintext and the corresponding ciphertext. A priori, they are assumed to be uniformly distributed.

**Ciphertext only attacks (COA)** – The attacker possesses certain amount of ciphertext, without the corresponding plaintext. For this attack to work, all the messages have to be encrypted with the same key.

**Chosen plaintext attacks (CPA)** – The attacker chooses a set of plaintexts a priori and receives the corresponding ciphertext.

**Adaptive chosen plaintext attacks (ACPA)** – The attacker make encryption queries, choosing subsequent plaintext depending on the ciphertext he received from previous requests.

**(Adaptive)Chosen ciphertext attacks (A/CCA)** – The attacks are similar to chosen plaintext and adaptive chosen plaintext attacks. The attacker chooses set of ciphertexts and obtains its decryption.

A cryptosystem vulnerable to ciphertext only attacks are certainly considered very weak. But attacks like known plaintext are still very realistic, however there are easy strategies to slow down these attacks in real world applications. Generally, a symmetric cryptosystem that is secure against adaptive chosen plaintext attack might be vulnerable to chosen ciphertext attacks but secure against rest of the above mentioned attacks. In our proposed system, we argue that ASC-1 is secure by reducing its (IND-CCA, INT-CTXT) security to the problem of distinguishing the case when the round keys are uniformly random from the case when the round keys are generated by a key scheduling algorithms.

## 3.7 Preliminaries

In this section we will draw a notion of an almost universal (AU) and almost XOR universal (AXU) hash function from Carter and Wegman [LC'79][MV'81].

**Definition 3.4. (AU & AXU)** - A family of hash functions  $\mathcal{H} = \{h : A \rightarrow \{0,1\}^b\}$  is  $\varepsilon$ -almost universal<sub>2</sub>, written  $\varepsilon$ -AU<sub>2</sub>, if for all distinct  $x, x' \in A$ ,  $Pr_{h \in \mathcal{H}}[h(x) = h(x')] \leq \varepsilon$ . The family of hash functions  $\mathcal{H}$  is  $\varepsilon$ -almost XOR universal<sub>2</sub>, written  $\varepsilon$ -AXU<sub>2</sub>, if for all distinct  $x, x' \in A$ , and for all  $c \in \{0,1\}^b$ ,  $Pr_{h \in \mathcal{H}}[h(x) \oplus h(x') = c] \leq \varepsilon$ .

The value of  $\varepsilon = \max_{x \neq x'} \{Pr_h[h(x) = h(x')]\}$  is called collision probability. The important measures worth noticing are how small its collision probability is and how fast one can compute its functions. Based on this we will introduce the concept of leak-safe almost XOR universal (LAXU) hash function. Figure 3.9 shows how these functions can be used as a building block to construct an unconditionally secure authenticated encryption scheme.

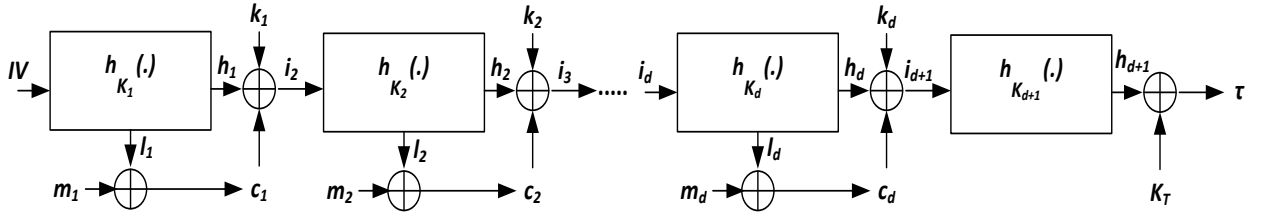
**Definition 3.5. (LAXU)** - A family of hash functions  $\mathcal{H} = \{h(m) = (l, h) | m \in M, l \in \{0,1\}^k, h \in \{0,1\}^n\}$  is leak safe  $\varepsilon$ -almost XOR universal<sub>2</sub>, written  $\varepsilon$ -LAXU<sub>2</sub>, if for all distinct messages  $m, m' \in M$ , for all leaks  $l \in \{0,1\}^k$  and any constant  $c \in \{0,1\}^n$ ,

$$Pr_{h \in \mathcal{H}} [\pi_h(h(m)) \oplus \pi_h(h(m')) = c | \pi_l(h(m)) = l] \leq \varepsilon.$$

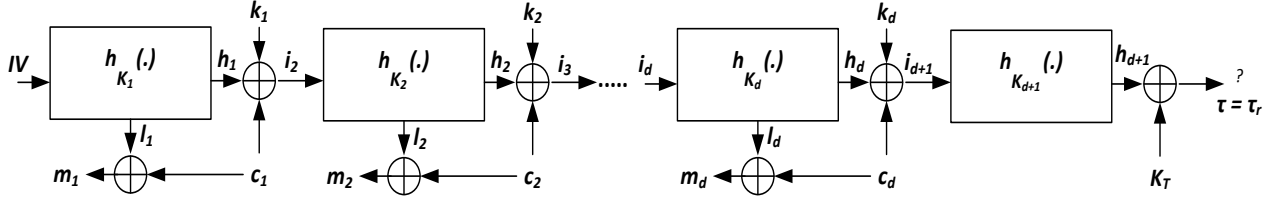
Where  $\pi_h(l, h) = h$  and  $\pi_l(l, h) = l$  are projection functions.

Let the message  $M$  consists of  $d$   $n$ -bit blocks. The ciphertext is computed as follows – a hash function  $h_{K_1}$  is randomly drawn from the family of hash functions  $\mathcal{H}$ . The function is applied to an initial value  $IV$  to get a leak  $l_1$  and hash value  $h_1$ . The leak  $l_1$  is XOR-ed with the message block  $m_1$  and produces the ciphertext  $c_1 = m_1 \oplus l_1$ . To encrypt the second block of message a new hash function  $h_{K_2}$  is randomly pulled from  $\mathcal{H}$  and applied to  $i_2 = k_1 \oplus h_1 \oplus c_1$ , where  $k_1$  is a random key, to get the a leak  $l_2$  and hash value  $h_2$ . Similarly the leak  $l_2$  encrypts the message block  $m_2$  into ciphertext  $c_2$ . This process is repeated until all the  $d$   $n$ -bit blocks of message  $M$  are encrypted. To calculate an authentication tag  $\tau$  for the message  $M$ , a random  $n$ -bit key  $K_T$  is XOR-ed with the hash value  $h_{d+1}$  i.e.,  $= K_T \oplus h_{d+1}$ , where  $h_{d+1}$  is obtained by applying a randomly drawn hash function to  $i_{d+1} = k_d \oplus h_d \oplus c_d$ . Finally the ciphertext  $C = IV || c_1 || c_2 || \dots || c_d || \tau$  is a concatenation of the initial value, the ciphertext blocks and the authentication tag.





Encryption. For message  $M = m_1 | m_2 | \dots | m_d$ , the encryption algorithm outputs a ciphertext  $C = IV | c_1 | c_2 | \dots | c_d | \tau$



Decryption. The algorithm will only output the message  $M = m_1 | m_2 | \dots | m_d$  if the computed tag  $\tau$  and received tag  $\tau_r$  are equal, else rejects the message.

Figure 3.9: An authenticated encryption scheme construction based on a LAXU hash function family on a CFB-like mode.

To decrypt and verify the authenticity of the ciphertext we assume that the recipient has knowledge of the secret keys that were used by the sender to encrypt the message. Similar to the encryption side,  $h_{K_1}$  is applied to the IV to get the leak  $l_1$  and hash value  $h_1$ . The leak  $l_1$  is then XOR-ed with the ciphertext block  $c_1$  and produces the message block  $m_1 = c_1 \oplus l_1$ . To decrypt the next block of ciphertext hash function  $h_{K_2}$  is applied to  $i_2 = k_1 \oplus h_1 \oplus c_1$ , to get the leak  $l_2$  and hash value  $h_2$ . The leak  $l_2$  decrypts the ciphertext block  $c_2$  into plaintext message  $m_2 = c_2 \oplus l_2$ . This process is repeated until all the  $d$   $n$ -bit ciphertext blocks are decrypted. To order to verify the authenticity of the received ciphertext, the recipient re-computes the authentication tag  $\tau = K_T \oplus h_{d+1}$ . If the recomputed tag  $\tau$  is equal to the received tag  $\tau_r$ , then the decryption algorithm returns the plaintext  $M = m_1 || m_2 || \dots || m_d$ . Otherwise the decryption algorithm rejects the ciphertext considering it not authentic. Following theorem establishes the security of the previous theorem.

**Theorem 1.** Suppose that  $\mathcal{H} = \{h(m) = (l, h) | m \in \{0,1\}^n, l \in \{0,1\}^n, h \in \{0,1\}^n\}$  is an  $\varepsilon$ -LAXU<sub>2</sub> family of hash functions such that (i)  $\pi_h(h(m))$  is a bijection, and (ii)  $Pr_{h \in \mathcal{H}} [\pi_l(h(m)) = l | m] = 2^{-n}$  for any message  $m$  and leak  $l$ . Then, the authenticated encryption scheme depicted in figure 3.9 achieves:

1. *Perfect secrecy*. The a posteriori probability that the message is  $M$  given a ciphertext  $C$  is equal to the a priori probability that the message is  $M$ .
2. *Unconditionally secure ciphertext integrity*. The probability that a computationally unbounded adversary will successfully forge a ciphertext is at most  $q_v \epsilon$ , where  $q_v$  is the number of the verification queries that the adversary makes.

*Proof.* From 1, the scheme has perfect secrecy if the initial value  $IV$  is independent of the message and all the leaks  $l_i$  and the key  $K_T$  have uniform probability distribution for any possible message, as in the analysis given below:

$$\begin{aligned}
 & \Pr[M = m_1 \parallel \dots \parallel m_d \mid C = IV \parallel c_1 \parallel \dots \parallel c_d \parallel \tau] = \\
 &= \frac{\Pr[M = m_1 \parallel \dots \parallel m_d] \times \Pr[C = IV \parallel c_1 \parallel \dots \parallel c_d \parallel \tau \mid M = m_1 \parallel \dots \parallel m_d]}{\sum_{M'} \Pr[M' = m'_1 \parallel \dots \parallel m'_d] \times \Pr[C = IV \parallel c_1 \parallel \dots \parallel c_d \parallel \tau \mid M' = m'_1 \parallel \dots \parallel m'_d]} \\
 &= \frac{\Pr[M = m_1 \parallel \dots \parallel m_d] \times 2^{-(d+1)n} \times \Pr[IV]}{\sum_{M'} \Pr[M' = m'_1 \parallel \dots \parallel m'_d] \times 2^{-(d+1)n} \times \Pr[IV]} \\
 &= \Pr[M = m_1 \parallel \dots \parallel m_d]
 \end{aligned}$$

For any message  $M'$ :

$$\begin{aligned}
 & \Pr[C = IV \parallel c_1 \parallel \dots \parallel c_d \parallel \tau \mid M' = m'_1 \parallel \dots \parallel m'_d] = \\
 &= \Pr[c_1 \parallel \dots \parallel c_d \parallel \tau \mid IV, M'] \times \Pr[IV \mid M'] \\
 &= \Pr[1 = m_1 \oplus c_1 \parallel \dots \parallel m_d \oplus c_d, K_T = h_{d+1} \oplus \tau \mid IV, M'] \times \Pr[IV] \\
 &= 2^{-(d+1)n} \times \Pr[IV].
 \end{aligned}$$

### 3.7.1 Classical attacks of cheating

The idea of authenticating a message is to assure receiver that the message is sent by a specified legitimate sender, even in the presence of an attacker who can intercept message and send a fake message to the receiver. When considering the authenticity of the message two classical attacks of cheating are – impersonation and substitution attack.

**Impersonation attack** – The adversary constructs and sends a ciphertext  $C'$  to the receiver before he sees the encryption of the message  $M$ . From the definition above the fact that the key  $K_T$  is uniformly random, the probability of success of an impersonation attack is at most  $2^{-n}$ . If the adversary makes  $q_I$  impersonation attempts, then the probability that at least one of these attempts will be successful is  $1 - (1 - 2^{-n})^{q_I} \leq q_I \times 2^{-n}$ .

**Substitution attack** – In case of substitution attack, the adversary intercepts the ciphertext  $C$  of a given message  $M$  and tried to replace it with a replace with a different ciphertext  $C'$  which he hopes to be accepted by the receiver. Following shows that the probability of success for this attack is at most  $q_s \times \epsilon$ , where  $q_s$  is the number of substitution attempts made by the adversary.

Let us assume that  $C = IV \| c_1 \| c_2 \| \dots \| c_d \| \tau$  is the encrypted text for message  $M$  and let  $C' = IV' \| c'_1 \| c'_2 \| \dots \| c'_d \| \tau'$  be the forged message constructed by the attacker to substitute the original ciphertext. In a case where both the ciphertexts  $C$  and  $C'$  differ only in their authentication tags i.e.,  $\tau' \neq \tau$ ,  $IV' = IV$  and  $c_j = c'_j$ ,  $1 \leq j \leq d$ , then the probability of successful substitution is zero. Now let us consider an interesting case where the forged ciphertext  $C'$  is different from the original ciphertext  $C$  in at least one block that is different from the tag block.

Let  $0 \leq j \leq d$  be the index of the first block where  $C$  and  $C'$  differ, and let  $\Delta i_{j+1} = c'_j \oplus c_j$  be the difference at the input of  $h_{K_{j+1}}$ , with  $c_0 = IV$  and  $c'_0 = IV'$ . Based on leaf-safe almost XOR universal (LAXU) hash function and invertibility properties of  $\mathcal{H}$ ,  $\Pr[\Delta h_{j+1} = 0 \mid M, C, C'] = 0$  and  $\forall_{\Delta \in \{0,1\}^n, \Delta \neq 0} \Pr[\Delta h_{j+1} = \Delta \mid M, C, C'] \leq \epsilon$ . The probability  $\Pr[\Delta h_{j+2} = 0 \mid M, C, C']$  is equal to the probability that  $\Pr[\Delta i_{j+2} = 0 \mid M, C, C']$ , and is at most  $\epsilon$ . When the input different  $\Delta i_{j+2}$  is nonzero, we get that  $\forall_{\Delta \in \{0,1\}^n, \Delta \neq 0} \Pr[\Delta h_{j+2} = \Delta \mid M, C, C'] \leq \epsilon$  and if continue in the similar fashion, we get that  $\forall_{\Delta \in \{0,1\}^n} \Pr[\Delta h_{d+1} = \Delta \mid M, C, C'] \leq \epsilon$ . The forged ciphertext will only be accepted as valid if  $h'_{d+1} \oplus K_T = \tau'$ , i.e., only if  $\Delta h_{d+1} = \Delta \tau$ , where  $\Delta \tau = \tau \oplus \tau'$ . From the previous analysis, this will happen with the probability no larger than  $\epsilon$ .

The probability that an adversary will be able to succeed with at least one substitution query is at most  $q_s \epsilon$  and the probability of success with verification queries i.e.,  $q_V = q_I + q_s$  is at most  $q_V \epsilon$  due to the fact that  $\epsilon \leq 2^{-n}$ .

### 3.8 Security of ASC-1

The security in encryption designs can be classified according to the assumed computational resources of an adversary. Security that holds when one assumes a suitable restriction on an attacker's computing power is called computational security whereas the security that holds even with the unbounded computational capabilities of an attacker is called information-theoretic security. In this section, we will cover both the cases and show if the underlying block cipher in ASC-1 is secure. Additionally, we will also show where one cannot tell apart the case when ASC-1 uses random round keys from the case when it used round keys derived by a key scheduling algorithm, then ASC-1 is secure authenticated encryption scheme.

### 3.8.1 The information-theoretic case

In order to establish the unconditional security of ASC-1, let us consider the case where scheme uses random keys. Figure 3.10, shows the two round Substitution-Permutation Network (SPN) structure. The input  $x = x_1 || \dots || x_n$  is an  $n \times m$ -bit string. The key addition operator is the bitwise XOR operator. The non-linear substitution layer consists of  $n$  S-boxes and each S-box is a non-linear permutation that transforms an  $m$ -bit string into an  $m$ -bit string. The linear mixing layer is defined by a  $n \times n$  matrix. As shown in figure 3.10, the mixing layer is not included in the second round since it does not affect our analysis. The leak  $l$  consists of  $s$  values  $v_1, \dots, v_s$ .

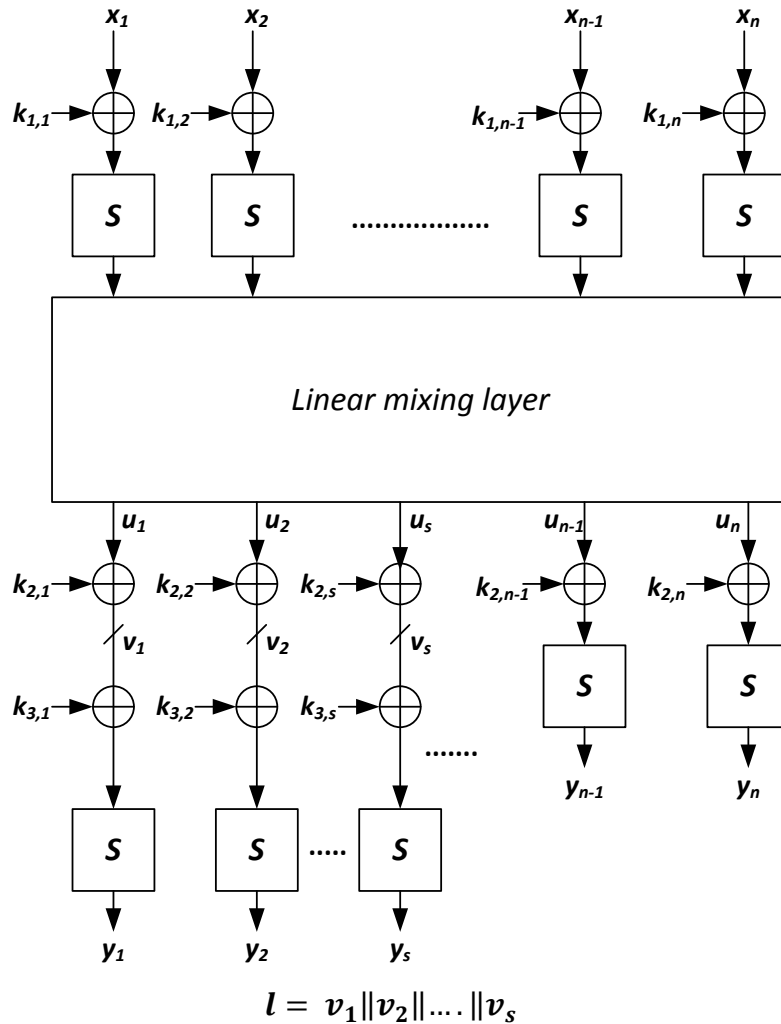


Figure 3.10: A two round SPN structure with a leak

Each possible key  $k_{1,1}, \dots, k_{1,n}, k_{2,1}, \dots, k_{2,n}, k_{3,1}, \dots, k_{3,s}$  defines a function that maps the input  $x$  into an output  $y$  and a leak  $l$ . The collection of such functions  $\mathcal{H}_{2R}$  forms a LAXU hash function family.

**Lemma 1.** Suppose that the keys in the transformation depicted in figure 4 are chosen uniformly at random. Then, we have that

$$\Pr[\Delta y = \Delta y \mid x = x, x' = x', l = l] = \Pr[\Delta y = \Delta y \mid \Delta x = x \oplus x'].$$

*Proof.* Let a function  $h$  is randomly drawn from the pool of functions  $\mathcal{H}_{2R}$ , where  $h$  is the key  $k_{1,1}, \dots, k_{1,n}, k_{2,1}, \dots, k_{2,n}, k_{3,1}, \dots, k_{3,s}$  and let  $l$  be the leak that is acquired by applying  $h$  to  $n$ -bit input string  $x$  and let  $x'$  be an input bit string distinct from  $x$ . The probability  $\Pr[\Delta y = \Delta y \mid x = x, x' = x', l = l]$  is the probability that the output difference  $y \oplus y'$  is  $\Delta y$  given  $x = x$ ,  $x' = x'$  and  $l = l$ . Due to the initial key addition, this probability is equal to the probability  $\Pr[\Delta y = \Delta y \mid \Delta x = x \oplus x', l = l]$  that the output difference is  $\Delta y$  given the input difference  $\Delta x = x \oplus x'$  and the leak  $l$ . To prove this lemma, we use the following observations:

1.  $\Pr[\Delta u \mid \Delta x, l] = \Pr[\Delta u \mid \Delta x]$ , where  $\Delta u = (\Delta u_1, \dots, \Delta u_n)$ ,  $\Delta u_i = u_i \oplus u'_i$ . This observation shows that, given the input difference  $\Delta x$  the output different  $\Delta u$  is independent of the leak  $l$  and this is because of the second key addition which makes the leak uniformly distributed for any possible value  $\Delta u$ .
2.  $\Pr[\Delta y \mid \Delta u, l] = \prod_{i=1}^s \Pr[\Delta y_i \mid \Delta u_i, v_i] \times \prod_{i=s+1}^n \Pr[\Delta y_i \mid \Delta u_i]$ . Given the difference  $\Delta u$ , the probability of having a difference  $\Delta y_i = y_i \oplus y'_i$  at the output of  $i$ -th S-box of the second round is independent of the probability of having a difference  $\Delta y_j, j \neq i$  at the output of some other S-box in the second round.
3.  $\Pr[\Delta y_i \mid \Delta u_i, v_i] = \Pr[\Delta y_i \mid \Delta u_i], i = 1, \dots, s$ . After the leak  $l = v_1, \dots, v_s$ . The intermediate values are bit wise XOR-ed with a third key, this makes the input to the S-boxes uniformly distributed and independent of the  $v_i$  values.

Based on the observations, following shows the proof of the theorem.

$$\begin{aligned}
\Pr[\Delta y|\Delta u, l] &= \sum_{\Delta u} \Pr[\Delta y|\Delta u, \Delta x, l] \Pr[\Delta u|\Delta x, l] \\
&= \sum_{\Delta u} \Pr[\Delta y|\Delta u, l] \Pr[\Delta u|\Delta x] \\
&= \sum_{\Delta u} \Pr[\Delta u|\Delta x] \times \prod_{i=1}^s \Pr[\Delta y_i|\Delta u_i, v_i] \times \prod_{i=s+1}^n \Pr[\Delta y_i|\Delta u_i] \\
&= \sum_{\Delta u} \Pr[\Delta u|\Delta x] \times \prod_{i=1}^s \Pr[\Delta y_i|\Delta u_i] \times \prod_{i=s+1}^n \Pr[\Delta y_i|\Delta u_i] \\
&= \sum_{\Delta u} \Pr[\Delta u|\Delta x] \times \prod_{i=1}^s \Pr[\Delta y_i|\Delta u_i] \\
&= \sum_{\Delta u} \Pr[\Delta u|\Delta x] \times \Pr[\Delta u|\Delta x] \\
&= \Pr[\Delta y|\Delta x]
\end{aligned}$$

**Corollary 1.** *The family of functions  $\mathcal{H}_{2R}$  defined by the 2-round transformation depicted in figure X is  $\varepsilon$ -LAXU<sub>2</sub> with  $\varepsilon = DP_{2R}$ , where  $DP_{2R}$  is the maximum differential probability of the 2-round SPN structure when there is no leak.*

*Proof.* Based on previous lemma, we get that  $\Pr[\Delta y = \Delta y \mid x = x, x' = x', l = l] = \Pr[\Delta y = \Delta y \mid \Delta x = x \oplus x'] \leq DP_{2R}$

The results shown till now are based on two round SPN structure. To show that one can use four AES rounds to construct a LAXU hash function, we will first consider the composition of transformations shown in figure 3.11. In the next lemma we will show the independence of the differential probability of  $F_1$  from the leak value  $l_2$  and vice versa. This could be achieved because of the key addition operation between  $F_1$  and  $F_2$ .

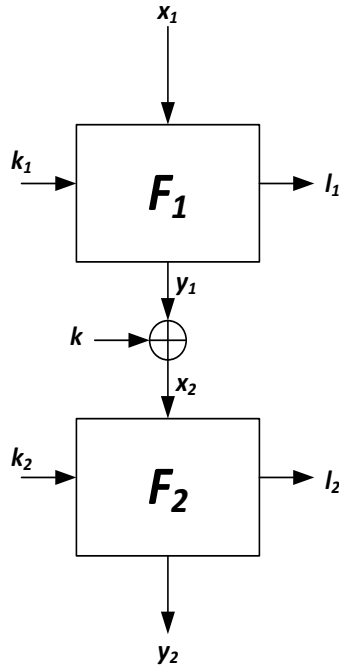


Figure 3.11: A composition of a transformation  $F_1$ , key addition and transformation  $F_2$ .

**Lemma 2.** *The following holds for the differential probabilities of the transformations  $F_1$  and  $F_2$  shown in figure 3.11:*

$$\Pr [\Delta y_1 = \Delta y_1 \mid \Delta x_1 = \Delta x_1, I_1 = l_1, I_2 = l_2] = \Pr [\Delta y_1 = \Delta y_1 \mid \Delta x_1 = \Delta x_1, I_1 = l_1]$$

and

$$\Pr [\Delta y_2 = \Delta y_2 \mid \Delta y_1 = \Delta y_1, I_1 = l_1, I_2 = l_2] = \Pr [\Delta y_2 = \Delta y_2 \mid \Delta y_1 = \Delta y_1, I_2 = l_2]$$

*Proof.*

$$\begin{aligned} & \Pr [\Delta y_1 = \Delta y_1 \mid \Delta x_1 = \Delta x_1, I_1 = l_1, I_2 = l_2] \\ &= \sum_{y_1} \Pr [\Delta y_1 = \Delta y_1, y_1 = y_1 \mid \Delta x_1 = \Delta x_1, I_1 = l_1, I_2 = l_2] \\ &= \sum_{y_1} (\Pr [\Delta y_1 = \Delta y_1 \mid y_1 = y_1, \mid \Delta x_1 = \Delta x_1, I_1 = l_1, I_2 = l_2] \times \\ & \times \Pr [y_1 = y_1, \mid \Delta x_1 = \Delta x_1, I_1 = l_1, I_2 = l_2]) \\ &= \sum_{y_1} (\Pr [\Delta y_1 = \Delta y_1 \mid y_1 = y_1, \mid \Delta x_1 = \Delta x_1, I_1 = l_1] \times \\ & \times \Pr [y_1 = y_1, \mid \Delta x_1 = \Delta x_1, I_1 = l_1]) \\ &= \Pr [\Delta y_1 = \Delta y_1 \mid \Delta x_1 = \Delta x_1, I_1 = l_1]. \end{aligned}$$

Here we used the fact that

$$\begin{aligned}
& \Pr [\Delta y_1 = \Delta y_1 | y_1 = y_1, \Delta x_1 = \Delta x_1, I_1 = l_1, I_2 = l_2] \\
&= \frac{\Pr[\Delta y_1 = \Delta y_1, y_1 = y_1, \Delta x_1 = \Delta x_1, I_1 = l_1, I_2 = l_2]}{\Pr[y_1 = y_1, \Delta x_1 = \Delta x_1, I_1 = l_1, I_2 = l_2]} \\
&= \frac{\Pr[I_2 = l_2 | \Delta y_1 = \Delta y_1, y_1 = y_1, \Delta x_1 = \Delta x_1, I_1 = l_1]}{\Pr[I_2 = l_2 | y_1 = y_1, \Delta x_1 = \Delta x_1, I_1 = l_1]} \times \\
&\times \frac{\Pr[\Delta y_1 = \Delta y_1, y_1 = y_1, \Delta x_1 = \Delta x_1, I_1 = l_1]}{\Pr[y_1 = y_1, \Delta x_1 = \Delta x_1, I_1 = l_1]} \\
&= \frac{\Pr[I_2 = l_2] \times \Pr [\Delta y_1 = \Delta y_1, y_1 = y_1, \Delta x_1 = \Delta x_1, I_1 = l_1]}{\Pr[I_2 = l_2] \times \Pr [y_1 = y_1, \Delta x_1 = \Delta x_1, I_1 = l_1]} \\
&= \Pr[\Delta y_1 = \Delta y_1 | y_1 = y_1, \Delta x_1 = \Delta x_1, I_1 = l_1]
\end{aligned}$$

The equalities  $\Pr[I_2 = l_2 | y_1 = y_1, \Delta x_1 = \Delta x_1, I_1 = l_1] = \Pr[I_2 = l_2]$  and  $\Pr[I_2 = l_2 | \Delta y_1 = \Delta y_1, y_1 = y_1, \Delta x_1 = \Delta x_1, I_1 = l_1] = \Pr[I_2 = l_2]$  shows that the leak  $l_2$  is independent of  $\Delta x_1, y_1, \Delta y_1$  and leak  $l_1$ , this is because the input  $x_2$  is uniformly distributed and independent of these values. Similarly we can shown that

$$\Pr [y_1 = y_1 | \Delta x_1 = \Delta x_1, I_1 = l_1, I_2 = l_2] = \Pr [y_1 = y_1 | \Delta x_1 = \Delta x_1, I_1 = l_1]$$

This proves the first part of the proof. The second part of the lemma can be proved in a similar fashion.

Let us now look into the situation shown in figure 3.12, where we say that the knowledge of the leak values  $I' = (l_1, \dots, l_s)$  does not change the output differential probabilities of the function  $F$ . A keyed non-linear function  $F$  is applied to an input  $x = (x_1 \dots \dots x_n)$  to produce the output  $y = (y_1 \dots \dots y_n)$  of  $n$  output values. Without loss of generality, we assume that the first  $s$  output values are leaked after a uniformly random key is XOR-ed with them.

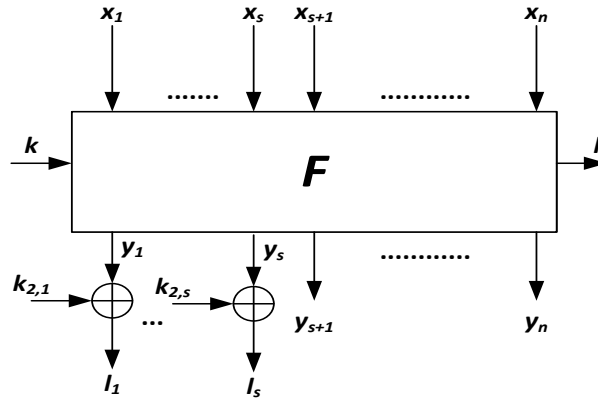


Figure 3.12: The first  $s$  output of a non-linear function  $F$  are “leaked” after a uniformly random key is added



to them

**Lemma 3.** Let  $\mathbf{o} = (l_1, \dots, l_s, y_{s+1}, \dots, y_n)$  denote the output of the transformation shown in figure XX. The following holds for the output differential probability  $\Delta \mathbf{o}$ .

$$\begin{aligned}
& \Pr [\Delta \mathbf{o} (\equiv \Delta y) = \Delta o \mid \Delta x = \Delta x, I = l, I' = l'] \\
&= \sum_y \Pr [\Delta \mathbf{o} = \Delta o, y = y \mid \Delta x = \Delta x, I = l, I' = l'] \\
&= \sum_y \Pr [\Delta \mathbf{o} = \Delta o, y = y \mid \Delta x = \Delta x, I = l, I' = l'] \times \Pr [y = y \mid \Delta x = \Delta x, I = l, I' = l'] \\
&= \sum_y \Pr [\Delta \mathbf{o} = \Delta o, y = y \mid \Delta x = \Delta x, I = l] \times \Pr [y = y \mid \Delta x = \Delta x, I = l] \\
&= \Pr [\Delta \mathbf{o} = \Delta o \mid \Delta x = \Delta x, I = l]
\end{aligned}$$

**Theorem 2.** Suppose that the initial state and all the keys in ASC-1 are uniformly random, then the scheme provides:

- Perfect secrecy and
- Unconditional ciphertext integrity, where the probability of success of any adversary making  $q_V$  verifying queries is at most  $q_V \times 2^{-113}$ .

*Proof.* In this theorem, we show that if the round keys are selected uniformly at random then the family of functions defined by four rounds of AES with leak extraction is an  $\varepsilon$ -LAXU<sub>2</sub> hash function family with  $\varepsilon = 2^{-113}$ . The keys used in the first round of 4R-AES transformation play the same role as key  $k_i$  of the construction shown in figure 3.9. The transformation defined by four rounds of AES is a bijection, and due to the uniform distribution of keys the leaks are uniformly random and independent of the input. Hence we can say that sufficient conditions of Theorem 1 are satisfied and the scheme provides perfect secrecy and unconditional ciphertext integrity.

As mentioned before, in our specification we assume that AddRound key i.e., key addition is the first operation in the round instead of last one as in AES specification. Additionally, all the keys are independent with uniform probability distribution. We use following notation:

- $x_i, i = 0, \dots, 3$  is the input to the  $i$ -th round and consists of 16 bytes  $x_{i,0}, \dots, x_{i,15}$ ;
- $y_i, i = 0, \dots, 3$  is the output of the MixColumns transformation of the  $i$ -th round and consists of 16 bytes  $y_{i,0}, \dots, y_{i,15}$ ;
- $z_i, i = 0, \dots, 3$  is the state after the leak extraction layer of the  $i$ -th round and consists of 16 bytes  $z_{i,0}, \dots, z_{i,15}$ ;

- $I_i, i = 0, \dots, 3$  is the leak extracted in the  $i$ -th round and consists of 4 bytes  $l_{i,0}, \dots, l_{i,15}$ ;

Let  $x'_0$  and  $x''_0$  are two distinct input values, and let us consider the output difference  $\Delta z_3$  given the input difference  $\Delta x_0 = x'_0 \oplus x''_0$ . By applying previous lemmas, we get:

$$\begin{aligned} & \Pr [\Delta z_3 = \Delta z_3 | x'_0 = x'_0, x''_0 = x''_0, I_0 = l_0, I_1 = l_1, I_2 = l_2, I_3 = l_3] \\ &= \Pr [\Delta z_3 = \Delta z_3 | \Delta x_0 = x'_0 \oplus x''_0, I_0 = l_0, I_1 = l_1, I_2 = l_2, I_3 = l_3] \\ &= \sum_{\Delta z_1} (\Pr [\Delta z_1 = \Delta z_1 | \Delta x_0 = x'_0 \oplus x''_0, I_0 = l_0, I_1 = l_1, I_2 = l_2, I_3 = l_3] \times \end{aligned} \quad (1)$$

$$\begin{aligned} & \times \Pr [\Delta z_3 = \Delta z_3 | \Delta z_1 = \Delta z_1, \Delta x_0 = x'_0 \oplus x''_0, I_0 = l_0, I_1 = l_1, I_2 = l_2, I_3 = l_3] \\ &= \sum_{\Delta z_1} (\Pr [\Delta z_1 = \Delta z_1 | \Delta x_0 = x'_0 \oplus x''_0, I_0 = l_0, I_1 = l_1] \times \end{aligned} \quad (2)$$

$$\begin{aligned} & \Pr [\Delta z_3 = \Delta z_3 | \Delta z_1 = \Delta z_1, I_2 = l_2, I_3 = l_3]) \\ &= \sum_{\Delta z_1} (\Pr [\Delta z_1 = \Delta z_1 | \Delta x_0 = x'_0 \oplus x''_0, I_0 = l_0] \times \end{aligned} \quad (3)$$

$$\begin{aligned} & \times \Pr [\Delta z_3 = \Delta z_3 | \Delta z_1 = \Delta z_1, I_2 = l_2] \\ &= \sum_{\Delta z_1} (\Pr [\Delta z_1 = \Delta z_1 | \Delta x_0 = x'_0 \oplus x''_0] \times \end{aligned} \quad (4)$$

$$\begin{aligned} & \Pr [\Delta z_3 = \Delta z_3 | \Delta z_1 = \Delta z_1]) \\ &= \Pr [\Delta z_3 = \Delta z_3 | \Delta x_0 = x'_0 \oplus x''_0] \\ &\leq DP_{4rAES}, \end{aligned}$$

Where  $DP_{4rAES}$  is the differential probability of four rounds of AES with no leak extraction and the rounds keys used in the transformation are random. The equation (2) follows from Lemma 2, the equation (3) follows from Lemma 3, and the equation (4) follows from Lemma 1.

Based on the previous inequality in mind, we get that the family of function defined by four rounds of AES with leak extraction is an  $\varepsilon$ -LAXU<sub>2</sub> hash function family with  $\varepsilon = DP_{4rAES} \leq 2^{-113}$  [LKJ].

### 3.8.2 Computational security analysis of ASC-1

Based on our previous proofs, we are able to show that ASC-1 is unconditionally secure authenticated encryption scheme provided that all the keys and the initial state are random. However, the key and the initial state of ASC-1 are derived using an underlying block cipher (AES) and a key scheduling algorithm. The security of the scheme in this case is based on two assumptions:

- The block cipher (e.g., AES) is indistinguishable from a random permutation, and

- One cannot tell apart the case when the initial state and the keys are random from the case when the initial state  $X_0$  and the tag key  $K_{3,0}$  are random, and the round keys are derived by applying a key scheduling algorithm to the random initial key  $K_{1,0} || K_{2,0}$ .

The first assumption is a standard assumption and it is used in many security proofs such as modes of operations of the block cipher. On the other side, the second assumption is a novel one and should be studied in detail. The second assumption claims that an attacker cannot win the game of distinguishing if the initial state and the keys are random or not. The attacker is given two oracles, an encryption oracle and a decryption oracle. A random coin  $b$  is flipped. If the outcome is zero, then a large table is generated with random strings, the number of strings in the table is equal to the maximum number of messages that can be encrypted. Now when the attacker submits an encryption query, the encryption oracle get the next random string from the table, extracts the initial value and all the rounds keys from the random string and encrypts the message. Similarly, when the attacker submits the decryption query, the decryption oracle gets the random string corresponding to the counter value given in the ciphertext, and uses it to decrypt the ciphertext. On the other hand, if the outcome of coin flipping is one, then the random string in the table consists of four 128-bit random values: an initial start  $X_0$  and three keys  $K_{1,0}$ ,  $K_{2,0}$  and  $K_{3,0}$ . When the attacker submits the encryption query, the encryption oracle uses the next available initial state and keys to encrypt the message following the ASC-1 algorithm. When decryption query is made then the decryption oracle uses the initial state and keys corresponding to the counter value given in the ciphertext to decrypt the ciphertext. The goal of the adversary is to guess the outcome of the coin flipping. The attacker wins if it can guess the value of  $b$  with probability significantly greater than  $1/2$ .

As mentioned in the previous section the design of ASC-1 was inspired by the LEX stream cipher, we are going to address the known attacks on LEX:

- The attack presented by Orr Dunkelman et al. [OD'08] showed that there are special difference patterns that can be observed in the output key stream and these patterns can be used to retrieve the secret key. This attack works under the assumption that single key is used to generate the key stream of at least  $2^{36.3}$  bytes (number of bytes used for successful attack). However, unlike LEX, we do not use the same round keys repeatedly. So, in order for a differential cryptanalysis to work, one has to be able to guess the round key differences. Since these round keys are far apart in the key scheduling process, the above mentioned attack cannot be applied to our scheme.
- The attack presented in [HWU] showed that the LEX stream cipher is vulnerable to slide attack. This attack looked for the repetitions of the state, which can easily be detected

because same states will generate the same pseudo-random key. The round keys in LEX are reused and the state of LEX is 128-bits, it is quite possible to find collisions. Whereas in our scheme, the state is a 384-bit string and it will be very hard to find any collisions. After analyzing, we can carefully deduce that ASC-1 is safe against slide attack.

### 3.9 Conclusions

In this chapter we have introduced the concept of Authenticated Encryption (AE) scheme – a scheme designed for protecting both message’s privacy and its authenticity. In this scheme, the encryption algorithm takes a key and a plaintext and returns a ciphertext. Given the ciphertext and the secret key, the decryption algorithm returns plaintext when the ciphertext is authentic; and invalid when the ciphertext is not authentic. Many solutions have existed for decades for the privacy and authentication problems, and the traditional approach for solving this problem is by combining them in a straightforward manner using so-called “generic composition”. However this is not a very efficient way of achieving both the security goals, the time it takes to encrypt and authenticate make twice as slow as either encryption or authentication. For past few years, there have been number of constructions that can achieve both privacy and authenticity, often almost twice as fast as any solution which uses generic composition.

To analyze the security of the AE schemes three generic composition methods are considered namely Encrypt-and-MAC, MAC-then-Encrypt, and Encrypt-then-MAC. We gave a brief overview of these in section 3.1.1, and show the security comparison in different composition.

With the popularity of highly-efficient one-pass AE schemes, several patents cover the usage of fast single-pass schemes. To avoid this two pass combined mode were developed, where one pass is used for encryption and another one for authentication. In section 3.2, we discussed two pass combined mode followed with the analysis of such schemes ex., CCM (CBC MAC with Counter Mode) and EAX mode.

Section 3.3 gives an overview of one-pass combined mode with few single-pass schemes proposed in the past. As opposed to two pass the goal of single pass authenticated encryption is to achieve faster encryption and message authentication in a single pass. When used for large messages these schemes are as fast as conventional encryption and twice as fast as the generic approach, such schemes were first proposed by Jutla in 2000 and immediately after that Gligor, also proposed two such schemes. AE stream ciphers are discussed in section 3.4. Instead of block ciphers, stream ciphers are used to provide privacy and authentication. Disadvantage of such approach is that one cannot reduce the security of the scheme to a well-known problem such as the indistinguishability of block cipher from random permutation.

In section 3.5, we proposed a single pass AE scheme. The design of the scheme has its roots in message authentication and encryption scheme that uses four rounds of AES as a building block. The scheme is inspired from LEX stream cipher, the ALRED MAC scheme and the MAC schemes proposed in [GJA,KMI]. Section 3.8 discusses the security of the scheme. Many approaches can be identified to evaluate the security of the cryptosystem. In our case we have discussed Information theoretic approach and Computational security of the scheme. We have argued the security of ASC-1 by showing that it is secure if one cannot distinguish the case when the round keys are uniformly random from the case when the round keys are derived by the key scheduling algorithm of ASC-1. Since our proposed design is inspired by the LEX stream cipher, we have also addressed known attacks on LEX and showed that our system is secure against those attacks.

## References

- [MBE] M. Bellare and C. Namprempre, Authenticated Encryption: Relations Among Notions and Analysis of the Generic Composition Paradigm, *In Advances in Cryptology - ASIACRYPT 2000*, LNCS vol. 1976, pp. 531-545, Springer-Verlag, 2000.
- [JKA] J.Katz and M. Yung. Unforgeable encryption and adaptively secure modes of operation. *Fast Software Encryption'00*, LNCS 2001, Vol. 1978/2001, pp.25-36, Springer, 2001
- [CSJ]. C.S. Jutla, Encryption Modes with Almost Free Message Integrity. *Advances in Cryptology - EUROCRYPT 2001*, LNCS vol. 2045, pp. 529{544, Springer, 2001.
- [VGL]. V. Gligor and P. Donescu. Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes. *Presented at the 2nd NIST Workshop on AES Modes of Operation*, Santa Barbara, CA, August 24, 2001.
- [VGL1]. V. Gligor and P. Donescu. Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes, *In the Proceedings of FSE 2002*, LNCS vol. 2355, pp. 1-20, Springer-Verlag, 2002.
- [PRO]. P. Rogaway, M. Bellare, J. Black, and T. Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption, *In the Proceedings of 8th ACM Conf. Comp. and Comm. Security (CCS)*, 2001.
- [GBE]. G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Duplexing the Sponge: Authenticated Encryption and Other Applications. *In the Second SHA-3 Candidate Conference*, 2010.

- [NFE]. N. Ferguson, D. Whiting, B. Schneier, J. Kelsey, S. Lucks, and T. Kohno. Helix: Fast Encryption and Authentication in a Single Cryptographic Primitive. *In the Proceedings of FSE 2003*, LNCS vol. 2887, pp. 330-346, Springer, 2003.
- [PHA]. P. Hawkes and G. Rose. Primitive Specification for SOBER-128, Available from <http://www.qualcomm.com.au/Sober128.html>.
- [SBE] S.Bellovin. Problem areas for the IP security protocols. In *Proceedings of the Sixth USENIX Security Symposium*, pp. 1–16, 1996.
- [JBL ]J. Black. Authenticated encryption. In H. van Tilborg, editor, *Encyclopedia of Cryptography and Security*, pages 11–21. Springer-Verlag, 2005.
- [JBL1] J. Black, and H. Urtubia. Side-channel attacks on symmetric encryption schemes: The case for authenticated encryption. In *Proceedings of the Eleventh USENIX Security Symposium*, D. Boneh, Ed., pp. 327–338, 2002.
- [TYL] T. Ylonen and C. Lonvick. The secure shell (SSH) transport layer protocol. RFC 4253, Jan.2006.
- [WDA] W. Dai. An Attack Against SSH2 Protocol. Email to the SECSH Working Group available from <ftp://ftp.ietf.org/ietf-mail-archive/secsh/2002-02.mail>, 6th Feb. 2002.
- [MEB1] M. Bellare, T. Kohno, and C. Namprempre. Breaking and Provably Repairing the SSH Authenticated Encryption Scheme: A Case Study of the Encode-then-Encrypt-and-MAC Paradigm. *ACM Transactions on Information and Systems Security*, 7(2):206–241, 2004.
- [MRA] M.R.Albrecht, K.G.Paterson, and G.J.Watson. Plaintext Recovery Attacks Against SSH. *In Proceedings of the 30th IEEE Symposium on Security and Privacy*. 2009.
- [KGP]K.G. Paterson and G.J. Watson. Plaintext-Dependent Decryption: A Formal Security Treatment of SSH-CTR.” In H. Gilbert (ed.), EUROCRYPT 2010, LNCS Vol. 6110, Springer, pp. 345-361,2010.
- [PKO] P.Kocker, A.Freier, and P.Karlton. *The SSL Protocol Version 3.0*. Netscape Communications Corp., (Mar. 1996), [home.netscape.com/eng/ssl3/index.html](http://home.netscape.com/eng/ssl3/index.html)
- [SHS]NIST. Secure hash standard, federal information processing standards publication 180-1, April 1995. <http://www.itl.nist.gov/fipspubs/fip180-1.htm>.
- [RON]R. Rivest. The MD5 message digest algorithm, RFC-1321,1992.
- [DWA]D. Wagner and B. Schneier. Analysis of the SSL 3.0 protocol. In *2nd USENIX Workshop on Electronic Commerce*, 1996. Revised version of November 19, 1996 available from <http://www.cs.berkeley.edu/~daw/ssl3.0.ps>.
- [HKR]H. Krawczyk. The order of encryption and authentication for protecting communications (or:

How secure is SSL?). In *Advances in Cryptology CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*. Springer-Verlag Heidelberg, January 2001

[SKE] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol, RFC 2401, November 1998.

[SKE1] S. Kent and R. Atkinson. IP Encapsulating Security Payload (ESP), RFC 2406, Nov. 1998.

[SKE2] S. Kent. IP Encapsulating Security Payload (ESP), RFC 4303 (obsoletes RFC 2406), Dec. 2005.

[DWH] D. Whiting, R. Housley, and N. Ferguson. Counter with CBC-MAC (CCM), RFC 3610, Internet Eng. Task Force, Sept. 2003.

[MEB2] M. Bellare, P. Rogaway, and D. Wagner. EAX : a conventional authentication encryption mode. *FSE 2004*.

[TKO] T. Kohno, J. Viega, and D. Whiting. CWC: A high-performance conventional authenticated encryption mode. In W. Meier and B. Roy, editors, *Fast Software Encryption, FSE 2004*, Springer-Verlag, 2004.

[CJU] C. Jutla. Encryption modes with almost free message integrity. In B. Pfitzmann, editor, *Advances in Cryptology EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 529{544. Springer-Verlag, Berlin, Germany, May 6-10, 2001.

[VGL] V. Gligor, P. Donescu. Extended Cipher Block Chaining Encryption, 2000. Available at <http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/xcbc/xcbc-spec.pdf>. October, 2004.

[RRO] R. Rogaway. OCB Mode: Parallelizable Authenticated Encryption. *ACM Transactions on Information and System Security*. Vol 6, pp 365-403, 2003.

[NFE] N. Ferguson, D. Whiting, and B. Schneier, J. Kelsey, S. Lucks, and T. Kohno. Helix: Fast encryption and authentication in a single cryptographic primitive. In *Fast Software Encryption, 10<sup>th</sup> International Workshop, FSE'03* T. Johansson, Ed., *Lecture Notes in Computer Science*, Springer-Verlag, 2003.

[PHA] P. Hawkes, and G. Rose. Primitive specification for SOBER-128. *Cryptology ePrint Archive Report 2003/48*. April 2003.

[MUL] F. Muller. Differential Attacks against the Helix Stream Cipher. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE'04*, volume 3017 of *Lecture Notes in Computer Science*, pages 94-108. Springer-Verlag, 2004.

[SPA] S. Paul, and B. Preneel : Near Optimal Algorithms for Solving Differential Equations of Addition with Batch Queries. In: Maitra, S., Madhavan, C.E.V., Venkatesan, R. (eds.) *INDOCRYPT 2005*. LNCS, vol. 3797, pp. 90–103. Springer, Heidelberg (2005).

[DWA] D. Watanabe and S. Furuya. A MAC Forgery Attack on SOBER-128. IN B. Roy and W. Meier, Editors, *Fast Software Encryption-2204*, volume 3017 of *Lecture notes in Computer Science* pages 472-482, springer, 2004.

- [SKG]G.Jakimoski and S.Khajuria. ASC-1: An Authenticated Encryption Stream Cipher. Selected Areas in Cryptography (SAC). 18<sup>th</sup> International Workshop, LNCS 7118, Springer, pp. 356-372, 2012.
- [ABI]A. Biryukov. The Design of a Stream Cipher LEX," In the Proceedings of SAC2006, LNCS vol.4536, pp.67-75, Springer, 2007.
- [JDE]J. Daemen and V. Rijmen. A New MAC Construction ALRED and a Specific Instance ALPHA-MAC," In the Proceedings of FSE 2005, LNCS vol. 3557, Springer, pp. 1-17, 2005.
- [JDE1] J. Daemen and V. Rijmen. The Pelican MAC Function," IACR ePrint Archive, 2005.
- [GJA]G. Jakimoski and K.P. Subbalakshmi. On Efficient Message Authentication Via Block Cipher Design Techniques," In Advances in Cryptology - ASIACRYPT 2007, LNCS vol. 4833, pp. 232-248, Springer-Verlag, 2007.
- [HWU]H. Wu, and B. Preneel. Resynchronization Attacks on WG and LEX, *Fast Software Encryption 2006*, LNCS 4047, pp. 422-432, Springer-Verlag,2006.
- [KMI]K. Minematsu and Y. Tsunoo. Provably Secure MACs from Differentially- Uniform Permutations and AES-Based Implementations," In the Proceedings of FSE 2006, LNCS 4047, Springer, pp. 226-241, 2006.
- [SH'49] C.Shannon. Communication theory of secrecy systems. Bell System Technical Journal 28 , 656 – 715, 1949.
- [SGM] S. Goldwasser and M. Bellare, Summer course cryptography and computer security at MIT, 1996–1999, Lecture Notes on Cryptography, August 1999.
- [OD'08]O. Dunkleman and N. Keller. A New Attack on the LEX Stream Cipher, Advances in Cryptology - ASIACRYPT 2008, LNCS vol. 5350, pp. 539-556, Springer,2008.
- [LC'79]L. Carter and M. Wegman. Universal hash functions, *Journal of Computer and System Sciences* 18, pp. 143–154, 1979.
- [MW'81]M.Wegman and L. Carter, New hash functions and their use in authentication and set equality, *Journal of Computer and System Sciences*, vol. 22, pp. 265–279, 1981.
- [LKJ]L. Keliher and J. Sui. Exact Maximum Expected Differential and Linear Probability for 2-Round Advanced Encryption Standard (AES), IACR ePrint Archive, 2005.
- [SKM]S. Kavut and M.D. Yucel. Slide Attack on Spectr-H64. In Alfred Menezes and Palash Sarkar, editors, INDOCRYPT, volume 2551 of LNCS, pages 34-47. Springer, 2002.





## **PART II – Implementation & Results**



## Field Programmable Gate Arrays (FPGAs)

With the development of advanced field-programmable devices the process of designing digital hardware has changed significantly over the past few years. Unlike prior to the discovery of these programmable logic, designers had to use specialized integrated circuits, each of which contained just a few gates known as *discrete logic*. Even to design a reasonably complex device, designers had to mount few tens of these chips on one board. This led to performance issues and more complex board. The first type of user-programmable chip that could implement logic circuits was the Programmable Read-Only Memory (PROM) [SBJ]. However PROMs were considered an inefficient architecture because logic functions rarely need more than a few product terms and a PROM contains a full decoder for its address inputs. So, PROMs were rarely used in practice for that purpose. The first device developed in 1970s by Philips, specifically for implementing logic circuit was the Programmable logic array (PLA). PLAs were one-time programmable chips consisting of two levels of logic gates: a programmable “wired” AND-plane followed by a programmable “wired” OR-plane. Figure 4.1 shows the structure of PLA, its inputs and complements are AND-ed together and connected to OR-plane output.

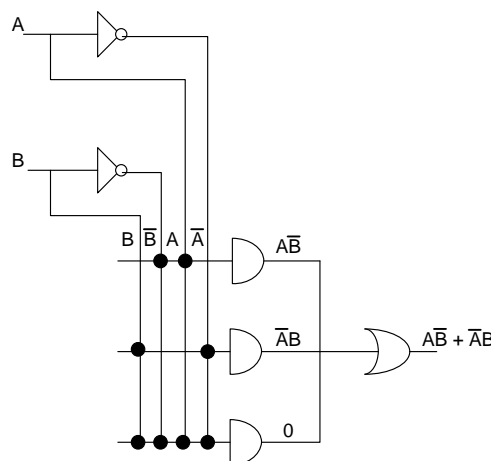


Figure 4.1: Programmable-Logic-Arrays

As a contribution to PLA architecture, Complex Programmable Logic Device (CPLD) chips include

PLD-like (Programmable Logic Device, usually a PLA) blocks also known as macrocells, at the borders of the chip, and a connection matrix located at the central part. CPLD is a more complex PLD that consists of multiple PLDs on a single chip with programmable interconnects. CPLDs are usually Flash-based, that is, the configuration of macrocells and the interconnection matrix is defined by contents of the on-chip flash memory. It means that CPLD need not to be configured after each power-up, unlike the SRAM-based FPGAs [BZE].

## 4.1 FPGA Architecture

Field programmable gate arrays (FPGAs) are classified into three categories: SRAM – based on static memory, Antifuse – one-time programmable and EPROM / EEPROM / Flash based – Erasable Programmable Read-only memory. Here we will be mainly focusing on SRAM-based FPGAs. SRAM-based FPGA is a semiconductor device comprises of an array of programmable logic blocks, surrounded by programmable I/O and connected by a programmable routing matrix. These are programmed by loading appropriate value into SRAM on the chip.

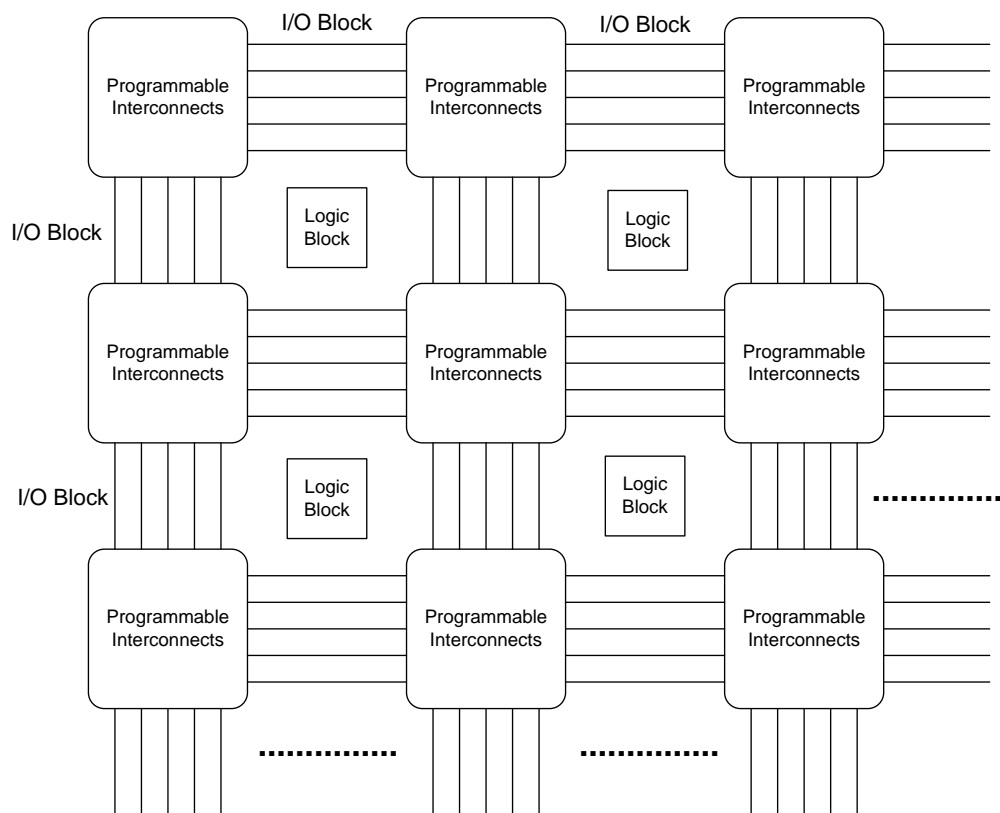


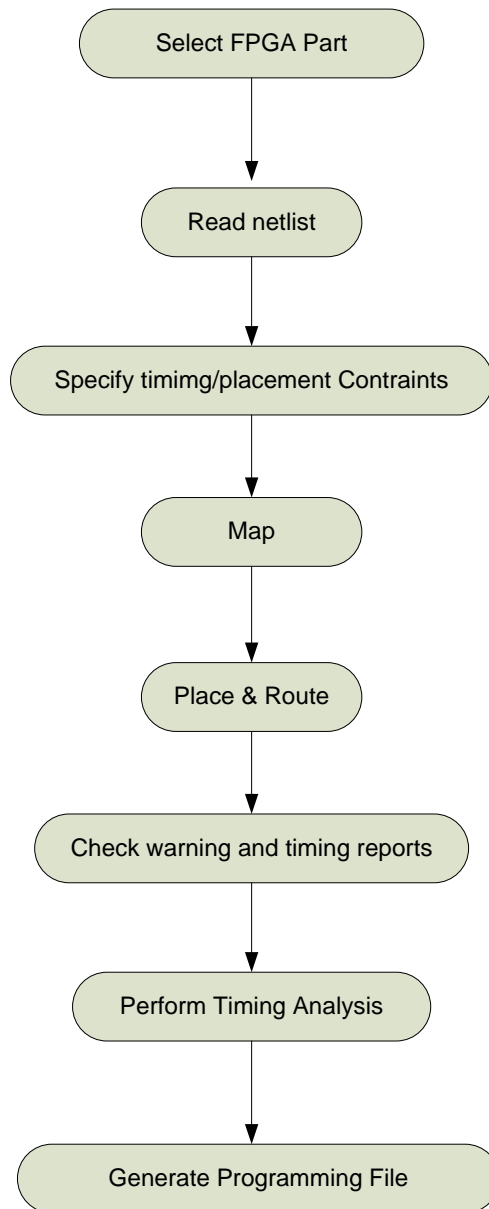
Figure 4.2: Field Programmable Gate Array

As shown in figure 4.2, these programmable logic blocks can perform the function of basic logic gates such as AND or XOR, or more complex combinational functions such as decoders or simple mathematical functions. Logic functions are implemented in small lookup tables called LUT. A logic block consists of a LUT and some flip-flops, connections between LUT and flip-flops are quite fast as the wires within the logic block are quite short. There are also short distance fast connections to and from neighboring logic blocks. These are normally used for regular structures such as adders and counters. Whenever a function larger than LUT need to be implemented, neighboring LUTs can be borrowed, but with the cost of significant overhead in terms of propagation delay (lower speed performance).

A hardware description language (HDL) provided for the user defines the behavior of the FPGA. Common HDLs are VHDL and Verilog. The unit of HLD text is processed by the design tools ex., Xilinx-ISE, a technology-mapped netlist is generated. The netlists are designed according to the specification of the FPGA architecture and fitted to that specific FPGA using a process called place-and-route. Before validating the map, place and route results, user also have the possibility to run different verification methods and simulations of the design. Once the design and validation process is complete, the binary file is generated to (re)configure the FPGA [XLX].

#### **4.1.1 FPGA Implementation Flow**

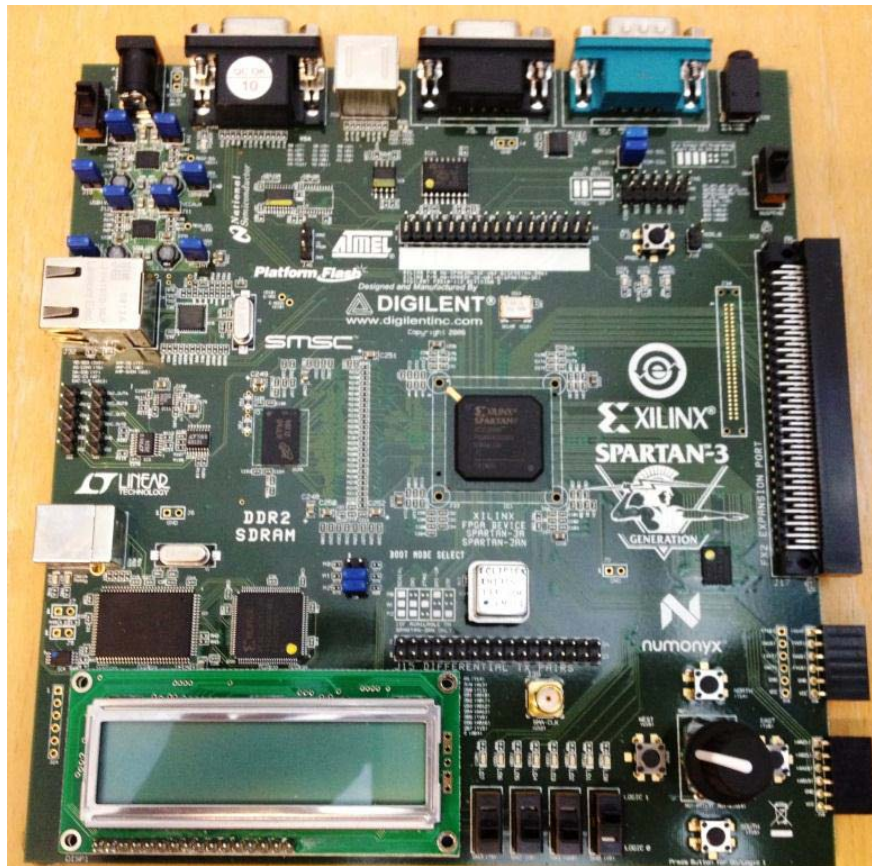
Before validating the design for implementation, an FGPA application developer first simulate the design at multiple stages. All simulators come with IEEE precompiled libraries. For a new design a new library is created and designated as the working library before Register transfer level (RTL) and testbench code can be compiled into it. RTL describes how data is transformed as it is passed from register to register and testbench help the developer to verify that the design is correct. After simulating, the design is synthesized, where the VHDL code is compiled and mapped to a netlist. The netlist is further transformed into gate level description where simulation is repeated to check for warnings and error reports. Before downloading the design to the FPGA, the implementation tool needs to know the exact FPGA part to target. As shown in the figure 4.3, performance objectives are communicated to the implementation tools through timing constraints, this tells the tool how fast the design must operate and which parts of the design should have the first allocation of logic and routing resources. Next the mapper takes the elements from the netlist and packs them into the logic components of the chosen FPGA, then the packed components are placed onto the array of components sites across the FPGA fabric. Finally a bit stream is generated and ready for downloading to the FPGA. In this thesis, Xilinx-Spartan 3AN is used as a test bed for implementation of our authenticated encryption stream cipher.



*Figure 4.3: Xilinx implementation Flow*

#### **4.1.2 Xilinx Spartan 3AN FPGA**

Spartan 3AN FPGA is a highly integrated device targeted for the cost-conscious and high volume market. This platform combines the best attributes of SRAM-based technology with reliable non-volatile flash technology in a single-chip solution. Spartan 3AN platform also provides advanced security features that safeguard against reverse-engineering, cloning and unauthorized overbuilding along with one of the industry's largest user flash [XLXS].



*Figure 4.4: Xilinx Spartan 3AN*

The Spartan 3AN family is available in five non-volatile device options, with system gates ranging from 50K to 1.4M gates, up to 576Kb block RAM, 16 Mb of total embedded flash, and I/O ranging from 108 to 502[XLXS]. The Spartan family builds on the success of the earlier Spartan-II family by increasing the amount of logic resources, the capacity of internal RAM, the total number of I/Os, and the overall level of performance as well as by improving clock management functions. Lots of enhancements are driven from the Virtex-II platform technology [XLX3]. Because of its low cost Spartan 3- FPGAs are perfectly match for wide range of consumer electronics applications, including broadband access, home networking and digital television equipment. The Spartan-3 family is a superior alternative to mask programmed ASICs; it also avoids the initial high cost and the lengthy development cycles. FPGAs in general, permits design upgrades in the field with no hardware replacement as compared to ASICs. Xilinx security division has designed a variety of security solutions for FPGAs as mentioned above. Variety of security solution as designed and implemented for high performance FPGA like Virtex-5 and above, which includes single chip



security solutions in collaboration with National Security Agency (NSA) [XLXN]. Also volume design security solution for Spartan – 3A family which is called Device DNA [XLXD], but the security threats addressed by Xilinx are limited and very specific to the applications [XLXA].

## 4.2 Role of FPGAs in SDR

The term “Software Defined Radio” was first proposed by Joseph Mitola [JMI]. Software Defined Radio (SDR) is a promising and rapidly evolving technology, generating widespread interest in the field of wireless communication. SDR technology offers flexible methods to implement radio functionality, such as signal generation, coding, modulation/demodulation and link-layer protocols in software. In other words, SDR is flexible and reconfigurable. The radio or behavior of a wireless device is able to be changed, upgraded and enhanced simply by changing the software. This gives the possibility of adapting the radios to the user preferences with different operating environments and allows supporting multiple standards without requiring separate hardware for each standard.

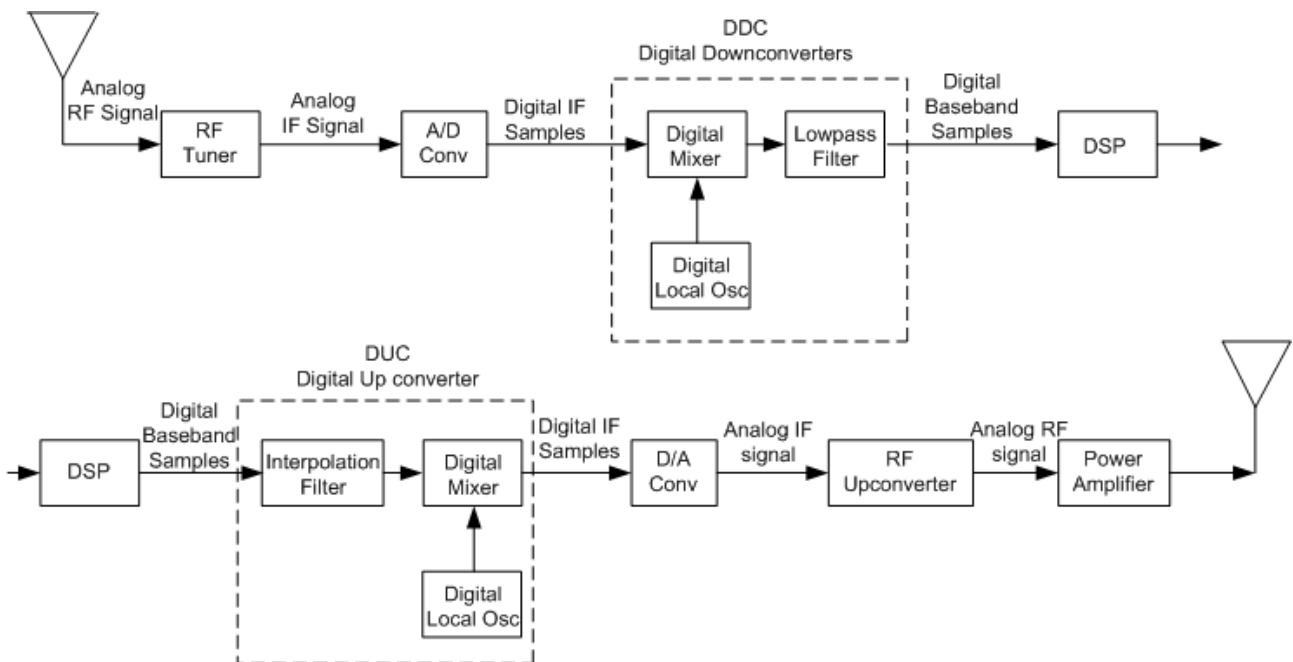


Figure 4.5: Software Defined Radio – a) Receiver, b) Transmitter

Figure 4.5, shows basic elements of Software radio system. The front end consists of analog RF tuner, which converts the high frequency RF signals down to a frequency that an A2D (Analog-to-Digital) converter can handle. The A2D output feeds the digital intermediate frequency (IF)

samples to the Digital Down-convertors (DDC), which consists normally of filtering and down-sampling at the high speed. Processing of IF is suitable application for FPGAs since its computational requirements are relatively simple and its speed requirement is high [ERJ, CUM]. Finally, baseband demodulation and decoding need a computation-intensive algorithm often implemented in a DSP. FPGAs with DSP blocks allow implementing these tasks and join all digital processing in a single chip. For past few years FPGAs have become an increasingly important resource for software defined radio. Implementing software functions like DDC in FPGAs gives significant advantages. These functions related to DDC have seen a shift from being delivered in Application-Specific ICs to FPGAs, the reason being that FPGA realizes lower power consumption and faster signal processing when compared to common micro processors.

Software Defined radio are capable of supporting military, public safety, satellite and general communication applications. Unfortunately, these devices are exposed to security threats – two approaches are considered for the secure communications systems for SDR [HKR]. Firstly, replacing the currently available cryptosystem with a stronger and more efficient cryptosystem and secondly, the development of security using SDR technologies. The main issue in these communication systems is security, whereby an adversary can take control of a radio system to their advantage. As a result, data and traditional application of a device can be changed. Thus, various security functions such as authenticity, integrity, confidentiality, etc, need to be supported in SDR receivers [CNA]. In this thesis, a security solution is proposed by designing and implementing an Authenticated Encryption cryptographic solution at physical layer ex., FPGA instead of higher layers, which will considerably maximize the speed and security of the receiver and also make the system more suitable for low-cost portable devices.

#### **4.2.1 Cross Layer Architecture of Software Defined Radio**

In terms of the traditional OSI model of a communication system, the software defined radio mainly concerns the two lowest layers of the OSI-model. Physical Layer is the most basic layer, gives us the resources for the transmission of bits. Next layer is the Data link layer, provides data transfer across the physical layer. Stream of data is packed into smaller units called frames, the transmission of these frames are controlled by Media Access layer (Sub-layer of data link layer). These two layers provide basic functions for data transfer.

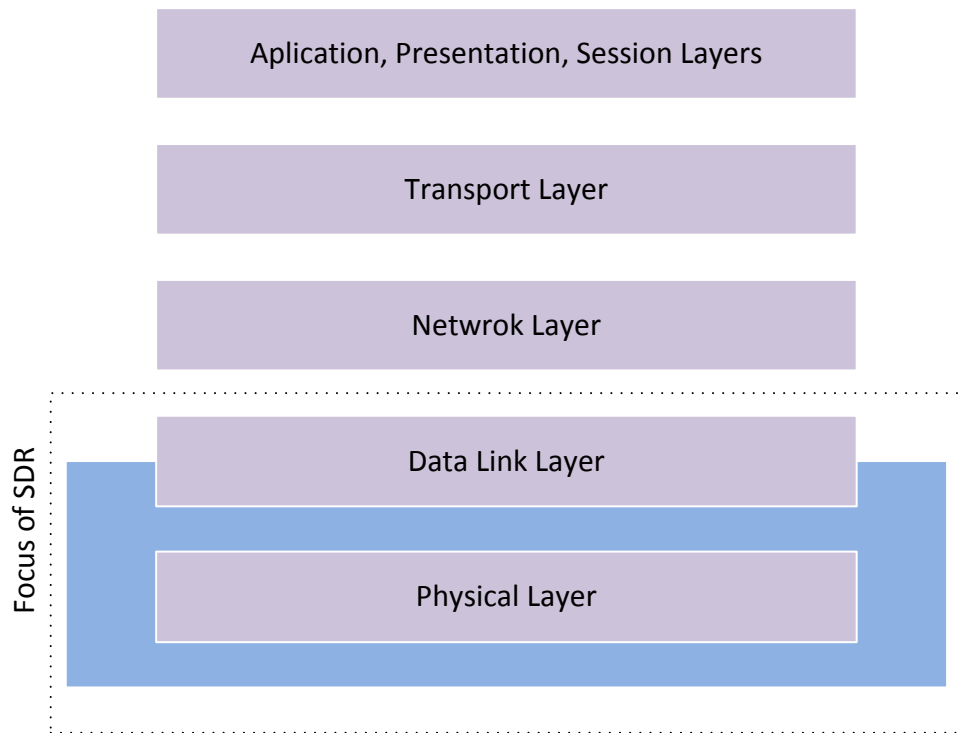


Figure 4.6: SDR Layers in OSI

As shown in figure 4.6, we can say that SDR is a software reconfigurable radio technology, but it is not the applications that are implemented on the radio. Since multiple applications and functions are the main features of future wireless systems, the OSI inspired definition is somewhat debatable.

*Levels of Software Defined Radio* – Sometimes it is not practical to develop radios that have all the features of fully software defined radio. Some radios support only few features while others are fully software controlled. Looking into this Wireless Innovation forum has divided the radios into five levels depending on what is configurable.

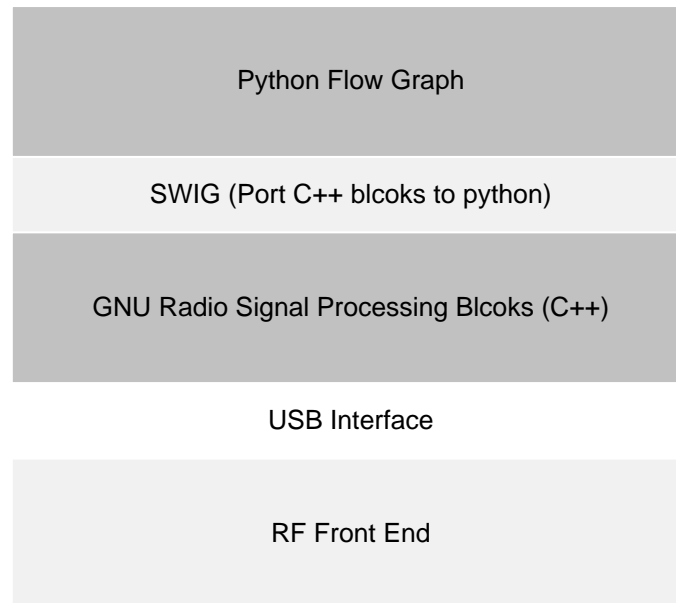
1. All the radios that cannot be changed by software fall into *level 0* category also known as Hardware Radios.
2. In *level 1*, limited functions can be changed by software ex., power levels, frequency etc; these kinds of radios are called Software-Controlled Radios.
3. *Level 2* kinds of radios are the one where reality falls. For most applications, state-of-the-art SDR currently falls into *level 2* kind of radios also known as “*Software Defined Radios*”. Significant portion of the radio is software controlled, parameters like frequency, modulation waveform generation / detection, security etc are controlled by the software. The RF front end still remains hardware based and non-configurable.

4. The radio where the distinction between configurable and non-configurable elements exists very close to the antenna and the “front end” is configurable falls under *level 3*. These radios are also known as Ideal software radios.
5. Finally the Ultimate software radio is a step forward from the *idea software radio*. In addition to the full programmability, it is also able to support a broad range of functions and frequencies at the same time.

In this thesis, security solution is based on level 2 radios, where a SDR consists of hardware frontend and software based base-band platform. The hardware front-end includes an antenna, an amplifier/filter, a down/up converter and AD/DA converter and the base-band part is built up by reconfigurable FPGA. For better understanding, let us look into a well known open source GNU Radio Framework and Universal Software Radio Peripheral (USRP) device.

#### **4.2.2 GNU Radio and USRP**

*GNU Radio* is defined as an open source software toolkit which when combined with minimal hardware allows constructing radios, and thus turns usual hardware into software problems [GNUR]. GNU Radio provides a library of signal processing blocks and the glue to tie these blocks. It also provides blocks for communication with USRP device and new blocks can be added as needed. The main goal is to combine the signal and data processing blocks and this can be achieved by creating a graph where the vertices are signal processing blocks and the edges represent the data flow between them. Programming in GNU Radio is usually the combination of C++ and python. Signal processing blocks are implemented in C++ while the graph and applications are developed in Python as shown in figure 4.7. SWIG is used as an interface compiler which allows easy integration of C++ into scripting language.

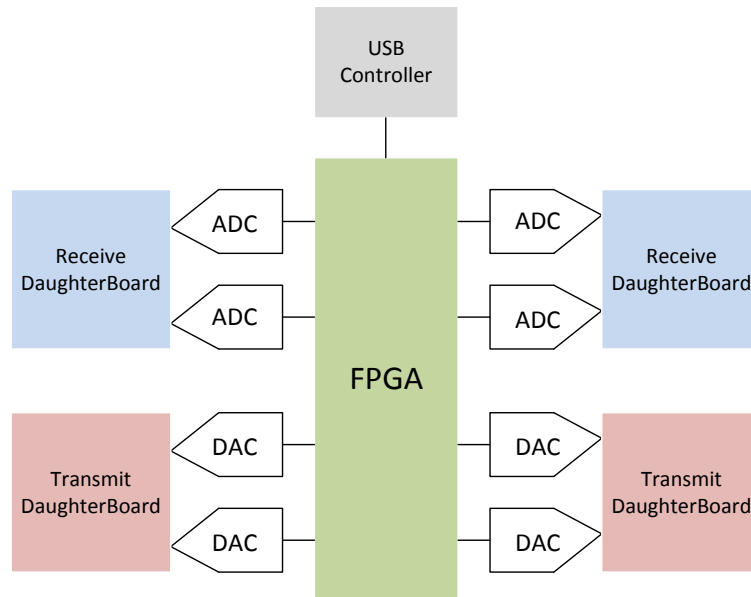


*Figure 4.7: Block diagram of GNU Radio Components*

In most of the cases, a signal processing block processes on a continuous stream of data passing from its input ports to its output ports. A special kind of signal blocks, called sources and sinks, have only output ports or input ports. Sources are the blocks that read from ADCs, a file and sinks write to DACs, a file or graphical display. Each of these blocks has input or output signature that defines the minimum and maximum number of I/O streams it can have.

*Universal Software Radio Peripheral (USRP)*- In addition to GNU Radio software toolkit additional hardware is needed to receive and send radio waves. The USRP is a basic SDR platform, which allows the creation of a Software Defined Radio using any computer with an USB 2.0 port. USRP was developed by Matt Ettus as a low-cost and flexible platform for Software Defined Radio[ETT]. It consists of a main board and up to four daughter boards as shown in figure 4.8. The main/mother board holds upto four 12-bit, 64 M sample/sec ADCs, four 14-bit 128M sample/sec DACs, Field Programmable Gate Array (FPGA) and a programmable USB 2.0 controller. The four daughter boards serve as RF front ends, two for receiving and two for transmitting and can be used simultaneously. In the USRP, high sampling rate processing takes place in the FPGA. It performs high bandwidth math and reduce the data rate to something that can be easily transferred over a USB 2.0 interface. The standard FPGA configuration includes digital down converters (DDC) implemented with the cascaded integrator-comp (CIC) filters. CIC filters are very high-performance filters using only adds and delays. The DDC converts the digitized frequency range, which the receiver daughter boards passed to the ADC, down to the base frequency, also called base band. Then the signals are decimated so that the data rate can be adapted by the USB interface. In case of transmission, the only transmit signal processing blocks in the FPGA are the interpolators. Then

the signal passes through a digital up converter (DUC) which exists as a special chip on the main board, not in the FPGA. DUCs convert the signal to a higher frequency range and finally sent to the DAC.



*Figure 4.8: Block Diagram of the USRP [GNUR]*

Recently several different models have been developed USRP 2, USRP N (Networked series) with increased FPGA resources and USRP E (Embedded series), a standalone SDR device. All the devices on the hardware feature a firmware which enables basic capabilities like Up/down sampling of RF signal, filtering, management of data timestamp and UDP packets on the Giga-bit Ethernet connection to the PC. However considerable amount of resources on the USRP FPGAs are left unused and implementation of additional features may be the valuable improvement to SDRs.

USRP N series includes a Xilinx Spartan 3A-DSP 3400 FPGAs where custom application can be implemented in the FPGA fabric. The FPGA also offers the potential to process up to 100 MHz of RF bandwidth in both transmit and receive directions [ETT]. In N series FPGA firmware are transferred manually over Gigabit Ethernet interface. The USRP E series is for deployable radio application where radio runs as a standalone SDR device. E series employ a unicore TI OMAP processor featuring an ARM Cortex A8 running at 720 Mhz and a TI C64 DSP processor [ETT].

### 4.3 Generic SDR Structure

In the past, computers from different manufactures were completely incompatible. Peripheral equipment from one source did not work with disk and files from a second source or with other system components from a third [WCS]. In addition to high costs and restricted choices, today's communication systems suffer from similar incompatibility issues and these restrictions are driving the integration of SDR technology into commercial, military, emergency response agencies – law enforcement, fire and emergency medical services. Developing a truly international handset is nearly impossible. Multi-mode handset provides some relief but it comes with increased cost, higher power consumption and limited flexibility. Industry has seen parallel evolution of incompatible radio standards all over world for the commercial market. For tactical side, older warfare models simply does not work in today's network-centric warfare model, where forces are tightly integrated and closely coordinated, which requires the ability to quickly and freely communicate across organizational boundaries. A similar situation exists for emergency response agencies across the globe.

SDR technology provides us a foundation for seamless interoperability between communication systems. A typical SDR radio architecture consists of radio hardware (RF front-end) and re-configurable hardware and software.

*RF front-end* – The basis of selecting RF front-end architectures depend on the complexity, cost, power distribution and number of external components. There are three popular RF front-end architectures in use today – *Heterodyne* (or superheterodyne), *Homodyne* (direct conversion) and *Low-IF* (digital - IF) architecture. In heterodyne receiver design, RF signal is converted into IF (intermediate frequency). The incoming signal mixes with the local oscillator to easily step it down to more manageable frequency. The design shifts two signals to the IF signal – the original RF signal and another frequency called the image frequency. However it requires an image rejection filter to remove the image frequency. The major disadvantage of this design is the number of components required. Another design is called homodyne or direct conversion receiver, the RF signal is directly converted to baseband by mixing it with local oscillator signal whose frequency is same as the carrier frequency. However for frequency and phase modulated signals, quadrature downconversion (In-phase 'I' and Quadrature 'Q') is necessary to avoid loss of information. Finally a Low-IF architecture, most commonly used architecture in SDRs. The design is the hybrid of heterodyne and homodyne architecture and comprises the advantages of both the architectures. The RF signal is converted into a low-IF frequency which is then downconverted to baseband signal in domain.

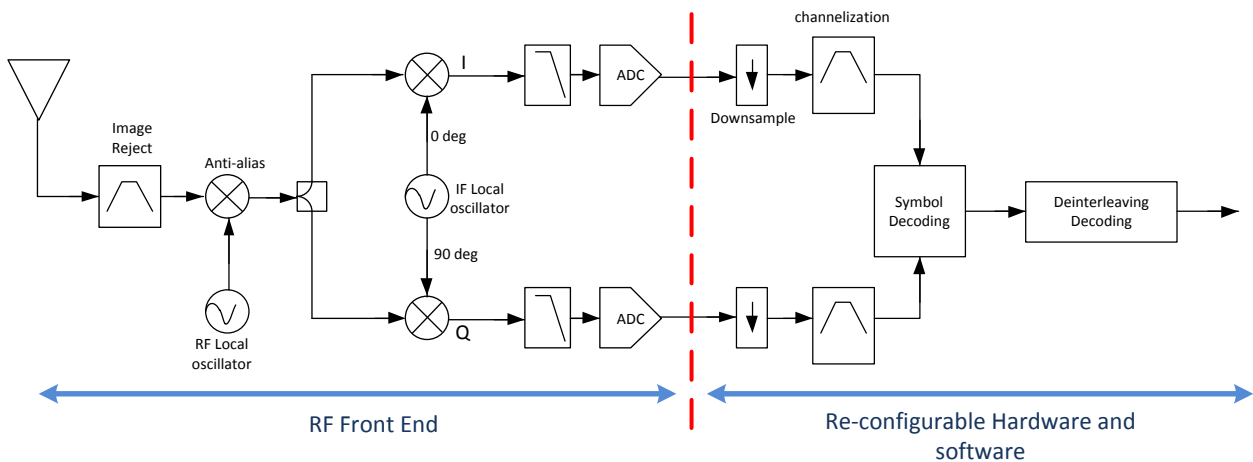


Figure 4.9: Practical SDR receiver

In the first SDR radio architecture proposed by Mitola, all the RF and baseband receive signal processing is digital, enabled by an A/D convertor (ADC) at the antenna. However to fulfill this requirement ADC in the SDR receiver must accomplish extraordinary specifications [AAA]. The present speed of ADCs limits the performance of digital front-end; therefore most radios require an RF to IF conversion. Figure 4.9, shows the practical SDR receiver. The ADCs are placed after IQ demodulator. This configuration requires more components and two ADC channels at the low-frequency IF. With the increase of speed in ADC, the convertors could be placed before IQ demodulator closer to the antenna. This will remove much of extra analog components and architecture begins to move towards an ideal architecture as the transition point to digital occurs much closer to the antenna.

*Re-configurable hardware and software* – In typical SDR the baseband processing – the physical layer is implemented in re-configurable hardware and software. The physical layer consists of RF hardware which includes mixers, filters, modulators/demodulators and amplifiers, as well as field programmable gate arrays (FPGAs).

After converting IF or baseband from analog to digital, the digitized waveforms enter in digital signal processing (DSP) domain. The complex I and Q baseband representation are then digitally filtered to pass the desired channel and last step involves symbol demodulation into bits, de-interleaving and decoding. If Analog to digital converter is placed before the IQ demodulation, then demodulator, local oscillator, filter and mixers are fully digital. The software that emulates these devices runs on DSP or general-purpose processor (GPP) or on a FPGA design.



### 4.3.1 SoC in Software Defined Radio

For defining an effective and efficient SDR design a standard programmable hardware platform is required. However there is limited technical information regarding hardware architectures for realizing SDR system-on-Chips (SoC). It is likely that large scale design integration is necessary to achieve lower form factors and reduced costs in order to target SDRs for wireless segment[PKG]. Recent developments in the reconfigurable platform such as Zynq from Xilinx present a new approach to SDR development[XLX]. Zynq- 7000 EPP from Xilinx is able to integrate / cut parts of standard tactical SDR into one single device, while providing the flexibility to support future waveforms [ASC].

General SDR SoC hardware requirements are mentioned [PKG] but end application requires certain SoC and system level architecture e.x., Software Communications Architecture (SCA) frameworks have additional requirements on SoC, such as partitioning into red and black side[SCAS]. The *Red* side, is the plaintext portion of the radio which contains a general purpose processor and a FPGA. The red part manages the confidential data which is generated by the data source. The *black* side of the radio consists of a black FPGA, a GPP and a modem FPGA for waveform processing. The black part manages encrypted data sent to or received from the unsecured physical communication channel. For data security assurance between encrypted and unencrypted portions of the radio in SoC, the design isolates Red and black parts to prevent information leaks of sensitive data out of the system in plaintext. This isolation is provided by the cryptographic boundary which is named Crypto Sub System (CSS). Data streams between red and black radio areas are encrypted and authenticated by the CSS [MLG].

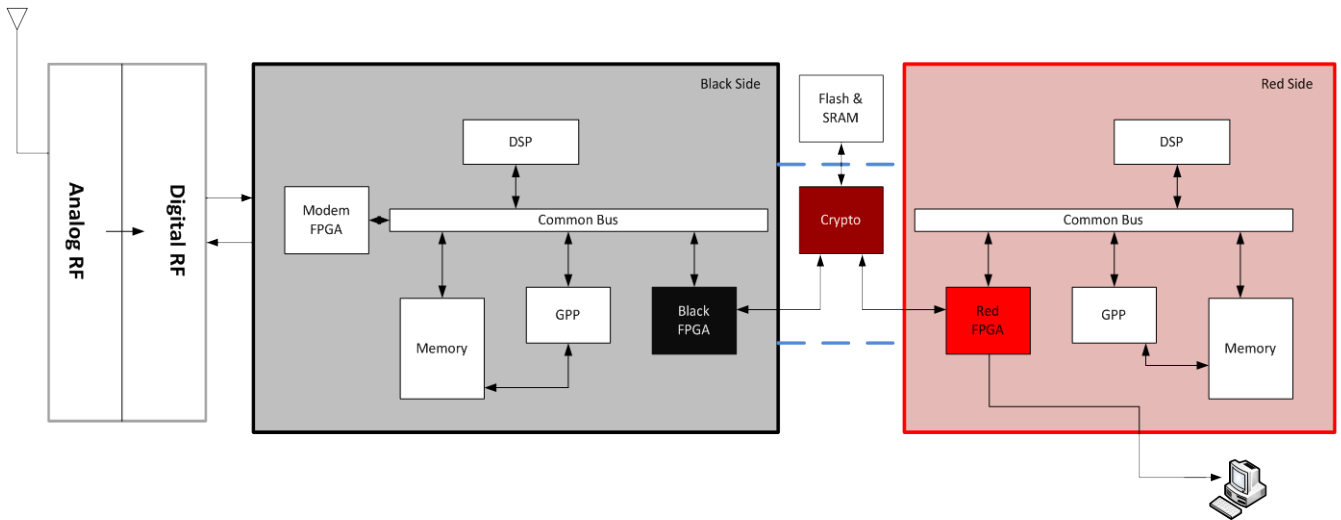
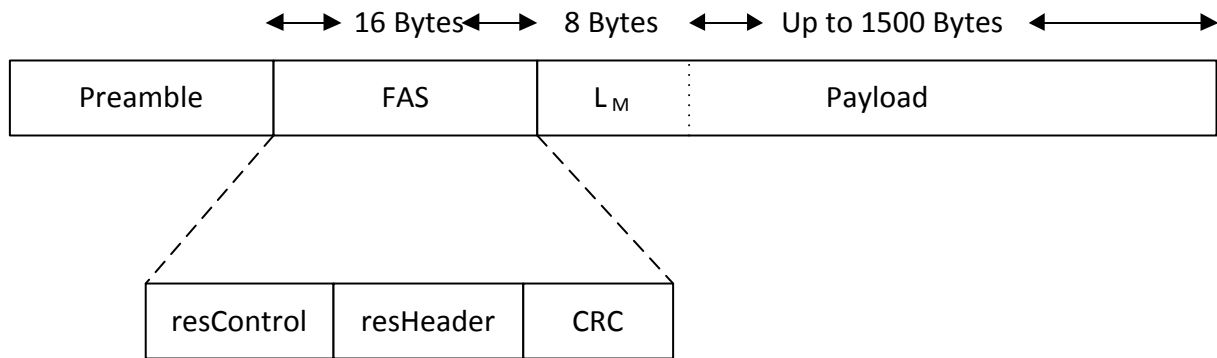


Figure 4.10: SCA Structure, showing the Red and Black FPGAs

As shown in figure 4.10, the required amount of devices (FPGAs and GPPs), the amount of I/O signaling and the high logic density of the devices makes this a non-optimal solution in terms of size, weight, power and cost. However with today's technology it is possible to have single-device platform with high-performance processing system and large programmable logic unit ex., Xilinx Zynq -7000 Extensible processing platform, the modern FPGA, black FPGA, red FPGA, red GPP and black GPP can all be combined into one device[ASC].

### 4.3.2 Secure Communication

Similar to SCA, a secure crypto module i.e., ASC-1 is placed in SoC. ASC-1 is responsible for the confidentiality and integrity of the data flow passing through it from both the sides. Messages encapsulated in data frames, consists of preamble, Frame Alignment Signal (FAS), headers and payload, as shown in figure 4.11 [SKK]. When the message passes through crypto core, only payload have to be encrypted, headers must remain in plaintext.



*Figure 4.11: Frame structure - Preamble is used at the start of each frame, as a sequence of 8-bit words to inform the receiving station that a new packet is arriving. FAS is a distinctive sequence of bits used to accomplish frame alignment, the signal consists of additional bits for status, control and error detection.  $L_M$ , 64-bit representation for the length of the message.*

In order to encrypt the payload, frame preprocessing is necessary where, the message is unpacked and the FAS and payload are extracted. The FAS and headers are then validated according to the policies. Once headers are validated, payload is encrypted and authenticated. Finally, the frame is packed and sent to the final destination.

To initiate the encryption core, we assume that the master key  $K_M$ , is exchanged by key exchange algorithms e.x., Elliptic Curve Cryptography (ECC) and a 56- bit representation of Nonce / Counter is exchanged during the process.

## 4.4 Implementation of ASC-1: An authenticated Encryption Stream Cipher

As mentioned in the previous chapters, the goal of Authenticated Encryption is to achieve faster encryption and message authentication by performing both in the single pass as compared to the traditional encrypt-then-mac approach, which requires 2-passes. ASC-1 gives us the possibility to encrypt and authenticate in 1-pass. Its key size can vary depending on the underlying block cipher, but our block cipher suggestion is Advanced Encryption Standard (AES) with 128-bit key. ASC-1 is divided into two steps – Initial phase generation, Encryption in CFB-like mode and authentication of the data. At the decryption side, same steps are repeated and the computed tag is matched with the received tag for verification.

#### 4.4.1 Initial phase generation

As shown in figure 4.12, Initial phase consists of an initialization vector  $X_0$  and three keys  $K_{1,0}, K_{2,0}, K_{3,0}$ . To calculate these values ASC-1 uses 56-bit of the counter and applies 128-bit AES block cipher to  $0^{70} \parallel 00 \parallel Cntr$ ,  $0^{70} \parallel 01 \parallel Cntr$ ,  $0^{70} \parallel 10 \parallel Cntr$ ,  $l(M) \parallel 00000011 \parallel Cntr$ , using Master key  $K_M$ . Where  $l(M)$ , is the 64-bit representation of the bit length of the Message M. In this scheme, we assume that the maximum number of messages and maximum length to be encrypted is  $2^{48}$ .

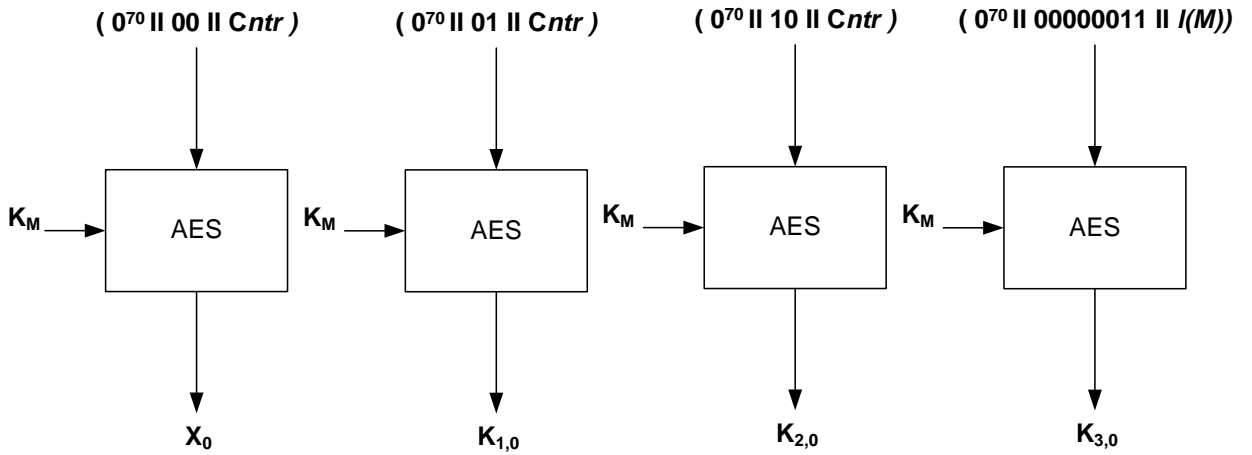


Figure 4.12: Initialization vector and Key Generation

To obtain the initial values, standard AES-128 bit implementation is used in ECB (Electronic Code book) mode. 128-bit input block is passed through four basic operations, SubBytes, ShiftRows, MixColumns and AddRoundKey. (Details are mentioned in Chapter 2). Key  $K_M$  is used to calculate all the initial values.

#### 4.4.2 Encryption Process

Before initializing encryption process, Key  $K_{1,0}$  and  $K_{2,0}$  are concatenated together and AES-256 key scheduling algorithm is applied to derive 14 round keys. Keys  $K_2, K_3, K_4$  and  $K_5$  are used as round keys in the first round and Keys  $K_7, K_8, K_9$  and  $K_{10}$  are used in the second round. Keys  $K_{11}$  and  $K_1$  are used as a whitening keys in the first and second rounds of 4R-AES transformation respectively. In AES key scheduling round keys can either be generated on-the-fly or it can be

stored in the internal memory. Whereas in ASC-1, because of using  $K_1$  and  $K_{11}$  for key whitening it is only possible to store the keys in the memory during the key setup phase, and then read them from this memory whenever they are required by the encryption/decryption unit. Dedicated embedded memory block are ideal for storing keys. A special feature of Xilinx Spartan-3 FPGA[XLX3] is used in ASC-1, which offers multiple block RAMs, organized in columns. Each block RAM contains 18 kb of fast static RAM[XLXS]. The Xilinx Spartan-3 xc3s700an has 20 block RAMs and it can be used as single or dual port RAM.

*ASC-1 Encryption Block* consists of 4 Round AES. To initialize the encryption module, 128 bit Initialization vector is provided as an input to the ASC-1 encryption algorithm. ASC-1 performs number of transformations to the input data to give a 128-bit leak  $l_1, l_2, \dots, l_{16}$  and output state  $y_1, y_2, \dots, y_{16}$ . ASC-1 stream cipher performs 4 discrete transformations – *AddRoundKey*, *SubBytes*, *ShiftRows* and *MixColumns* – in that specific order, as shown in figure 4.13. Four bytes are leaked at the end of every round and positions of the leaks depend on the number of the round (even or odd). Finally whitening key byte is added before each extracted byte.

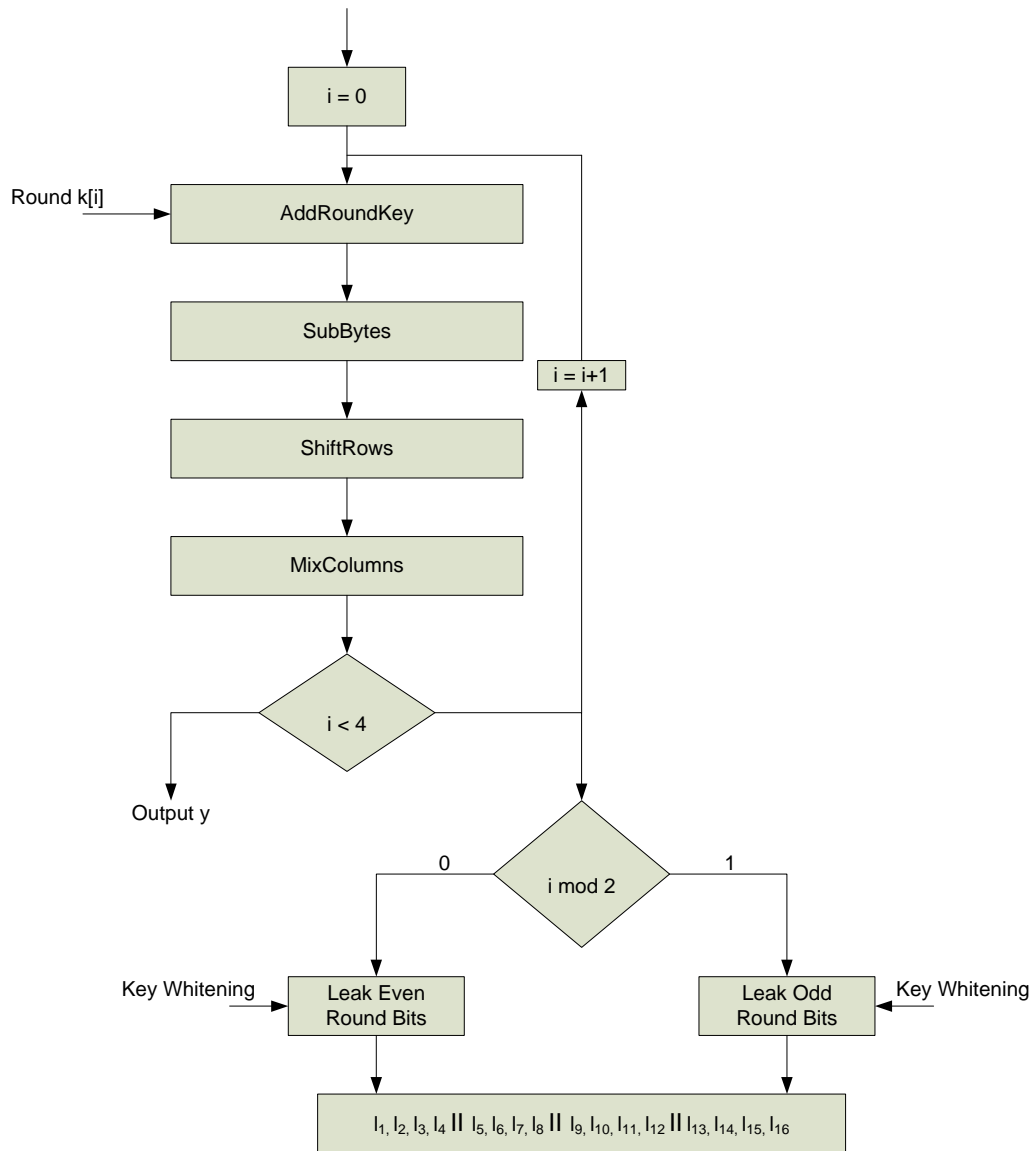


Figure 4.13: ASC-1 Flowchart

#### 4.4.3 Proposed ASC-1 Architecture

The high-level architectural organization of the ASC-1 encryption core is presented in figure 4.14. The system is divided into five logical blocks. The Initial input interface is responsible for feeding data to the key logic and the processing core. Key logic handles all the key scheduling operations and processing core block performs all the main encryption process. SBox block is a ROM that is used for the SubBytes transformation by key logic and core block. Finally the Control unit is used for the synchronization and communication with the external logic. Let us further look into the functionality of each logic block in detail.

*Initial input interface* – For initial phase generation i.e., initialization vector  $X_0$  and three keys  $K_{1,0}, K_{2,0}, K_{3,0}$ , a new counter/nonce is loaded. The Initial input interface concatenates the values of the counter with the pre-defined values stored in the local registers. The processing core unit is then notified that an initial state is available for processing.

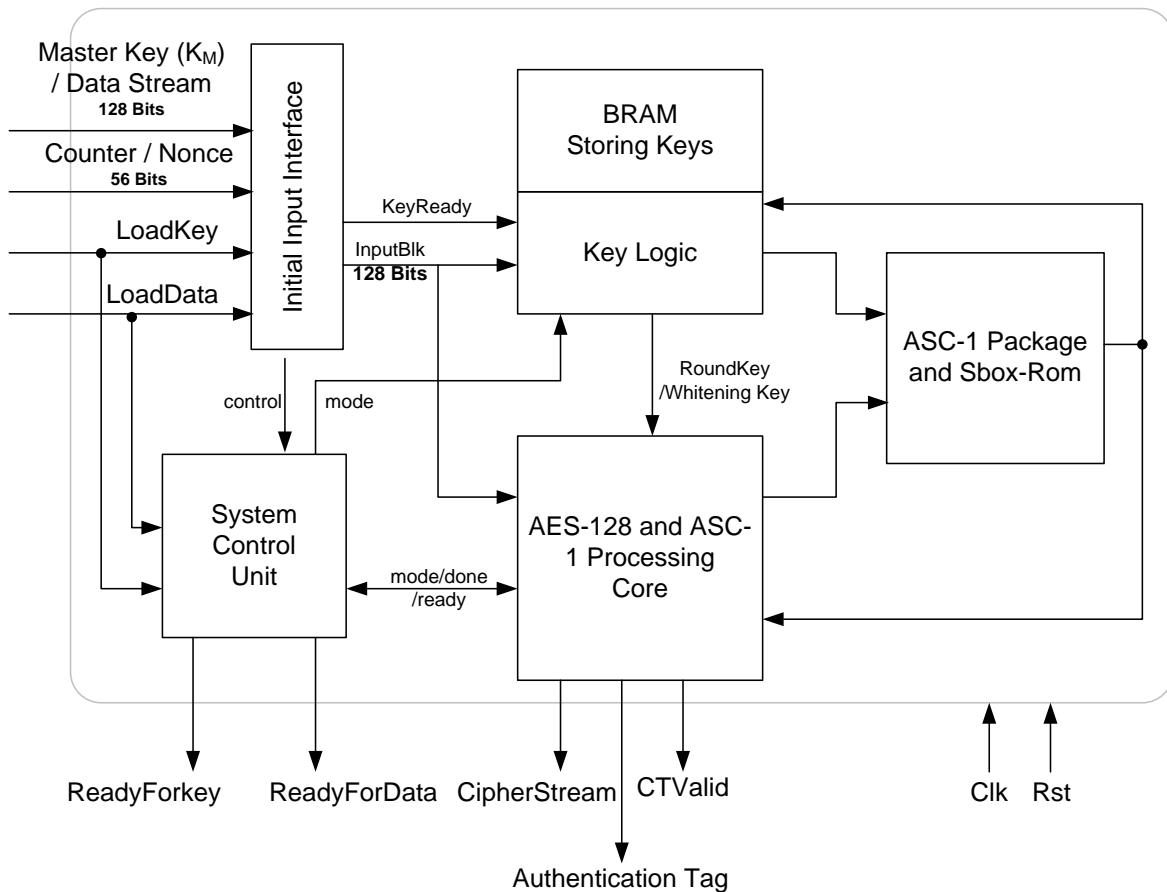


Figure 4.14: Block diagram of ASC-1

*Key Logic* – As shown before, the scheme requires a new round key for every encryption round. When a new key (128-bits for AES and 256-bits for ASC-1 encryption algorithm) is loaded, the block starts generating round keys based on a single external key. Three different approaches can be used to calculate the keys. In the online approach, a new round key is calculated at every encryption round using the previous round key. Another approach is known as “offline” or “stored-key”, all the round keys are calculated upon the reception of the initial cipher key before the start of encryption core and stores them in a local memory. The memory is accessed at every

encryption round in order to provide the necessary round key. The third method for the generation is to use an external source for example a key generator or an external processor. The external source calculates the keys and seeds them sequentially to the AES processing core.

This implementation is based on the second approach for both the schemes i.e., AES-128 and ASC-1. In this approach all the round keys are stored in the memory for a number of reasons. For initial phase generation same key ( $K_M$ ) is used to encrypt initialization vector (IV), two initial keys ( $K_{1,0}, K_{2,0}$ ) for key scheduling for ASC-1 encryption and key ( $K_{3,0}$ ) for authentication of data. The round keys derived from the Master key is stored in the memory and during the encryption process right round key is accessed from the memory to perform encryption operation. In case of encryption of stream data using ASC-1, 256-bit key ( $K_{1,0} || K_{2,0}$ ) is loaded to the key logic for key expansion. Fourteen round keys ( $K_1, K_2, \dots, K_{14}$ ) are derived and stored in the memory. The key logic block performs two main functions. The key expansion process and read/write round keys to the memory block. The first one is performed whenever a new cipher key is inserted to the block and second one is to fetch round keys from the local memory for encryption process. All the operations in the key logic are controlled by the system control unit.

Figure 4.15 a, b demonstrates the generation of new round keys from the previous round. In order to generate new round key, two operations are performed, RWord (Rot Word) and SWord (Sub Word usually called as SubBytes). The first one cyclically shifts the bytes of last 32-bit word of the previous round key by one position to the left and SWord operation simply takes the output of the RWord and applies SubBytes transformation independently on each byte of a state using a substitution table (S-BOX). As shown below, key expansion routine for 256-bit key is slightly different than 128-bit. SubWord () operation is applied to  $Rkey\ word_4$  prior to XOR.



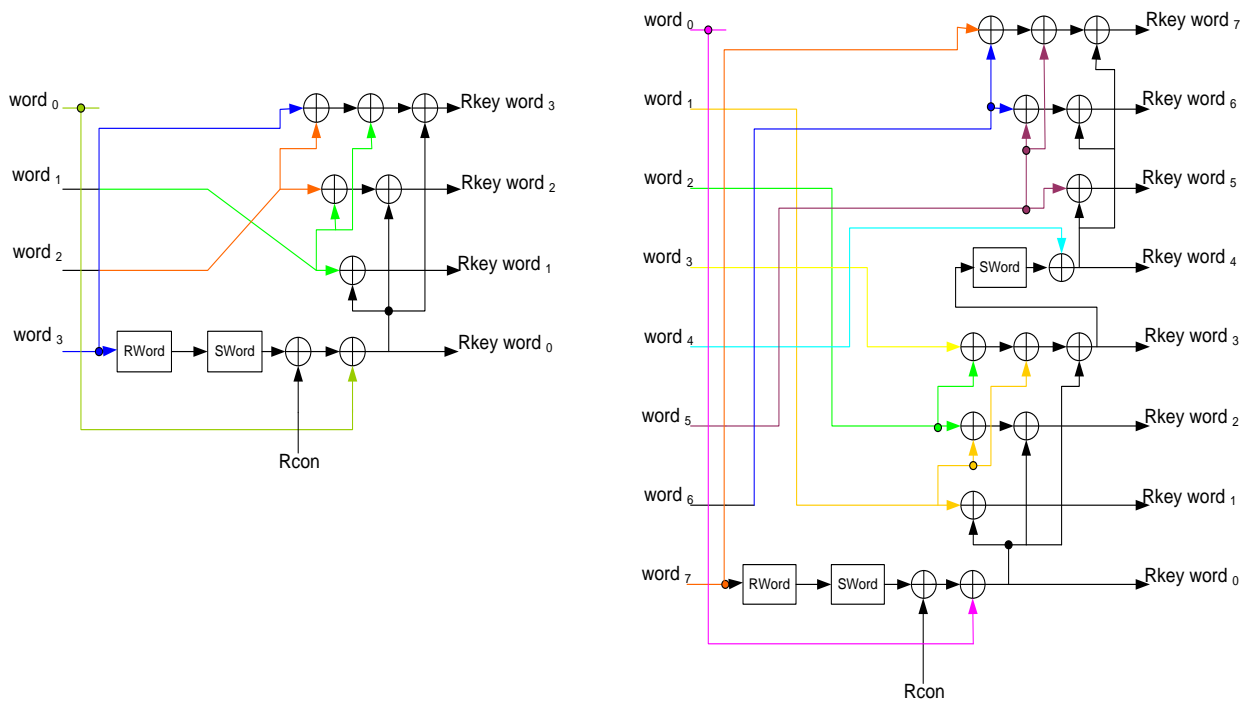


Figure 4.15 a, b: Fully parallel pipelined structure - a) key expansion round for 128-bit AES, b) key expansion round using 256-bit AES for ASC-1.

**AES and ASC-1 Processing Core** - Figure 4.16 shows the process core block performing AES-128 and ASC-1 encryption process. AES encryption core is used only for the generation of Initialization vectors and the keys used in ASC-1. This could be before initializing ASC-1 encryption core or whenever new IV or keys are needed. In order to complete the encryption process in AES of the data block, 10 encryption rounds are needed. Each round performs SubBytes, ShiftRows, MixColumns and AddRoundkey transformations. The SubBytes transformation is applied on each byte of input block, altering its value by a non-linear manner. The S-Box byte substitution can be implemented either by using combinational logic or a ROM containing all 256 possible pre-calculated outcomes. In this implementation S-Box is implemented as a ROM, because the cost of implementing combinational logic for the S-Box in resource usage is significantly larger with no or very little extra performance is gained. Xilinx Spartan -3AN / Virtex V FPGAs provide fast on-chip memories, called BlockRAMs, which are ideal for implementation of ROM. After byte substitution in SubBytes, ShiftRow transformation is applied to each row of the state and changes the position of each byte in that row. Next MixColumns transformation is applied to the state in a column by column fashion. This transformation is quite simple and can be implemented by a network of XOR gates. Finally AddRoundkey transformation is applied, where 128-bit state is XORed with a 128-bit round key.

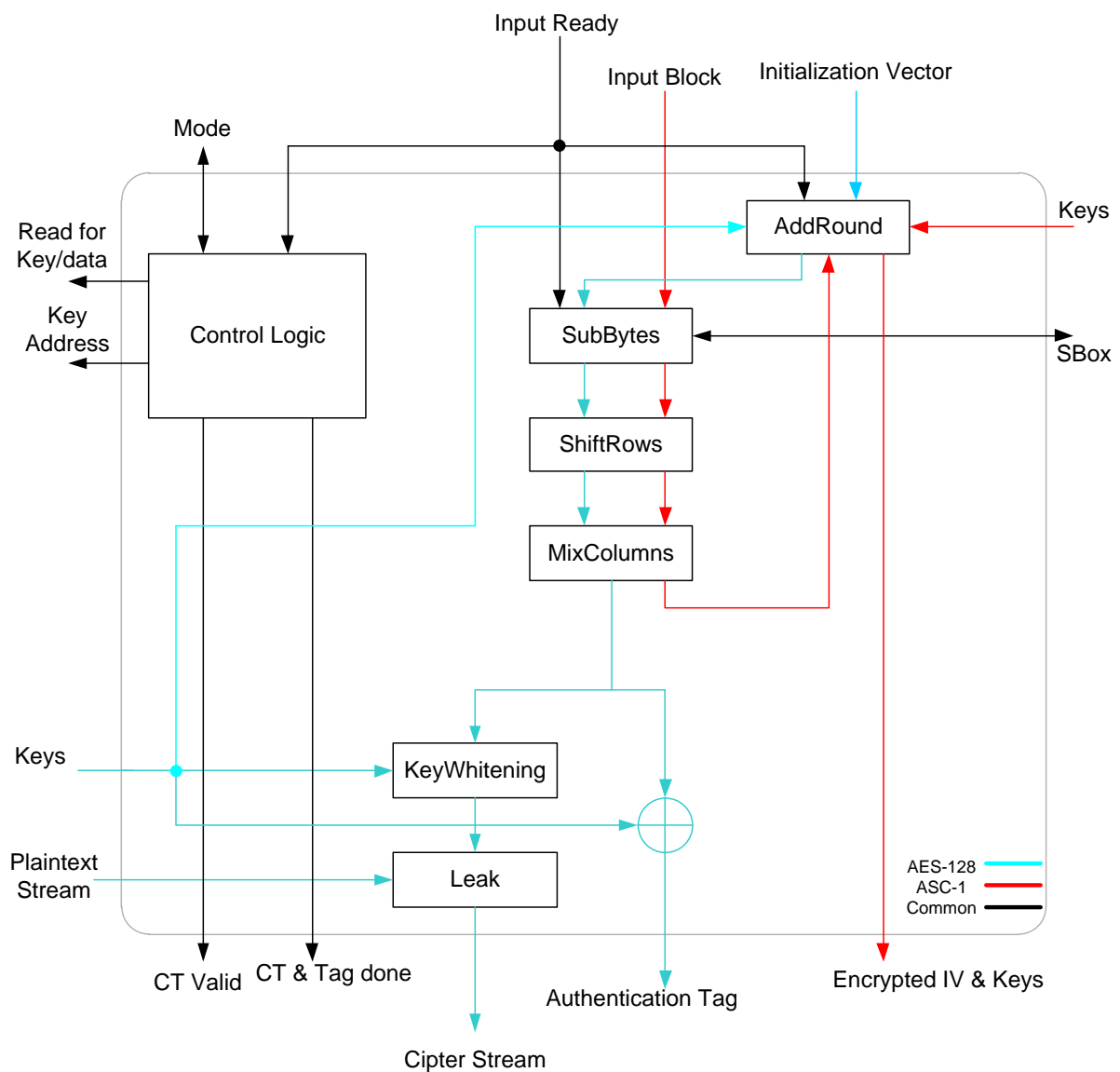


Figure 4.16: AES-128 & ASC-1 encryption core block

Once the Initialization Vector and keys are encrypted using AES-128, keys are fed into key logic

block for the calculation of round keys and IV is used to initialize ASC-1 scheme. From figure 4.16, the underlying block used in ASC-1 is AES, so same transformations are applied to the block but in different order. In ASC-1 unlike AES, Addroundkey transformation is the first block and after MixColumns, KeyWhitening is applied to the specific bytes before extracting from intermediate rounds. Four Round AES ASC-1, operates in a Cipher Feedback (CFB) mode which means that the processing of each plaintext block has to be completed before the processing of the next one starts. Therefore, implementation presented here is sequential. However from figure 4.15, parts of implementation are implemented in parallel pipeline mode.

Systems control unit is implemented as a finite state machine to supervise the core between AES and ASC-1, generate address for accessing the round keys from the block and handle communication between blocks. The unit generates the signal to notify the external source that a new plaintext may be loaded as soon as core is ready. Finally Authentication tag is calculated once  $n$  numbers of blocks are encrypted (maximum number of messages and maximum length to be encrypted is  $2^{48}$ ).

## 4.5 Conclusions

Software Defined Radio is a promising and rapidly evolving technology, generating widespread interest in the field of wireless communication. The main feature of SDR is its ability to dynamically adapt according to the radio environment through the re-configurability of its components. This feature gives SDR systems the ability to support a variety of mobile radio standards. Instead of implementing radio functional blocks on inflexible ASICs in the past, the technologies like FPGAs, DSPs and GPPs are used to build software radio blocks. These components have reconfigurable capability and deliver flexibility of programmable architecture with power efficient and performance.

We have discussed the architecture and design implementation flow of the FPGA in section 4.1. The basic design flow of FPGA consists of writing a code in hardware description language and a text bench, followed by simulating the model together with its text environment. The design is synthesized to a particular FPGA (in our case – Xilinx Spartan 3AN), then placed and routed on the chip. Finally the hardware was created and the bitstream (Image) is downloaded to the chip to program it. We discussed the role of FPGAs in SDR in section 4.2 and for better understanding we have also discussed a well known open source GNU radio framework and USRP device.

In Section 4.3 we presented a generic SDR structure – a typical SDR radio architecture with RF front-end and re-configurable hardware and software. We have also discussed SDR system-on-

chips (SoC) with an example of recent development presented by Xilinx to integrate standard tactical SDR into one single device. Based on our proposed AE scheme, section 4.4 presents the design and implementation of ASC-1 on FPGA. The target device used for this implementation is Xilinx Spartan- 3 sxc3s700an FPGA. The proposed architecture is divided into five functional blocks including a key logic and AES & ASC-1 encryption core. Whole architecture is controlled by a systems control unit implemented as a finite state machine to supervise the core between AES and ASC-1, generate addresses for accessing the round keys from the block and handle communication between blocks.

## References

- [SBJ]S. Brown and J. Rose. FPGA and CPLD Architectures: A Tutorial, IEEE Design & Test of Computers, pp.42-57, 1996.
- [BZE]B. Zeidman. Designing with FPGAs and CPLDs. CMP Books, Lawrence, KS, 2002.
- [XLX] All Programmable Technologies and Devices. [http:// www.Xilinx.com](http://www.Xilinx.com)
- [XLXS] Spartan-3 Generation Configuration User Guide, UG332 (v1.6) October 26,2009.
- [XLX3] XILINX Spartan-3 FPGA Family, 2009.  
[http://www.xilinx.com/support/documentation/data\\_sheets/ds099.pdf](http://www.xilinx.com/support/documentation/data_sheets/ds099.pdf)
- [XLXN] M.McLean & J. Moore. Xilinx FPGA-Based Single Chip Cryptographic Solution,2007.  
<http://www.mil-embedded.com/pdfs/NSA.Mar07.pdf>
- [XLXD] Security Solution Using Spartan-3 Generation FPGAs , Wp266, v 1.1, 2008  
[http://www.xilinx.com/support/documentation/white\\_papers/wp266.pdf](http://www.xilinx.com/support/documentation/white_papers/wp266.pdf)
- [XLXA] Advanced Security Schemes for Spartan- 3A/3AN/3A DSP FPGAs, WP267 ,v1.0, 2007  
[www.xilinx.com/support/documentation/white\\_papers/wp267.pdf](http://www.xilinx.com/support/documentation/white_papers/wp267.pdf)
- [JMI ]J. Mitola. Cognitive radio: Making software radios more personal, *IEEE Pers. Commun.*, vol. 6, no. 4, pp. 13–18, Aug. 1999.
- [ERJ]E. Ramon and J. Carrabina. Using FPGAs for Software Defined Radio and Convolution Encoding. The last VliS output to the GUI, Systems: a PHY layer for an 802.15.4 transceiver, V Jornadas de and is used to display the results. Computacion reconfigurable y Aplicaciones (JCRA), Granada, 14-16 September, 2005.
- [CUM]M. Cummings and S. Haruyama. FPGA in the Software Radio, IEEE Comm. Magazine, pp.108-112, Feb. 1999.

[HKR]H. Uchikawa, K. Umebayashi, and R. Kohno. Secure download system based on software defined radio composed of FPGAs. In *Proceedings of the 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 2002.

[CNA] C Li, N.K. JHA and A. Raghunathan. Secure reconfiguration of Software-Defined Radio; ACM Transactions on Embedded Computing Systems, Vol. 11, No.1, March 2012

[GNUR]GNU Software Radio project. <http://www.gnu.org/software/gnuradio/>.

[ETT] Ettus Research LLC, The USRP, <http://www.ettus.com/>

[WCS] Wireless Communication Standards, Awareness Guide, U.S. Department OF Homeland Security. <http://www.dhs.gov>

[AAA]A. A. Abidi. The Path to the Software-Defined Radio Receiver. *IEEE J. Solid-State Circuits*, vol. 42, no. 5, pp. 954-966, May 2007.

[PKG] P.K Gopalakrishnan. Hardware Platforms for Software Defined Radio, KPIT Cummins Infosystems Ltd, V1.2 Sept-2011

[ASC]A Schreber. Hands on RF: Handle multiple waveforms in a software-defined radio platform. Xilinx. Jan-2012

[SCAS] Software Communications Architecture Specification, JTRS std 2.2.2, Rev. FINAL, Mai 2006. <http://sca.jpeojtrs.mil/>

[MLG]M. Grand, L. Bossuet, G. Gogniat, B. Le Gal and D. Dallet. *A Reconfigurable Crypto Sub System for the Software Communication Architecture*. In *Proceedings of MILCOM 2009*.

[SKK] S.K Khaleelahmed, M. Kantikiran, Dr. K. Ramakrishna and A. Anitha. Novel Implementation of SDR Multi Channel TDM Non Coherent DBPSK On Reconfigurable FPGA. *International Journal of Advanced Engineering Sciences and Technologies*. Vol 5, Issue 2, pp. 282-289. 2011.

# 5

## Hardware Implementation Results

The results for hardware implementation of “ASC-1: An Authenticated Encryption Stream Cipher” are tabulated in this chapter. Area utilization, throughput and latency are provided for FPGA implementation. ASC-1 is implemented in VHDL and the hardware used is Xilinx Spartan -3AN. The software used for this design is Xilinx ISE -12.4. This is used for writing, debugging and optimizing, and all the simulations are done in ISim simulator. The system is tested and verified against test vectors (See Appendix I).

This chapter also covers different hardware architecture and explores area / delay tradeoffs in the implementation. The results also show the optimal payload lengths for maximum throughput corresponding to the Bit Error Rate (BER) on different data rates.

### 5.1 Parameters of Hardware Implementation

Hardware implementation of Encryption schemes depends on several performance parameters like implementation area (number of slices used in FPGA), throughput and latency. However, for authenticated encryption schemes, the throughput is not a static value, but is dependent on the length of the message.

ASC-1 uses Cipher feedback mode of operation, which means same forward function, is used for encryption as well as decryption of plaintext. The throughput of encryption/decryption is defined as the number of bits encrypted (decrypted) in a unit time. The unit of throughput is Mbit/s or Gbits/s. Latency is defined as a time necessary to encrypt/decrypt a single block of data (Plaintext /ciphertext). The unit of latency is ns (nanosecond).

The general formula for throughput and Latency computation is as follows [GAJ] :

$$\text{Throughput} = \frac{\text{size of the block (in bits)} * \text{number of blocks processed simultaneously}}{\text{Latency}}$$

$$\text{Latency} = \text{Number of clock cycles to encrypt a single block of data} * \text{clock cycle period}$$

## 5.2 Block cipher modes of operation

As presented in chapter 4, ASC-1 is divided into two steps – Initial phase key generation, Encryption in CFB-like mode and authentication of data. The first part is calculated by using non-feedback mode, such as Electronic Code Book mode (ECB) and Encryption is done in feedback mode, such as Cipher feedback mode. In non-feedback modes of operation, encryption of each block is performed independently from other blocks and all the blocks can be encrypted in parallel. Whereas in feedback mode of operation, encryption of next block of data cannot be started until encryption of previous block is completed, which means all block must be processed sequentially. Normally this limitation in feedback modes does not concern decryption process and several blocks of ciphertext can be processed in parallel. However ASC-1 can only be processed sequentially at encryption and decryption because of the calculation of authentication tag, which is based on all the previous blocks.

### 5.2.1 Hardware Architecture for Feedback cipher modes

*Basic Iterative Architecture:* Iterative architecture is the most basic hardware architecture, where only single round of the block cipher is implemented as a combinational logic. As shown in the figure 5.1, input data block is fed to the circuit and stored in the register. For every clock cycle, intermediate round data after the completion of one round is fed back to the circuit.

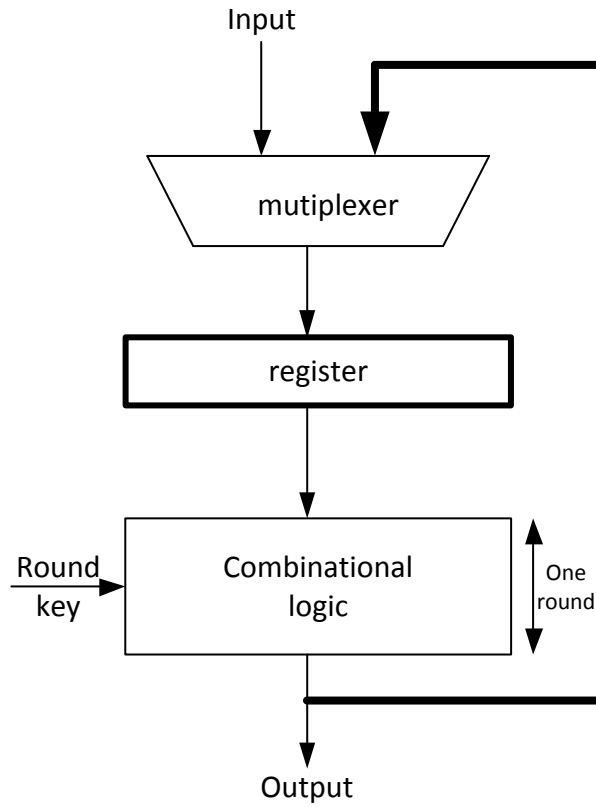


Figure 5.1: Iterative Architecture of Block cipher

In Iterative architecture only one block of data is processed at a time. So the number of clock cycles needed to encrypt one block is equal to number of rounds in a cryptosystem. The throughput and latency in case of basic iterative architecture is [GAJ]

$$Throughput_{ite} = \frac{\text{size of the block (in bits)}}{Latency_{ite}}$$

$$Latency_{ite} = \text{Number of rounds} * \text{clock cycle period}$$

*Partial and full loop unrolling:* Figure 5.2-a&b, shows architecture with partial loop and full loop unrolling. The main difference between partial loop and basic iterative architecture is that the combinational logic in partial loop implements N rounds instead of one round. Even though the number of clock cycles to encrypt a single blocks of data decreases by a factor of N, the minimum clock period increase by a factor slightly smaller than N. This leads to a small increase in throughput and decrease in latency at the cost of large area penalty [JOZ]. The multiplexer and feedback loop is omitted in full loop unrolling. This leads to small increase in throughput and



decrease in area as compared to partial loop unrolling. Both the loop unrolling architectures provide increase in circuit speed for both feedback and non-feedback modes. But this increase is minimal at the cost of circuit space used by the architecture.

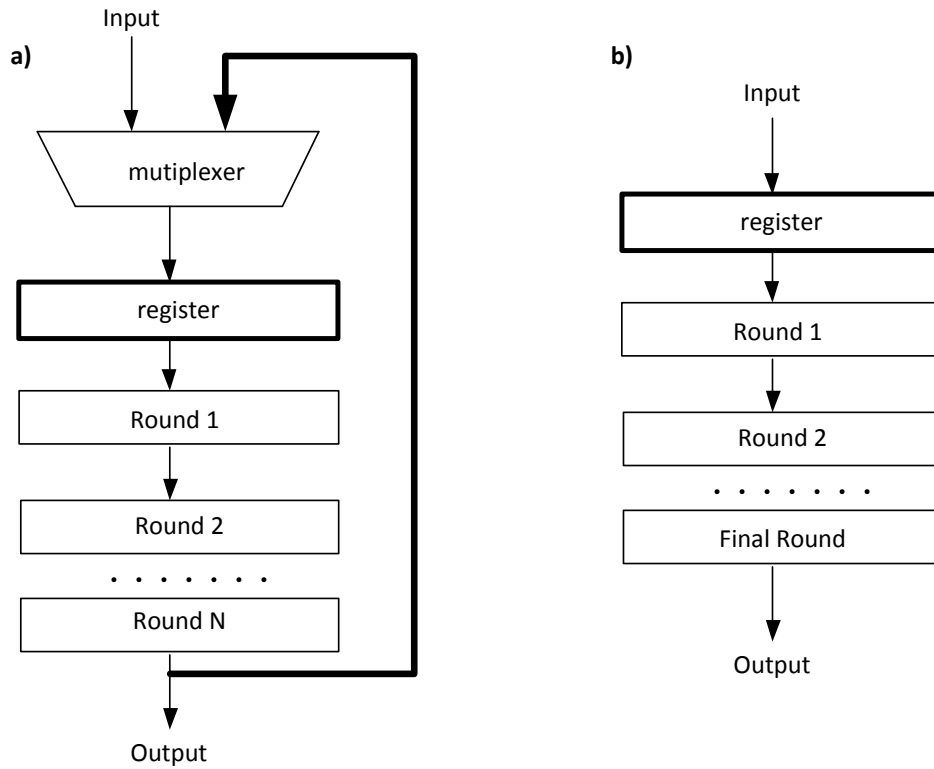


Figure 5.2: Hardware Architecture - a) Partial loop unrolling, b) with full loop unrolling

### 5.2.2 Hardware Architecture in non-feedback cipher mode

**Pipelining:** The Pipelined architecture can increase the speed of encryption/decryption by processing blocks of data simultaneously. In the cases where the available circuit area is not large enough to fit all the rounds of block cipher, partial outer-round pipelining architecture could be a solution, as shown in figure 5.3-a. It is realized by inserting rows of registers among combinational logic on the boundaries between two subsequent cipher rounds. With this, N blocks of data can be processed at the same time, with each of these block stored in a different register at the end of clock cycle. The area of the circuit and throughput in case of partial outer-round pipelining is proportional to N, but the latency remains same as in the basic iterative architecture[GAJ]. Similar to full loop unrolling architecture, full outer-round pipelining architecture can be applied where

feedback loop is not required, shown in figure 5.3-b. This can only be realized if circuit area is not a constraint.

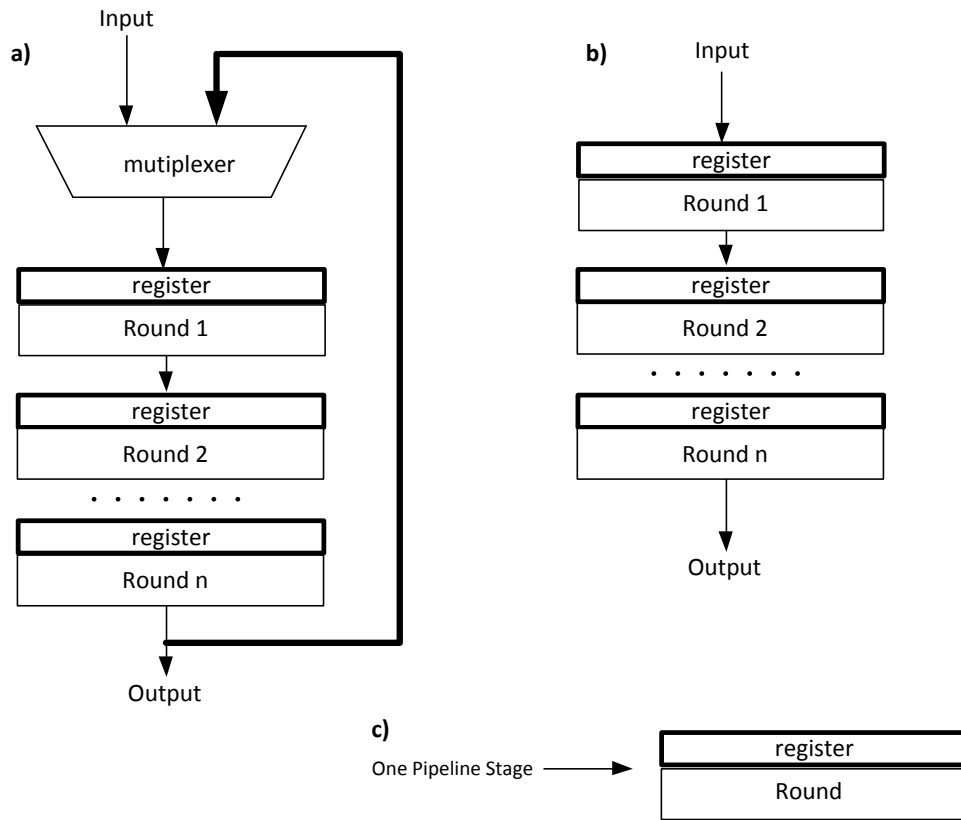


Figure 5.3: Hardware architecture for non-feedback cipher modes – a) partial outer round pipelining, b) full outer round pipelining, c) one pipelining stage.

For optimal pipeline architectures, the registers can also be inserted inside of a cipher round known as inner round pipelining, as in figure 5.4-a. By combining the outer and inner loop pipeline architecture and inserting right amount of registers inside each cipher round will increase the throughput with minimal increase in the circuit area. Adding more registers after optimal amount of registers, may still increase the throughput, but the throughput to area ratio will get worse. The throughput for the partial and full inner- and outer- round pipelining is [GAJ]:

$$Throughput_{par\_pipe}(N, n) = \frac{\text{size of the block (in bits)} * \text{number of blocks processed simultaneously}(N)}{\text{Number of clock cycles to encrypt a single block of data} * Clk_{inner\_round}(n)}$$

Where  $n$  is the number of inner-round registers,  $N$  is the number of outer-round registers, and  $Clk_{inner\_round}(n)$  is the clock period for inner-round pipelining.

$$Throughput_{full\_pipe}(N, n_{opt}) = \frac{\text{size of the block (in bits)}}{Clk_{inner\_round}(n_{opt})}$$

Where  $n_{opt}$  is the maximum number of inner-round registers optimal for the throughput to area ratio.

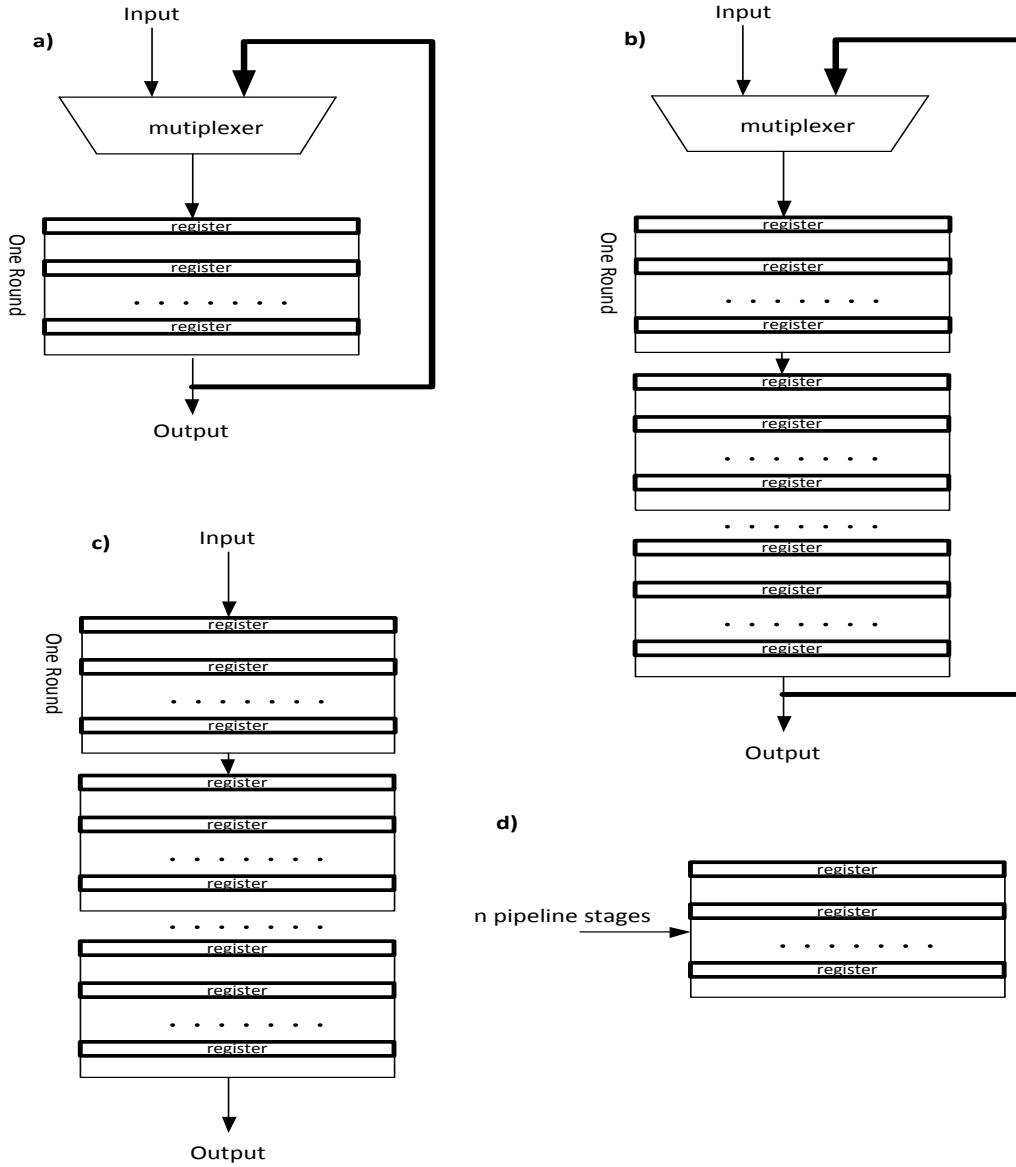


Figure 5.4: Optimal Hardware Architecture for non-feedback cipher modes – a) Inner-round pipelining, b) Partial inner and outer – round pipelining, c) full inner and outer – round pipelining, d) one round n-stage pipeline.

### 5.3 Performance of ASC-1 Crypto core

In Authenticated Encryption stream cipher (ASC-1) scheme, the underlying block cipher (AES) is used only in the forward “encrpytion” direction for both ASC-1 encryption and decryption. This characteristic make ASC-1 an attractive candidate for hardware where area is limited.

Before looking into the overall performance of the core, let us look at the implementation and results of basic operations of AES in FPGAs.

*SubBytes* – The SubByte transformation is applied to each byte of the state matrix, where byte is replaced with another byte from S-Box. The S-Box byte substitution function can be implemented either by using combinational logic [ABR] or using a 256 X 8 bit look up table, using ROM (Read Only Memory). Use of ROM is the most optimal implementation in terms of area/performance – in an FPGA. To access ROM, inputs used as addresses and output is acquired at the data out bus. A state matrix consists of 16 bytes and for each byte substitution 16 ROMs have to be used. FPGA used in this implementation Xilinx Spartan-3AN provides fast on-chip memories, called BlockRAMs. BlockRAMs can be configured as dual port ROMs, as shown in figure 5.5. This reduces the amount of ROMs in half i.e., 8. This whole process requires only one clock cycle.

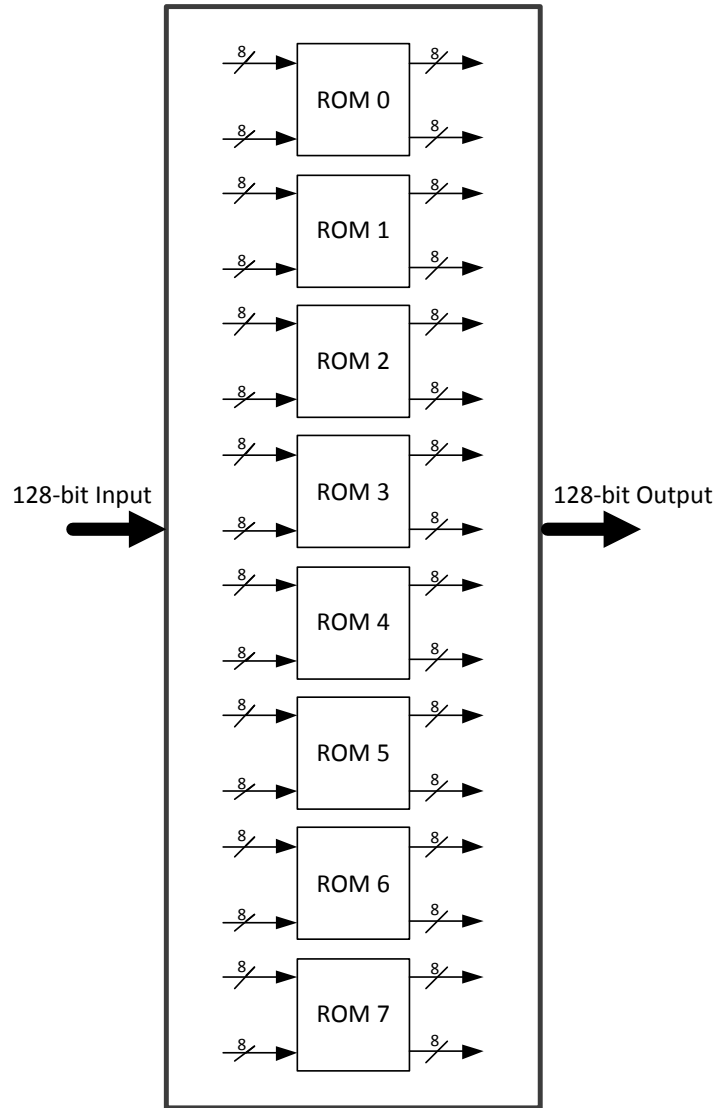


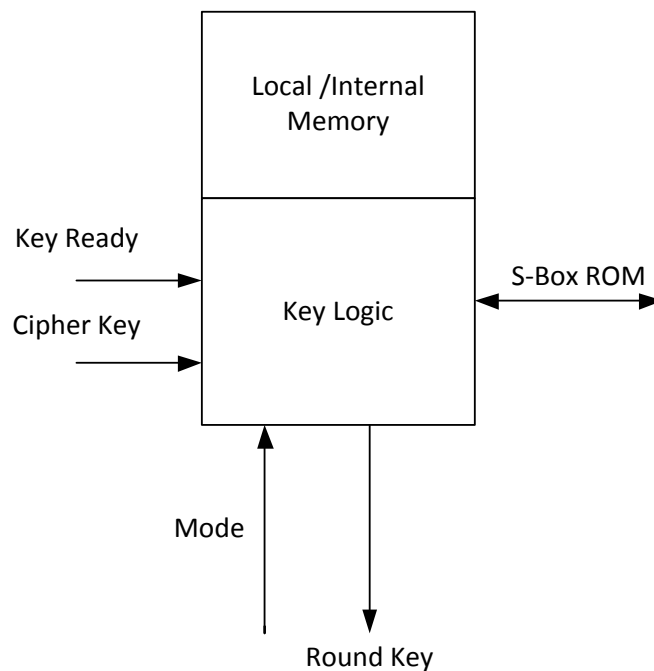
Figure 5.5: S-Box organized as 8 banks of 256 x 8 dual port ROMs.

The working of other basic operations like ShiftRows, MixColumns and AddRoundKey performed in AES is already mentioned in the previous chapters. Following Table 5.1 Shows the performance of these transformations, the system is set to maximum clock frequency of 250 MHz and clock period of 4 ns. All these transformations take only a fraction of space on FPGA as shown in Table 5.1.

Basic Operation	Clock Cycle	Number of Slices
<b>ShiftRows</b>	<b>1</b>	<b>74 (1 %)</b>
<b>MixColumns</b>	<b>1</b>	<b>197(3 %)</b>
<b>AddRoundKey</b>	<b>1</b>	<b>147(2 %)</b>

Table 5.1: Performance of Basic Operations in AES block cipher

*Key Expansion* - Key scheduling in ASC-1 encryption core caters round keys for initial phase generation (AES-128, 10 round keys) and for encryption core (AES-256, 14 round keys). There are different possible ways to generate round keys (mentioned in previous chapter). But for ASC-1 the optimal approach is “offline” or “stored key approach”[MGR, ABR]. In this approach all the round keys are calculated upon the reception of the initial cipher key and store them in an internal memory. Keys are then read from the memory whenever they are required by the core, as shown in figure 5.6. Even though this approach requires a key setup phase, for round keys processing and storing in the local memory, the calculations can be performed in parallel with the encryption core using previous key. This results in no performance penalty because of key scheduling.



*Figure 5.6: Key Logic Unit – 10 round keys are calculated for AES-128 bits for initial phase generation and 14 rounds keys for Encryption and decryption of data block. Mode controls the key size and makes sure right round key is read from local memory.*

Same as the size of an encryption block in AES, the size of each round key is 128 bits. For the sake of performance (area/delay) comparison, 128-bit & 256-bit key expansion is implemented in parallel and in basic iterative architecture. The performance and area consumption of both the architectures are shown in Table 5.2 based on behavioral simulation. The system is set to maximum clock frequency i.e., 250 Mhz and clock cycle of 4 ns. Performances of both the key expansion are shown for parallel and Iterative architectures. Based on the observation from table, number of clock cycles used to compute key expansion for AES-256 in parallel architecture is less than the clock cycles used for AES-128. This is due to the fact that the AES-128 works on single

state matrix for the key expansion of the cipher key whereas AES-256 works on two matrices for key expansion.

Performance	AES-128 Key Expansion		AES-256 Key Expansion	
	Parallel	Iterative	Parallel	Iterative
Number of Clock Cycles	15	22	12	23
Number of Slices	5000	3447	5916	5818

Table 5.2: Performance of AES-128 & AES-256 Key Expansion in Parallel and Iterative Architecture

*Advanced Encryption Standard (AES):* AES – 128 bits encryption cipher is used in initial phase for the calculation of Initialization Vector (IV) and keys used for encryption and authentication of data. Same Key ( $K_M$ ) is used to encrypt all the initial values in ECB non-feedback mode. With encryption in non-feedback mode, processing of data blocks can be performed independently from other blocks and all the blocks can be encrypted in parallel. The key expansion is performed only once and stored in the internal logic. Table 5.3 shows the throughput, latency and area used for parallel and iterative hardware architectures. The system is set to 250 Mhz and clock cycle of 4ns.

AES-128 Encryption		
Performance	Iterative	Parallel
Number of Slices	1736	15550
Number of Clock Cycles	62	30
Latency (ns)	248	120
Throughput (Gbps)	0.516	up to 32

Table 5.3: Performance of AES-128 Encryption in Parallel and Iterative Architecture

From table 5.3, a huge trade-off between area and performance of the system can be clearly seen. Number of slices used in parallel architecture is almost 9 times than iterative architecture. But on the other side, the throughput of the Iterative architecture is much lower than parallel architecture.

Figure 5.7 shows proposed architectures. The iterative architecture consists of one AES functional unit for  $n - 1$  rounds and final round functional unit for  $n$ th round. Three registers are added in the circuit, the input and intermediate register is used as buffers for inter-round transformations.

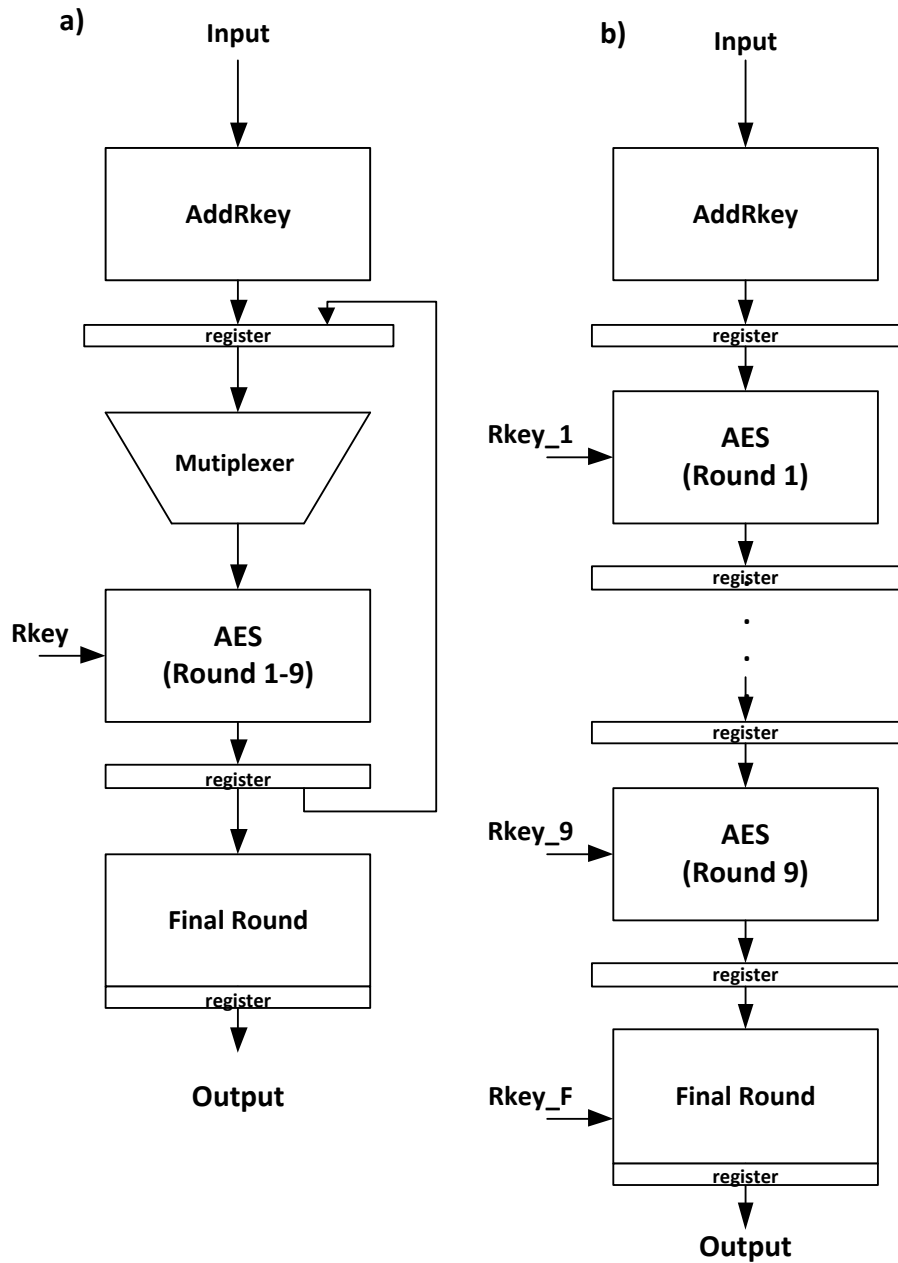


Figure 5.7: Proposed AES-128 Hardware Architectures – a) Iterative. b) Parallel Pipelined.

In figure 5.7-b, parallel pipelined AES architecture is proposed. It consists of 10 AES functional blocks and registers are placed between each AES round computation.



*ASC-1 Encryption Core:* Encryption block in ASC-1 consists of 4-Round AES and operates in Cipher feedback mode fashion to compute an authentication tag over the encrypted message. As mentioned, in feedback modes it is not possible to encrypt next block of data until encryption of previous block is completed. This left us with the choice of only encrypting the data block sequentially. The performance of ASC-1 is shown in table 5.4.

ASC-1 Encryption Core	
Performance	Iterative
Number of Slices	1796
Number of Clock cycles	41
Latency (ns)	164
Throughput (Gbps)	~0.780

*Table 5.4: Performance of ASC-1 Encryption core Iterative Architecture*

As compared to AES iterative architecture, data is processed only 4 times instead of 10 times and Initial, and final rounds are not included. As shown in figure 5.8, the order of bits transformations inside each round is also different as compared to AES; AddRound key transformation is performed at the start of each round unlike AES.

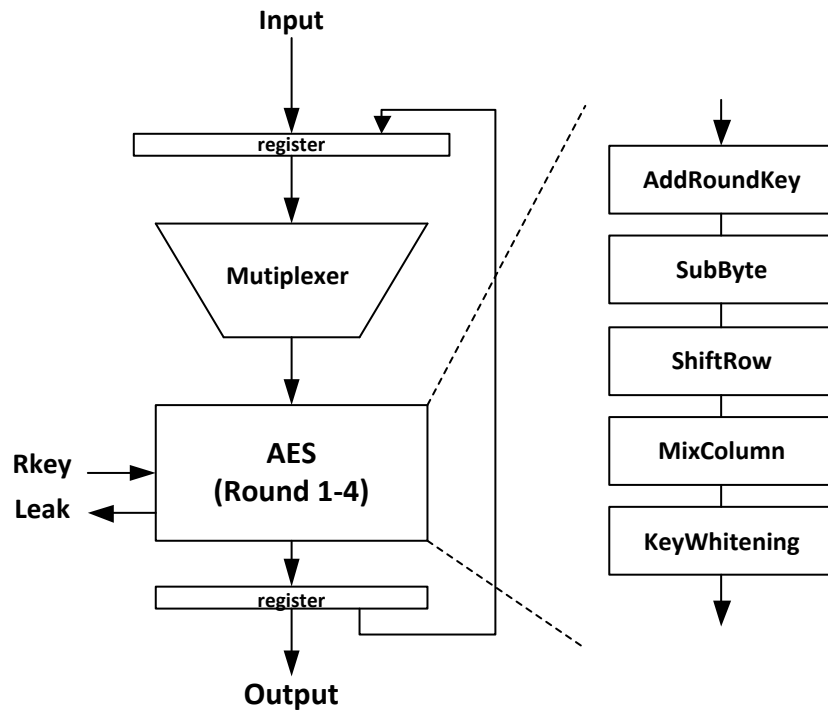
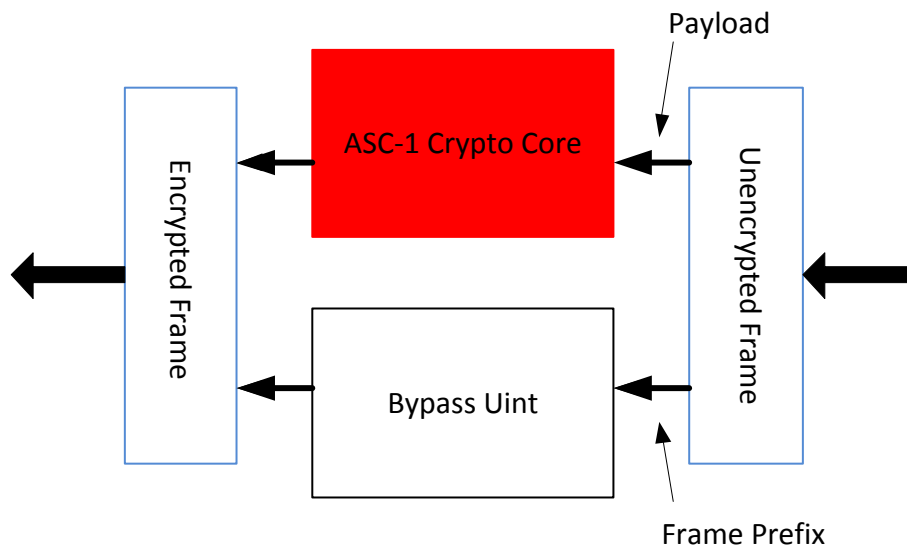


Figure 5.8: Proposed ASC-1 Iterative Hardware Architectures

## 5.4 Frame Delay

Data frame consists of preamble, Frame Alignment signal (FAS), headers and payload. As shown in figure 5.9, when the frames passes through the core, only payload have to be encrypted and rest remains in the plaintext. However to initiate the encryption of the payload requires Initial phase generation i.e. calculating initialization vector and keys for the encryption based on Counter ( $Cntr$ ) and Master Key ( $K_M$ ). The process is repeated for every frame, where Counter values are varied but Master Key remains same for the session. Two different approaches are proposed – Key setup during transmission or parallel key setup with the encryption core.



*Figure 5.9: Crypto Architecture*

In first approach, the key setup is triggered at the start of the transmission. The unencrypted prefix of the frame i.e., preamble, FAS, header etc is validated and passed though the bypass unit and waits for the encrypted and authenticated payload. Once the encryption is done, the whole frame is packed and sent to the transmitter. During the transmission of the frame, key setup for next frame is performed and stored in the logic. Based on previous sections two hardware architectures are suggested - Basic iterative and parallel architecture. Depending on the area constraints and acceptable delay for the specific applications, either of the architecture for initial phase can be chosen. Iterative architecture has a latency of 248 ns, whereas parallel architecture takes about half the time but three times in area. However in either of the architectures this approach may cause some minor end-to-end delays.

To overcome these delays, keys can also be computed in parallel with the encryption core. In this approach, initial phase is generated before the start of the transmission and keys are stored in the internal logic. For subsequent frames new keys are generated in parallel with the encryption core processing last block of the previous frame. This approach may not cause any delays but it comes with the cost of high area consumption.

## 5.5 Payload Length on effective throughput

This section shows throughput optimization by varying the payload length. Throughput in this analysis is defined as number of payload bits per second received correctly. Key assumption in the analysis is - no losses due to collision i.e., packet losses are caused only by bit errors. Figure 5.10 & 5.11, based on calculation, shows the variation of throughput with payload length at a bit error rate of  $10^{-4}$  and  $10^{-5}$  in a channel. From the figures, higher bit error rates are not shown. Due to the higher packet error rate the effective throughput leads to zero. The payload size is varied from as low as 20 bytes to 2500 bytes and extra 20 Bytes for frame prefix is added.

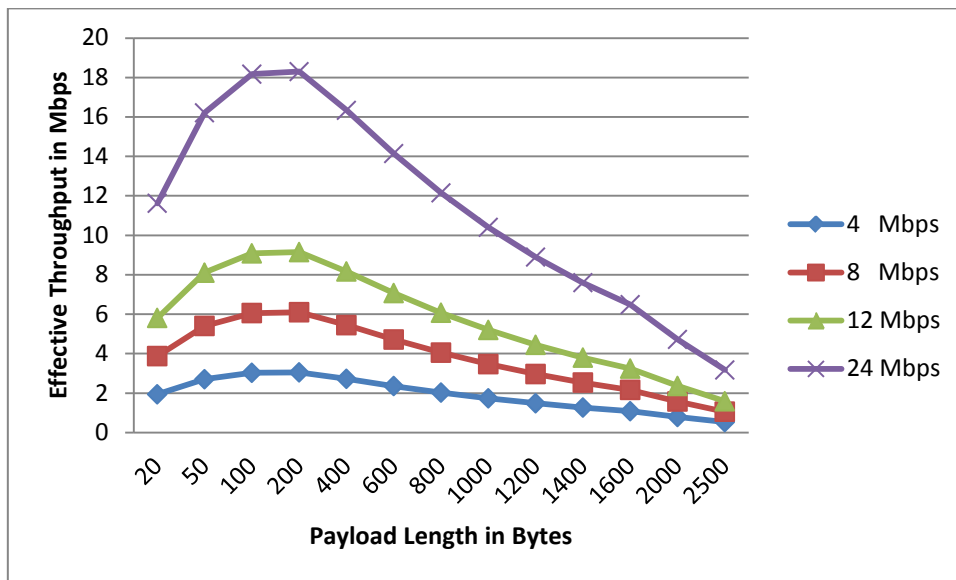


Figure 5.10: Throughput vs. Payload length at a bit error rate of  $10^{-4}$  in a channel

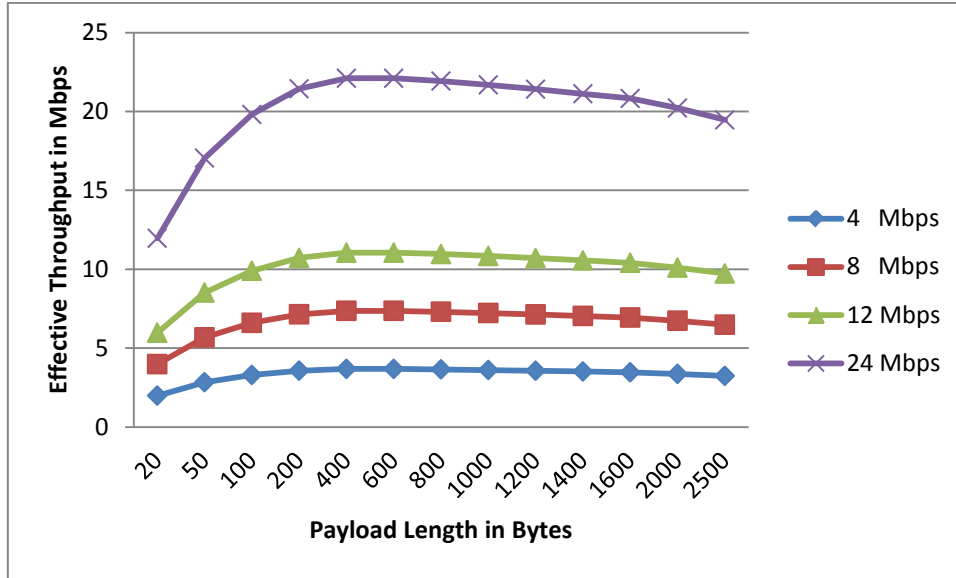


Figure 5.11: Throughput vs. Payload length at a bit error rate of  $10^{-5}$  in a channel.

The graphs identify maximum throughputs with optimal payload lengths for data rates with specified bit error rate in the channel. As shown in figure 5.10, with a bit error rate of  $10^{-4}$  corresponding to data rates at 4 , 8 and 12 Mbps, maximum channel efficiency is around the payload length of 200 bytes corresponding to an effective throughput of 3.04, 6.09 and 9.14 Mbps respectively. However reducing or increasing the payload length does not increase throughput, for example, with data rate of 8 Mbps and payload length of 20 bytes will give the effective throughput of 3.87 Mbps and with 2500 bytes the effective throughput is 1.05 Mbps. Similar results can be concluded from figure 5.11, where the maximum channel efficiency for 8 Mbps data rate is around 600 bytes corresponding to a throughput of 7.36 Mbps.

Based on the performance results of ASC-1, iterative architecture of initial key setup phase has a latency of 248 ns and the core design gives us the throughput of approximately 800 Mbps. Keeping these results in mind, in figure 5.12 we have shown the throughput vs. payload length for both the cases, i.e., with no latency and latency of 248 ns per frame. This is achieved by adding extra bits per frame representing bit length spaces i.e., 52 bits and 208 bits to the data rates at 200 and 800 Mbps respectively. Similar to our previous analysis the maximum channel efficiency is around the payload length of 200 bytes with the effective throughput of 147.33 Mbps and 534.20 Mbps. It is also interesting to notice that for higher data rate and smaller payloads the latency of 248 ns significantly affects the throughput. However, a larger payload for higher data rate or smaller data rate, the throughput is not much affected.

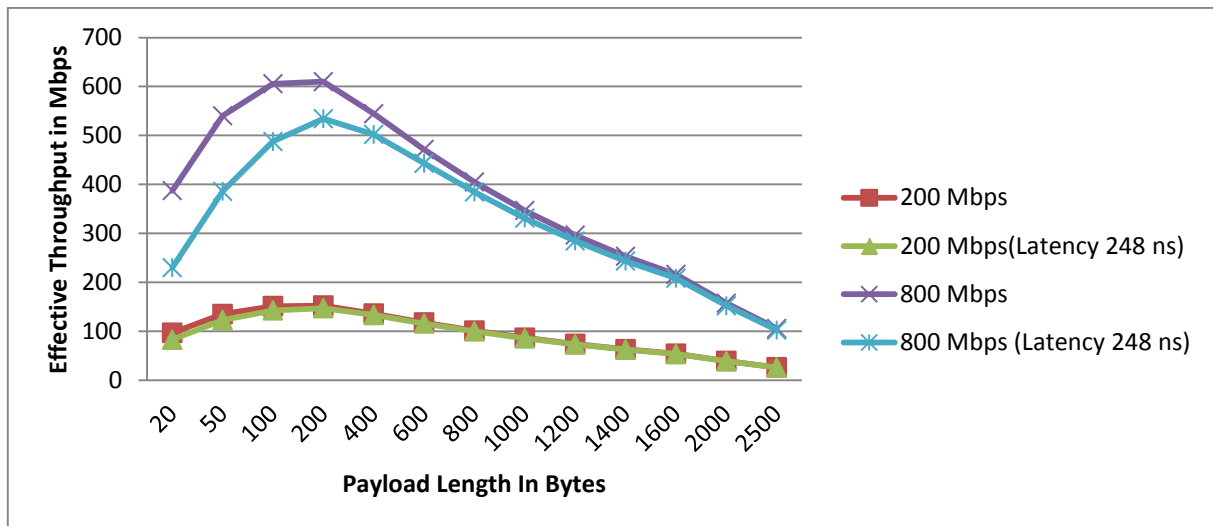


Figure 5.12: Throughput vs. Payload length at a bit error rate of  $10^{-4}$  in a channel with no latency and with latency of 248ns per frame.

## 5.6 LTE and WiMAX

Newest cellular standards like 3GPP LTE and WiMAX represent low latency, high data rate, incorporating OFDMA/MIMO, adaptive modulation and coding techniques. Deploying dedicated base terminals even for small areas is costly and time consuming process. One of the major disadvantages of implementing new base stations configured to support LTE is that existing hardware equipment will either have to be upgraded or replaced. This will boost the resources of the network that will need ongoing management and maintenance. Recently SDR approach has been adopted where it is possible to accommodate future developments and improvements in network functionality without having to replace older equipment. This is realized by remotely changing the radio standard that network components operate on by simply installing new software. Both LTE and WiMAX use OFDM as a modulation technique in their physical layer procedure. Before getting to LTE and WiMAX let us look briefly into OFDM.

Orthogonal Frequency Division Multiplexing (OFDM) is a multicarrier technique. As compared to single carrier technique high-rate data stream is transmitted on a signal channel whereas in OFDM, channel is divided into more than one channel using multiple orthogonal subcarriers for improved spectral efficiency. Similar to other modulation systems OFDM modulation system is made up of transmitter and receiver. This system is divided into four main stages [SSI]:

- Splitting data stream into many parallel data streams;
- Symbol generation;
- Converting data into time domain;
- Converting the parallel data streams back again in to serial domain digital signal;

### 5.6.1 OFDM in LTE and WiMAX

In both the technologies, the available bandwidth resource is divided in time and frequency to form smaller blocks to support many users simultaneously. These blocks are used for modulation using Orthogonal Frequency Division Multiple Access (OFDMA). The available spectrum in OFDMA is divided into number of orthogonal subcarriers with the spacing of  $\Delta f$  between them, where  $\Delta f = 15 \text{ KHz}$  for LTE and  $\Delta f = 10.94 \text{ KHz}$  for WiMAX and these subcarriers are grouped together to form a Resource block [IXP][IDA]. Fixed numbers of subcarriers are used in both the technologies i.e., 12 in LTE and 18 in WiMAX. Depending on the system configuration, the resource block is then defined in time for number of OFDM symbols i.e., 5-14 symbols. These blocks are then grouped in a frame of 10 ms for LTE and 5 ms for WiMAX. Base station decides for the allocation of resource blocks to a user for data transmission.

OFDMA have many advantages as compared to other techniques ex., best spectral efficiency, Inter symbol interference (ISI) can be minimized adjusting cyclic prefix, flat fading due to smaller spacing etc. These capabilities make OFDM ideal choice for broadband technologies. However, one of the major disadvantages of OFDM is Peak to Average Power Ratio (PAPR), which results into a need of high linear RF power amplifier. This is because when all the subcarriers are modulated and added together, the amplitude may shoot very high as compared to the average amplitude value. In order to overcome this issue, modulation technique called Single Carrier OFDMA (SC-FDMA) is used. Similar to OFDM, SC-FDMA divides the spectrum bandwidth into multiple subcarriers and maintains orthogonality between subcarriers in frequency selective channel. However unlike OFDM, SC-FDMA signal modulated onto a given subcarrier is a linear combination of all data symbols transmitted at the same time. This helps reducing the peak amplitude at the output. LTE uses SC-FDMA in the uplink to save battery power on the user equipment. Table 5.5 summarizes the main physical layer parameters for LTE and WiMAX.

Feature	3GPP LTE-Advanced	IEEE 802.16m Mobile WiMAX
Multiple Access Scheme	Downlink : OFDMA Uplink : SC-FDMA	Downlink : OFDMA Uplink : OFDMA
Physical Resource Block Size	12 sub-carriers x 14 OFDM/SCFDMA  Symbols = 168 Resource elements	18 sub-carriers x 6 OFDM  Symbols = 108 Resource elements
Usable Bandwidth at 10 Mhz	600 sub-carriers x 15 kHz (Subcarrier spacing ) = 9 MHz Spectrum Occupancy = 90%	864 sub-carriers x 10.9375 kHz (Subcarrier spacing ) = 9.45 MHz Spectrum Occupancy = 94.5%
Usable Resource Elements per 5 ms	42000 Resource Elements	44064 Resource Elements
Modulation and Coding Scheme Levels	27 Levels	32 Levels
Theoretical peak bit rate in ideal case (2 x 2 MIMO)	Uplink : ~ 85 Mbps Downlink : ~ 25 Mbps	Uplink : ~ 40 Mbps Downlink : ~ 8 Mbps

*Table 5.5 : Physical Layer Parameter for LTE and WiMAX [SAH]*

Modulation schemes like BPSk, QPSK, 16QAM or 64 QAM are used with various forward error correcting codes with varying coding rate from 1/16 to 3/4.

### 5.6.2 Confidentiality and Integrity in LTE and WiMAX

A primary security concern of a device using a radio channel, which is an open channel, is to protect the traffic confidentiality and integrity. In this section we will look into the security standards for LTE and WiMAX.

LTE uses two standardized algorithms to provide confidentiality and integrity i.e.,  $f_8, f_9$  respectively [3GPP]. The underlying block cipher for both the algorithms is KASUMI. The block cipher is based on MISTY 1 [MIM]. However, in order to increase the level of security and to meet demanding hardware implementation requirements ex., low power and low area consumption in hardware, several improvements were made in order to evolve to KASUMI. The block cipher is an eight round Feistel block cipher, inputs 64-bit data block and 128-bit key and produces 64-bit output.



Confidentiality Algorithm  $f_8$  – The  $f_8$  cryptographic algorithm is a stream cipher that is used to encrypt/decrypt the data between the length of 1 and 20,000 bits under a master key  $CK$ .  $f_8$  stream cipher uses the underlying block cipher in a Output Feedback Mode (OFB) as a keystream generator. Figure 5.13 shows the structure of confidentiality algorithm  $f_8$ .

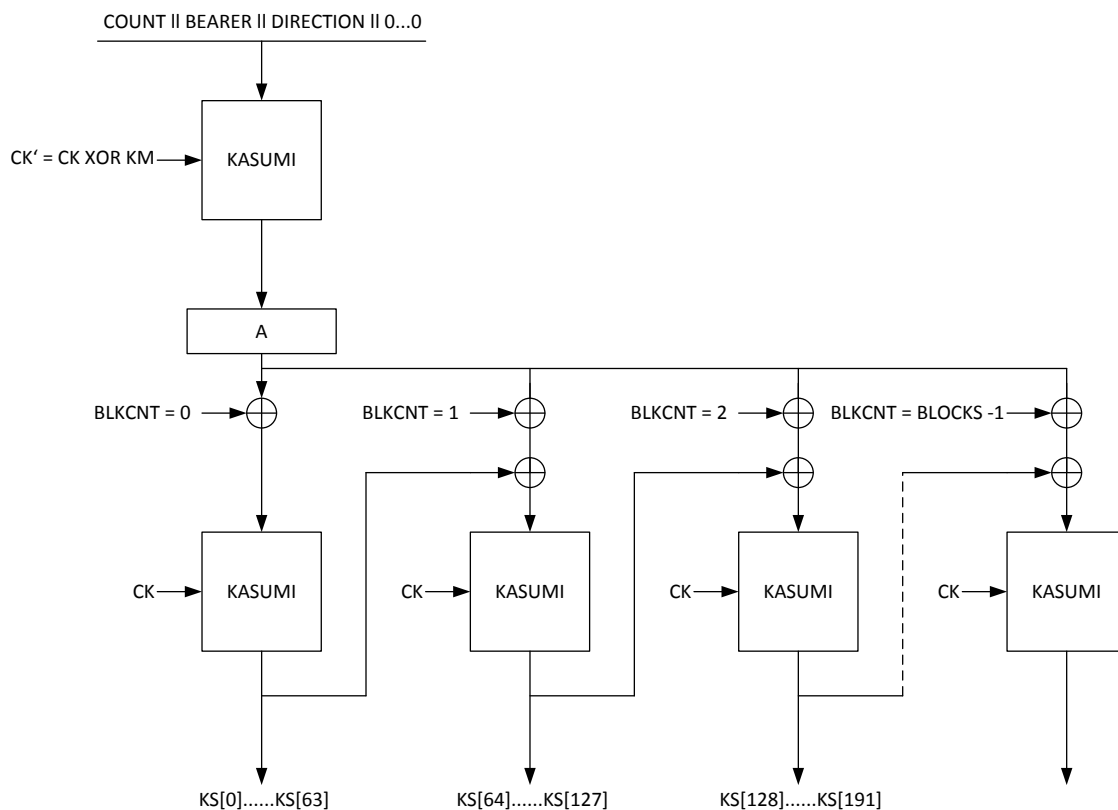


Figure 5.13: Confidentiality Algorithm  $f_8$ . BLKCNT is a 64-bit counter. COUNT, BEARER and DIRECTION are padded to become a full length datablock. Master Key  $CK$  is XOR-ed with fixed mask  $KM$  and derived key  $CK'$  is used to in initial KASUMI block cipher.

Integrity Algorithm  $f_9$  – For the authenticity of the data, the integrity algorithm computes 32-bit Message Authentication Code (MAC) on the data block under the integrity key  $IK$ . The integrity algorithm uses a variant of CBC MAC and same block used in confidentiality algorithm. Figure 5.14 shows the structure of integrity algorithm  $f_9$ .

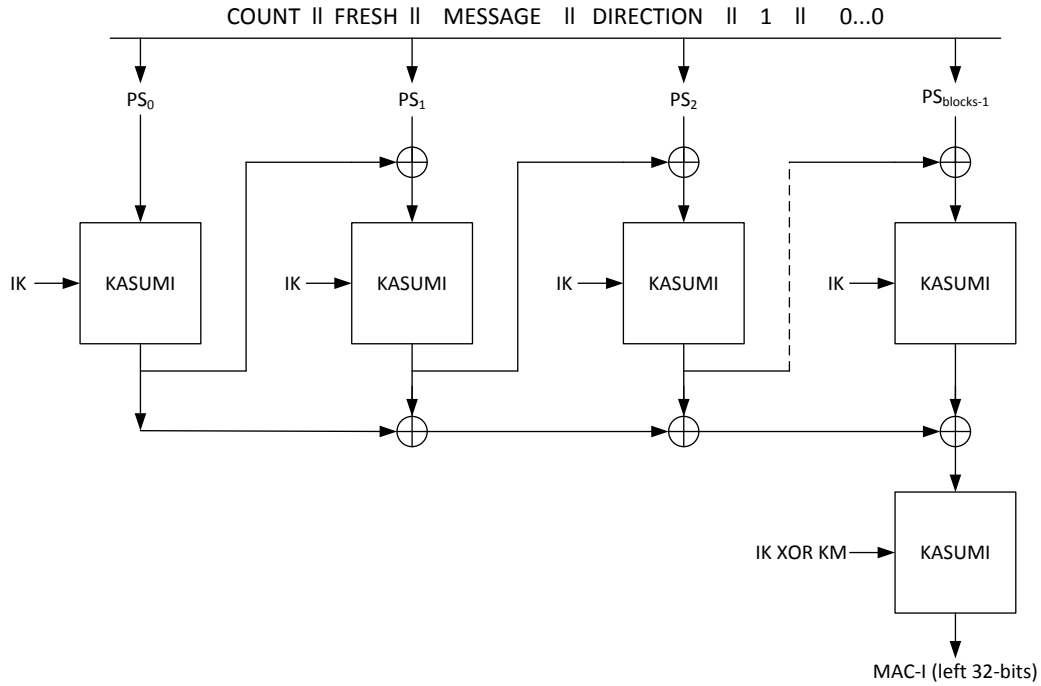


Figure 5.14: Integrity Algorithm  $f_9$ . The outputs of all the XOR-ed block computations are XOR-ed to the input of the final block computation. The leftmost 32-bits of the final output is taken as the output value MAC-I.

Unlike many other technologies, the security has been included in WiMAX system at the very start. In IEEE 802.16-2004, 802.16e and 802.16m standards, MAC layer contains a security sub-layer [LCU][KSC]. The key aspects of WiMAX security are traffic confidentiality, data authentication, key management and data integrity. The current IEEE standard supports three modes of operation for the encryption of data: CBC, Counter (CTR) and CTR with CBC message authentication code (CCM). The underlying block cipher used in WiMAX is AES[PRA]. However CTR mode is preferred over CBC as it is considered stronger and less complex to implement, it also offers encryption block preprocessing and can be implemented in parallel, which increases the throughput of the system. Combining CBC with CTR i.e., CCM adds the extra feature of verifying the authenticity of encrypted messages. CCM is considered the most secure and preferred suite for these standards because it provides integrity and confidentiality to the data.

After analyzing security design and various implementations, we can conclude that in both the cases data encryption and authentication is applied using 2-pass approach, where one pass is used for the confidentiality and other for the integrity of the data. Looking into the hardware implementations of LTE confidentiality ( $f_8$ ) and integrity ( $f_9$ ) algorithms, both the functions uses a number of cascaded KASUMI blocks in order to achieve data confidentiality and integrity. From

[AHV], maximum throughput of  $f_8$  and  $f_9$  on Xilinx vertex-E series, i.e., XCV300E-6BG432 is 240.96 Mbps with maximum clock frequency of 30.12 MHz. Whereas the proposed Authenticated Encryption scheme in this thesis, i.e., ASC-1 can achieve a throughput of approximately 800 Mbps by performing both encryption and message authentication in a single pass as opposed to above mentioned approaches.

## 5.7 Conclusions

FPGAs have become a popular target for implementing cryptographic block cipher. As well-designed FPGA solution is capable of designing some of the algorithmic flexibility equivalent to software implementation with throughputs that are comparable to custom ASIC designs.

In this chapter we have summarized the results for hardware implementation of ASC-1. Hardware implementation of such encryption schemes depends on several performance parameters like implementation area, power consumption, maximum throughput and maximum throughput to area ratio. However, in case of authenticated encryption schemes, the throughput is not a static value, but is dependent on the length of the message. Section 5.1 gives the general formula for throughput and latency. In section 5.2 we have discussed hardware architectures verses the block cipher modes of operation such as feedback and non-feedback modes. The motivation for this discussion is due to the fact that the first part of ASC-1 is calculated by using non-feedback mode and encryption part is done in feedback mode. Section 5.2.1 explained the hardware architectures for feedback modes such as basic iterative and partial/full loop unrolling architecture. In basic iterative architecture only single block cipher is implemented as a combinational logic where as in partial loop  $N$  rounds are implemented as a combinational logic. The later gives a small increase in throughput at the cost of large area penalty. In section 5.2.2, we have defined pipeline hardware architecture for non-feedback cipher modes, where the throughput of the architecture could be increased by processing blocks of data simultaneously.

In section 5.3 we have presented the overall performance of the core by discussing the functional block of the scheme. Parts of the scheme are implemented in iterative and parallel pipeline hardware architecture for the sake of comparison between the performance parameters such as area utilization and throughput. This section also shows a huge trade-off between area and performance of the system in case of encryption of Initialization vector and keys in initial phase generation. Number of slices used in parallel architecture is almost 9 times than iterative architecture, but the throughput of the iterative architecture ( $\sim 0.5$  Gbps) is much lower than parallel architecture ( $\sim 32$  Gbps). In our proposed sequential ASC-1 design, with maximum operational frequency we have achieved a throughput of approximately 0.8 Gbps and this can be suitably used for SDR applications.

Additionally, we have also explored any possible frame delay due to the initial key setup with every frame in section 5.4. Based on the available resources two different approaches are proposed – key setup during transmission and parallel key setup with the encryption core. Based on our results we assume that our proposed scheme is suitable for SDR application with negligible or no delays. In section 5.5, we showed the maximum throughput with optimal payload lengths for data rates with specified BERs in the channel.

Finally in section 5.6 we looked into newer cellular standards, LTE and WiMAX and the use of OFDM as a modulation technique in their physical layer procedure. We have also briefly discussed advantages and disadvantages of OFDM and concluded that OFDM is an ideal choice for broadband services. In the end, we have discussed security standards in LTE and WiMAX, where in both the cases encryption and authentication of data is applied using two different algorithms. Whereas our proposed scheme achieves both the security features in a one single algorithm.

## References

- [MGR] M. Grand, L. Bossuet, G. Gogniat, B. Le Gal and D. Dallet. *A Reconfigurable Crypto Sub System for the Software Communication Architecture*, in Proceedings of MILCOM 2009
- [ABR ] A. Brokalakis, A.P. Kakarountas and C.E. Goutis. A High- Throughput Area Efficient FPGA Implementation of AES-128 Encryption, In *Proceedings Of IEEE 2005 International Workshop on Signal Processing Systems (SiPS'05)*, Athens, Greece, pp. 116-121, Nov. 2–4, 2005.
- [AAZ] A. Aziz and N. Ikram. An FPGA-based AES-CCM crypto core for IEEE 802.11i architecture. *International Journal of Networks Security*, 5(2):224232, 2007.
- [JOZ] J. Zambreno, D. Nguyen and A. N. Choudhary. Exploring Area/Delay Tradeoffs in an AES FPGA Implementation, *proceedings of Field Programmable Logic and Application (FPL) 2004*, Lecture Notes in Computer Science 3203, pp. 575–585, Springer-Verlag, 2004.
- [GAJ] K. Gaj and P. Chodowiec. FPGA and ASIC Implementations of AES. Chapter 10 in C.K. Koc (Ed.), *Cryptographic Engineering*, pp. 235-320, Springer, Dec. 2008.
- [PCH] P. Chodowiec and K. Gaj. Very Compact FPGA Implementation of the AES Algorithm. In C. D. Walter, C. etin Kaya Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003*, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings, volume 2779 of Lecture Notes in Computer Science, pages 319–333. Springer, 2003.
- [BZH] B. Zhou, Y. Peng, K. Gaj and Z. Zhou. Implementation and comparative analysis of AES as a stream

cipher. In *Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on*, pp. 396-400, 2009.

[AHO]A. Hodjat, and I. Verbauwhede. *A 21.54 Gbits/s Fully Pipelined AES Processor on FPGA*, 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'04), pp. 308-309, April 2004

[CNA]C.Nalini, Nagaraj,P.V. Anandmohan and D.V. Poornaiah. An FPGA Based Performance Analysis of Pipelining and Unrolling of AES Algorithm. *Advanced Computing and Communications (ADCOM)*, IEEE Press, pp.477-482, 2006.

[CHI]C.P Fan and J.K Hwang. Implementations of high throughput sequential and fully pipelined AES processors on FPGA.*Intelligent Signal Processing and Communication Systems(ISPCS)* IEEE Press, pp.353-356,2007.

[LCU]L. Cuilan A simple encryption scheme based onWiMAX. Presented at the Int. Conf. E-Business and Information System Security, Wuhan, China, 2009.

[3GPP] 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 1: *f8* and *f9* Specification (Release 10), 3GPP TS 35.201 v10.0.0 , 2011

[KSC]K. Scarfone, C. Tibbs and M.Sexten. Guide to securing WiMAX Wireless Communication, NIST – Special Publication 800-127.

[SSI]S.Stefania, I.Toufik, and M.Baker. *LTE – The UMTS Long Term Evolution from Theory to Practice*. Hoboken: John Wiley & Sons, 2009.

[IXP]IEEE Xplore. IEEE Standard for Local and Metropolitan Area Networks Part 16: Air Interface for Broadband Wireless Access Systems Amendment 3: Advanced Air Interface.Last modified May 6, 2011.

[IDA] I.Daemon. History of the Public Switched Telephone Network (PSTN). Last modified 2005. <http://www.inetdaemon.com/tutorials/telecom/pstn/history.shtml>.

[SAH] S. Ahmadi . *Mobile WiMAX A Systems Approach to Understanding IEEE 802.16m Radio Access Technology*. Burlington: Elsevier Press, 2011.

[MIM] M. Matsui. New block encryption algorithm MISTY. In *Fast Software Encryption'97*, volume 1267 LNCS, Springer-Verlag. pp. 54-68, 1997.

[AHV] A.Satoh and S. Morioka. Small and High-Speed Hardware Architectures for the 3GPP Standard Cipher KASUMI. In *Proceedings of the 5th International Conference Information Security, ISC 2002* . LNCS vol. 2433 Springer 2002.

[PMO]P. Kitsos, M. D. Galanis and O. Koufopavlou. High-Speed Hardware Implementations of the KASUMI Block Cipher. Proceedings of the 2004 IEEE International Symposium on Circuit and Systems ISCAS'04 , Vancouver, Canada,2004.

[PRA] P. Rengaraju, C.H. Lung and A. Srinivasan. Measuring and Analyzing WiMAX Security and QoS in Testbed Experiments. Communications (ICC), 2011 IEEE International Conference. pp. 1-5. 2011.



## Summary and Future Scope

In this chapter, we will summarize the contributions of this thesis and present open problems of varying generality.

### 6.1 Contributions of This Thesis

Due to variety of wireless communication standards (LTE, WiFi, WiMax etc), we need highly flexible and interoperable communication systems. SDRs are used to meet the flexibility constraint. In order to avail this enabling technology that is applicable across a wide range of areas within the wireless infrastructure, these radios have to propose cryptographic services such as confidentiality, integrity and authentication. Therefore, integration of cryptographic services into SDR devices is essential. Following are the highlights of our contribution:

- A design of an authenticated encryption scheme, “ASC-1: An Authenticated Encryption Stream Cipher”. With it’s outside the box construction we aimed to explore new design approaches without scarifying the security and the performance.
- Security analysis of ASC-1, by showing that it is secure if one cannot distinguish the case when the round keys are uniformly random from the case when the round keys are derived by the key scheduling algorithm of ASC-1.
- Careful design and optimal implementation of ASC-1 for reconfigurable chips by analyzing optimum choice of hardware architecture for cryptosystems.
- Analysis of results based on performance parameters which are implementation area, maximum throughput and maximum throughput to area ratio.

The objective of this thesis was to design an authenticated encryption scheme with the focus to achieve high throughput and low overhead for SDRs. We worked into two very different research topics. One topic was the design of an authenticated encryption scheme that can accomplish both message secrecy and authenticity in a single cryptographic primitive. The other topic was the implementation of this design on re-configurable hardware in SDRs by closely observing the trade-off between area/throughput performance parameters.



We accomplished our first part of objective by proposing an authenticated encryption stream cipher (ASC-1) scheme that is designed using a stream cipher approach instead of a block cipher mode approach [SKG]. The goal here was to achieve faster encryption and messages authentication by performing both the encryption and message authentication in a single pass as opposed to the traditional encrypt-then-mac approach, which requires two passes. The design of the scheme uses four round AES block cipher as a building block. ASC-1 is inspired by the LEX stream cipher and similar to LEX, ASC-1 uses leak extraction from different AES rounds to compute the key material that is XOR-ed with the message and transform it into ciphertext. Block ciphers are usually built from a round function in an iterative way. Using just a single round of block and repeated use of the round function leaves patterns, which could be used to break the cipher [JBO]. Adding extra rounds certainly gives additional security to the cipher but it also increases the complexity of the encryption. Thus, the designers have to find the right trade-off between the security and performance of the cipher. In case of ASC-1, we show that the scheme is secure if one cannot distinguish the case when the round keys are uniformly random from the case when the round keys are derived by the key scheduling algorithm of ASC-1. It is not uncommon to make this assumption that the round keys are random when analyzing the security of cryptographic primitive. For instance, this assumption is always made when proving the resistance of a block cipher to linear and differential cryptanalysis. ASC-1 is divided into two parts – Initial phase generation and encryption in CFB-like mode together with the authentication of data. For initial phase 56-bit representation of a counter is used that provides a unique initialization vector and authentication key is generated using 64-bit representation of the bit-length of the message, if one substitutes a ciphertext with a different-length ciphertext, then the probability of success of this attack will be  $2^{-n}$ . Even though ASC-1 is inspired from LEX stream cipher and the leak positions are same as in LEX. However, unlike LEX, we add a whitening key byte before each extracted byte; this gives additional security to the scheme. Round keys for ASC-1 encryption scheme are derived from AES-256 key scheduling algorithm and keys used in each round are far apart in the key scheduling algorithm. In order for an attack to work one has to be able to guess the round key differences as well.

The second direction of the research in this thesis was inspired from the fact that the information security is one of the key relevant aspects of SDR, whether it is for data transmission or downloading of radio parameters or upgrades [RFL, LMI, ABR]. We have tried to accomplish this by designing and implementing ASC-1 authenticated encryption scheme on FPGAs. The target device used for this implementation is Xilinx Spartan -3 xc3s700an FPGA. The crypto module i.e., ASC-1 is placed on the re-configurable chip is responsible for the confidentiality and integrity of the data flow passing through it from both the sides.

The proposed ASC-1 architecture is divided into five functional blocks including a key logic and AES & ASC-1 encryption core. The round keys in a key logic functional block are calculated using an “offline” or “stored key” approach, where all the round keys are calculated upon the reception of the initial cipher key before the start of encryption core and stores them in a local memory. This is done because for initial phase generation, round keys are derived using the same master key to encrypt initialization vector, two initial keys for key scheduling for ASC-1 encryption and key used for the authentication tag. Also this approach works well for the encryption of data because of the key whitening operation. The encryption core block is used for performing AES-128 and ASC-1 encryption process. AES encryption core is only used for the generation of initialization vector and keys used in ASC-1.

In this thesis we have analyzed several performance parameters like Implementation area (number of slices used in FPGA), throughput and latency based on different hardware architectures for feedback and non-feedback cipher modes. The first part of ASC-1, i.e., the initial phase is calculated by using non-feedback mode, such as Electronic code book mode (ECB). Where the encryption of each block is performed independently for other blocks and all the blocks can be encrypted in parallel. The second part i.e., encryption of data is done in cipher feedback mode (CFB), where encryption of next block of data cannot be started until encryption of previous block is completed, which means all blocks must be processed sequentially. For the sake of comparison between the performance parameters such as area utilization and throughput, parts of the system are implemented in both iterative and parallel architecture. Additionally, we have also looked into any possible frame delay due to the initial key setup with every frame. Based on the available resources two different approaches are proposed – key setup during transmission and parallel key setup with the encryption core.

In conclusion, this thesis has proposed a new efficient single pass authenticated encryption stream cipher (ASC-1) for SDRs, with a goal to provide secrecy and authenticity to data transmission for reconfigurable chips.

## 6.2 Open Problems

The subject of security for SDR systems is quite broad and covers many issues. According to the wireless innovation forum, definition of a secure SDR in a broad sense is a device that maintains the integrity and privacy of the system and the information distributed across it. It includes

mechanisms to ensure accurate contact delivery to intended recipients, denial of interception by intruders, rejection of attempts to gain unauthorized access, mechanisms for configuration management of software download, and record-keeping with non-repudiation of actions taken by all participants[WIF].

Based on this definition possible direction could be to combine the proposed cryptographic scheme with other secure algorithms which provide countermeasures against unauthorized FPGA bitstream/image modification, key management issues etc. Other requirements that need to be explored for the security of the underlying hardware are integrity of the platform, downloading upgrade and secure storage of the keys. Additional issues are the challenges in dynamic spectrum access environment, security problems like spectrum misuse and selfish misbehaviors, licensed user emulation attacks, common control channel jamming etc are still prominent security threats[AEI].

In recent years, the study of block ciphers and hash functions has matured significantly. Certainly combining these two together and to find a mode of operation which can achieve both privacy and authentication using only a single pass is still quite new, many open problems still remain.

## References

- [SKG] G.Jakimoski and S.Khajuria. ASC-1: An Authenticated Encryption Stream Cipher. Selected Areas in Cryptography (SAC). 18<sup>th</sup> International Workshop, LNCS 7118, Springer, pp. 356-372, 2012.
- [JBO] J Borghoff. Cryptanalysis of Lightweight Ciphers. Ph.D Thesis. Technical University of Denmark (DTU). Dept of Mathematics. 2010
- [RFL] R. Falk, J. F. Esfahani, and M. Dillinger. Reconfigurable radio terminals - threats and security objectives *SDRF-02-I-0056*, Tech. Rep., 2002.
- [LMI] L. Michael, M. Mihaljevic, S. Haruyama, and R. Kohno. A framework for secure download for software-defined radio, *IEEE Communications Magazine*, vol. 40, no. 7, pp. 88–96, July 2002.
- [ABR] A. Brawerman, D. Blough, and B. Bing. Securing the download of radio configuration files for software defined radio devices. *In Proceedings of the International Workshop on Mobility Management & Wireless Access Protocols*, pp. 98–105, Sept. 2004.
- [WIF] Wireless Innovation Forum. International Tactical Radio Security Services API Specification. Document WINNF-09-S-0011. V 1.0.0. 2011
- [AEI] A. Fragkiadakis, E. Tragos, I. Askoxylakis. A survey on security threats and detection techniques in cognitive radio networks. In: *IEEE communications surveys and tutorials*, pp. 1–18. 2011.





## Appendix – Test Vectors

The appendix contains the example vectors for ASC-1. The vectors presented here are only for the encryption side. ASC-1 encryption phase is divided in to two parts – Initial phase generation and encryption and authentication of data. All the vectors are presented in hexadecimal notation.

### A.1 ASC-1 Preprocessing / Initial Phase

**Master Key (K)** – C6 56 82 7F C9 A7 99 17 6f 29 4C EC 6C D5 59 8B

**Counter (Cntr)** – 11 22 33 44 55 66 77 (56 bit representation)

**Initialization Vector -  $X_0$**  =  $E_K(0^{70}||00||Cntr)$   
= 1D 1C CF 6D 85 DF 31 57 51 62 11 0B D8 CF 9B C0

**Initial Key -  $K_1$**  =  $E_K(0^{70}||01||Cntr)$   
= BA AE 5B A2 5E 66 60 5E B6 30 5D 0C D2 0C 47 8B

**Initial Key -  $K_2$**  =  $E_K(0^{70}||10||Cntr)$   
= B3 BE A6 AE 65 C5 35 B2 70 CA 1D 98 32 56 FF 8D

**Length of the message -  $l(M)$**   
= 00 00 00 00 00 00 01 80 (64 bit representation)

**Authentication key -  $K_3$**  =  $E_K(l(M)||0^6||11||Cntr)$   
= 7F 05 94 5E 3D 73 26 FF 98 05 FD 7E FD F3 AF 7F

## A.2 Key Expansion

### 1. Key Expansion 256 bit - $K_1 \parallel K_2$

= BA AE 5B A2 5E 66 60 5E B6 30 5D 0C D2 0C 47 8B  
B3 BE A6 AE 65 C5 35 B2 70 CA 1D 98 32 56 FF 8D

Key Whitening Round key 2 -  $K_1$  = baae5ba2 5e66605e b6305d0c d20c478b

AES Round key 1 -  $K_2$  = b3bea6ae 65c535b2 70ca1d98 3256ff8d

AES Round key 1 -  $K_3$  = 0ab80681 54de66df e2ee3bd3 30e27c58

AES Round key 1 -  $K_4$  = b726b6c4 d2e38376 a2299eee 907f6163

AES Round key 1 -  $K_5$  = da57fde1 8e899b3e 6c67a0ed 5c85dcb5

Discarded key -  $K_6$  = fdb13011 2f52b367 8d7b2d89 1d044cea

AES Round key 2 -  $K_7$  = 2c7e7a45 a2f7e17b ce904196 92159d23

AES Round key 2 -  $K_8$  = b2e86e37 9dbadd50 10c1f0d9 0dc5bc33

AES Round key 2 -  $K_9$  = 821bb992 20ec58e9 ee7c197f 7c69845c

AES Round key 2 -  $K_{10}$  = a211317d 3fabec2d 2f6a1cf4 22afa0c7

Key Whitening Round key 1 -  $K_{11}$  = efbf7f01 cb1727e8 256b3e97 5902bacb

Discarded key -  $K_{12}$  = 6966c562 56cd294f 79a735bb 5b08957c

Round keys for key expansion -  $K_{13}$  = fbd16f38 30c648d0 15ad7647 4cafcc8c

Round keys for key expansion -  $K_{14}$  = 401f8e06 16d2a749 6f7592f2 347d078e

Discarded key -  $K_{15}$  = 44147620 74d23ef0 617f48b7 2dd0843b

### 2. Key Expansion 256 bit - $K_{13} \parallel K_{14}$

= fb d1 6f 38 30 c6 48 d0 15 ad 76 47 4c af cc 8c  
40 1f 8e 06 16 d2 a7 49 6f 75 92 f2 34 7d 07 8e

Discarded key in case of final Round -	$K_1 =$	fbd16f38	30c648d0	15ad7647	4cafcc8c
AES Round key 3 -	$K_2 =$	401f8e06	16d2a749	6f7592f23	47d078e
AES Round key 3 -	$K_3 =$	05147620	35d23ef0	207f48b76	cd0843b
AES Round key 3 -	$K_4 =$	106fd1e4	06bd76ad	69c8e45f	5db5e3d1
AES Round key 3 -	$K_5 =$	d205486c	e7d7769c	c7a83e2b	ab78ba10
Discarded key -	$K_6 =$	72d3252e	746e5383	1da6b7dc	4013540d
AES Round key 4 -	$K_7 =$	ab259f65	4cf2e9f9	8b5ad7d2	20226dc2
AES Round key 4 -	$K_8 =$	c540190b	b12e4a88	ac88fd54	ec9ba959
AES Round key 4 -	$K_9 =$	b7f654ab	fb04bd52	705e6a80	507c0742
AES Round key 4 -	$K_{10} =$	9650dc27	277e96af	8bf66bfb	676dc2a2
Key Whitening Round key 3 -	$K_{11} =$	9bd36e2e	60d7d37c	1089b9fc	40f5bebe
Discarded key -	$K_{12} =$	9fb67289	b8c8e4263	33e8fdd	54534d7f
Discarded key in case of final Round	$K_{13} =$	5630bc0e	36e76f722	66ed68e	669b6830
Discarded key in case of final Round	$K_{14} =$	aca2378d	146ad3ab	27545c76	73071109
Discarded key -	$K_{15} =$	d3b2bd81	e555d2f3	c33b047d	a5a06c4d

### A.3 ASC-1 Encryption

**Message**  $M = m_1, m_2, m_3$

$m_1 = 00112233\ 44556677\ 8899aabb\ ccddeeff$

$m_2 = 11112222\ 44447777\ 9999aaaa\ ccccffff$

$m_3 = 00002222\ 33336666\ 8888bbbb\ eeeeffff$



### **Round 1<sub>1</sub>**

Initialization vector ( $X_0$ ) = 1D1CCF6D 85DF3157 5162110B D8CF9BC0

Round 1: AddRoundKey ( $X_0 \oplus K_2$ ) = AEA269C3E01A04E521A80C93EA99644D

Round 1: SubBytes = E43AF92E E1A2F2D9 FDC2FEDC 87EE43E3

Round 1: ShiftRows = E4A2FEE3 E1C2432E FDEEF9D9 873AF2DC

Round 1: MixColumns = 33419FB6 E995D7E5 E8F38AA2 75223DF9

Round 1: KeyWhitening  $K_{11}$  (Odd Round) = D8 41 64 b6 e9 95 d7 e5 97 f3 39 a2 75 22 3d f9

Round 1 : LeakKey ( $l_{1...4}$ ) = D8649739

---

### **Round 1<sub>2</sub>**

**Input Round 2** = D8 41 64 b6 e9 95 d7 e5 97 f3 39 a2 75 22 3d f9

Round 2 : AddRoundKey (Input Round 2  $\oplus K_3$ )  
= D2 F9 62 37 BD 4B B1 3A 75 1D 02 71 45 C0 41 A1

Round 2 : SubBytes = B5 99 AA 9A 7A B3 C8 80 9D A4 77 A3 6E BA 83 32

Round 2 : ShiftRows = B5 B3 77 32 7A A4 83 9A 9D BA AA 80 6E 99 C8 A3

Round 2 : MixColumns = FA 63 BE 64 1A 2D 76 86 DE 97 F3 B7 07 A7 82 BE

Round 2 : KeyWhitening  $K_{11}$  (Even Round)  
= FA 63 BE 64 D1 2D 61 86 DE 97 F3 B7 20 A7 6A BE

Round 2 : LeakKey ( $l_{5...8}$ ) = D161206A

---

### **Round 1<sub>3</sub>**

**Input Round 3** = FA 63 BE 64 D1 2D 61 86 DE 97 F3 B7 20 A7 6A BE

Round 3 : AddRoundKey (Input Round 3  $\oplus K_4$ )  
= 4D 45 08 A0 03 C3 E2 F0 7C BE 6D 59 B0 D8 0B D0

Round 3 : SubBytes = E3 6E 30 E0 7B 8B 98 8C 10 AE 3C CB E7 61 2B 70

Round 3 : ShiftRows = E3 8B 3C 70 7B AE 2B E0 10 61 30 8C E7 6E 98 CB

Round 3 : MixColumns = 17 DA 80 69 D4 A1 B8 D3 3F 0E 9E 62 34 43 E4 49

Round 3 : KeyWhitenting  $K_{11}$  (Odd Round)

= 32 DA EB 69 D4 A1 B8 D3 01 0E 09 62 34 43 E4 49

Round 3 : LeakKey ( $l_{9...12}$ ) = 32EB0109

---

#### **Round 1<sub>4</sub>**

**Input Round 4** = 32 DA EB 69 D4 A1 B8 D3 01 0E 09 62 34 43 E4 49

Round 4 : AddRoundKey (Input Round 4  $\oplus K_5$ ) = e8 8d 16 88 5a 28 23 ed 6d 69 a9 8f 68 c6 38 fc

Round 4 : SubBytes = 9b 5d 47 c4 be 34 26 55 3c f9 d3 73 45 b4 07 b0

Round 4 : ShiftRows = 9b 34 d3 b0 be f9 07 c4 3c b4 47 55 45 5d 26 73

Round 4 : MixColumns = 12 2d d9 2a b4 9a 1e b4 ad d3 f9 1d 38 e6 c1 52

Round 4 : keyWhitenting  $K_{11}$  (Even Round) = 12 2d d9 2a ED 9a 1C b4 ad d3 f9 1d 82 e6 0A 52

Round 4 : LeakKey ( $l_{13...16}$ ):: ED1C820A

---

**Leak  $r_1$**  =  $l_{1...4} || l_{5...8} || l_{9...12} || l_{13...16}$  = D8649739 D161206A 32EB0109 ED1C820A

**Message  $m_1$**  = 00112233 44556677 8899aabb ccddeeff

**Ciphertext  $c_1$**  = D8 75 B5 0A 95 34 46 1D BA 72 AB B2 21 C1 6C F5

---

#### **Round 2<sub>1</sub>**

**$m_2$**  = 11112222 44447777 9999aaaa cccfffff

**Input round 1 :: (Round 1<sub>4</sub> output  $\oplus c_1$ )** = CA 58 6C 20 78 AE 5A A9 17 A1 52 AF A3 27 66 A7

Round 1: AddRoundKey (Input round 1  $\oplus K_7$ ) = E6 26 16 65 DA 59 BB D2 D9 31 13 39 31 32 FB 84

Round 1: SubBytes = 8e f7 47 4d 57 cb ea b5 35 c7 7d 12 c7 23 0f 5f

Round 1: ShiftRows = 8E CB 7D 5F 57 C7 0f 4d 35 23 47 b5 c7 f7 ea 12

Round 1: MixColumns = 63 db 5e 81 b3 9e 59 ab fd 0f 5c 4a 6f 05 c9 6b

Round 1: KeyWhitening  $K_1$  (Odd Round) = D9 db FO 81 b3 9e 59 ab A6 0f FE 4a 6f 05 c9 6b

---

Round 1 : LeakKey ( $l_{1...4}$ ) = D9 F0 A6 FE

---

### **Round 2<sub>2</sub>**

**Input round 2** = D9 db F0 81 b3 9e 59 ab A6 0f FE 4a 6f 05 c9 6b

Round 2 :: AddRoundKey (Input round 2  $\oplus K_8$ ) = 6B 33 9E B6 2E 24 84 FB B6 CE 0E 93 62 C0 75 58

Round 2: SubBytes = 7f c3 0b 4e 31 36 5f 0f 4e 8b ab dc aa ba 9d 6a

Round 2: ShiftRows = 7f 36 ab 6a 31 8b 9d 4e 4e ba 0b 0f aa c3 5f dc

Round 2: MixColumns = 65 9f ba c8 37 ce 49 d9 4d 33 f3 7d 92 0a a8 da

Round 2: KeyWhitening  $K_1$  (Even Round) = 65 9f ba c8 **69** ce **2F** d9 4d 33 f3 7d **F2** 0a **F6** da

Round 2 : LeakKey ( $l_{5...8}$ ) = 69 2F F2 F6

---

### **Round 2<sub>3</sub>**

**Input round 3** = 65 9f ba c8 69 ce 2F d9 4d 33 f3 7d F2 0a F6 da

Round 3 :: AddRoundKey (Input round 3  $\oplus K_9$ ) = E7 84 03 5A 49 22 77 30 A3 4F EA 02 8E 63 72 86

Round 3: SubBytes = 94 5f 7b b3 3b 93 f5 04 0a 84 87 77 19 fb 40 44

Round 3: ShiftRows = 94 93 87 44 3b 84 40 be 0a fb 7b 04 19 5f f5 77

Round 3: MixColumns = 5e 7f de 3b 1f 56 e6 ee 7d 6e 0b 96 51 d4 2e 6f

Round 3: KeyWhitening  $K_1$  (odd Round) = **E8** 7f **EE** 3b 1f 56 e6 ee **20** 6e **07** 96 51 d4 2e 6f

Round 3 : LeakKey ( $l_{9...12}$ ) = E8 EE 20 07

---

### **Round 2<sub>4</sub>**

**Input round 4** = E8 7f EE 3b 1f 56 e6 ee 20 6e 07 96 51 d4 2e 6f

Round 4 :: AddRoundKey (Input round 4  $\oplus K_{10}$ ) = 4A 6E DF 46 20 FD 0A C3 0F 04 1B 62 73 7B 8E A8

Round 4: SubBytes = d6 9f 93 5a b7 54 67 23 76 f2 af aa 8f 21 19 c2

Round 4: ShiftRows = d6 54 af c2 b7 f2 19 5a 76 21 9e 2e 8f 9f 67 aa

Round 4: MixColumns = 26 56 9a 05 3b 39 99 9d 3f a3 02 79 72 a9 3b 3d

Round 4: KeyWhitening  $K_1$  (Even Round) = 26 56 9a 05 E9 39 95 9d 3f a3 02 79 35 a9 B0 3d

Round 4 : LeakKey ( $l_{13...16}$ ) = E9 95 35 B0

---

**Leak  $r_2 = l_{1...4} || l_{5...8} || l_{9...12} || l_{13...16} =$**  D9 F0 A6 FE 69 2F F2 F6 E8 EE 20 07 E9 95 35 B0

**Message  $m_2 =$**  11 11 22 22 44 44 77 77 99 99 aa aa cc cc ff ff

---

**Ciphertext  $c_2 =$**  C8 E1 84 DC 2D 6B 85 81 71 77 8A AD 25 59 CA 4F

---

### Round 3<sub>1</sub>

$m_3 =$  00002222 33336666 8888bbbb eeeeefff

**Input round 1 :: (Round 2<sub>4</sub> output  $\oplus c_2$ ) =** EE B7 1E D9 C4 52 10 1C 4E D4 88 D4 10 F0 7A 72

Round 1: AddRoundKey (Input round 1  $\oplus K_2$ ) = AE A8 90 DF D2 80 B7 55 21 A1 1A 26 24 8D 7D FC

Round 1: SubBytes = E4 C2 60 9E B5 CD A9 FC FD 32 A2 F7 36 5D FF B0

Round 1: ShiftRows = E4 CD A2 B0 B5 32 FF 9E FD 5D 60 FC 36 C2 A9 F7

Round 1: MixColumns = 8D 28 BD 23 46 55 DB 2E 9A 1B 7F C2 6F B3 BF C4

Round 1: KeyWhitening  $K_{11}$  (Odd Round) = 16 28 6E 23 46 55 DB 2E F4 1B 51 C2 6F B3 BF C4

Round 1 : LeakKey ( $l_{1...4}$ ) = 16 6E F4 51

---

### Round 3<sub>2</sub>

**Input round 2 =** 16 28 6E 23 46 55 DB 2E F4 1B 51 C2 6F B3 BF C4

Round 2 : AddRoundKey (Input round 2  $\oplus K_3$ ) = 13 3C 18 03 73 87 E5 DE D4 64 19 75 03 63 3B FF

Round 2 : SubBytes = 7D EB AD 7B 8F 17 D9 1D 48 43 D4 9D 7B FB E2 16

Round 2 : ShiftRows = 7D 17 D4 16 8F 43 E2 7B 48 FB AD 1D 7B EB D9 9D

Round 2 : MixColumns = 01 22 E3 68 59 4F 9E DD 36 54 D5 B4 94 5B 85 9E

Round 2 : KeyWhitening  $K_{11}$  (Even Round) = 01 22 E3 68 39 4F 49 DD 36 54 D5 B4 47 5B F9 9E

Round 2 : LeakKey ( $l_{5...8}$ ) = 39 49 47 F9

---

### Round 3<sub>3</sub>

**Input round 3** = 01 22 E3 68 39 4F 49 DD 36 54 D5 B4 47 5B F9 9E

Round 3 : AddRoundKey (Input round 3  $\oplus K_4$ ) = 11 4D 32 8C 3F F2 3F 70 5F 9C 31 EB 1A EE 1A 4F

Round 3 : SubBytes = 82 E3 23 64 75 89 75 51 CF DE C7 E9 A2 28 A2 84

Round 3 : ShiftRows = 82 89 C7 84 75 D3 A2 64 CF 28 23 51 A2 E3 75 E9

Round 3 : MixColumns = DC 5D 09 C0 55 4B 58 2B 8F AB 52 E3 FD 09 8B A2

Round 3 : KeyWhitening  $K_{11}$  (odd Round) = CC 5D 80 C0 55 4B 58 2B 36 AB AE E3 FD 09 8B A2

Round 3 : LeakKey ( $l_{9...12}$ ) = CC 80 36 AE

---

### Round 3<sub>4</sub>

**Input round 4** = CC 5D 80 C0 55 4B 58 2B 36 AB AE E3 FD 09 8B A2

Round 4 : AddRoundKey (Input round 4  $\oplus K_5$ ) = 1E 58 C8 AC B2 9C 2E B7 F1 03 90 C8 56 71 31 B2

Round 4 : SubBytes = 72 6A E8 91 37 DE 31 A9 A1 7B 60 E8 B1 A3 C7 37

Round 4 : ShiftRows = 72 DE 60 37 37 7B C7 91 A1 A3 E8 A9 B1 6A 31 E8

Round 4 : MixColumns = CA 42 35 46 B5 02 71 DC E6 76 29 FA 1E DE 9A 58

Round 4 : KeyWhitening  $K_{11}$  (Even Round) = CA 42 35 46 F5 02 84 DC E6 76 29 FA A0 DE 24 58

Round 4 : LeakKey ( $l_{13...16}$ ) = F5 84 A0 24

---

**Leak  $r_3$**  =  $l_{1...4} || l_{5...8} || l_{9...12} || l_{13...16}$  = 16 6E F4 51 39 49 47 F9 CC 80 36 AE F5 84 A0 24

**Message  $m_3$**  = 00002222 33336666 8888bbbb eeeeffff

**Ciphertext  $c_3$**  = 16 6E D6 73 0A 7A 21 9F 44 08 8D 15 1B 6A 5F DB

---

### Round 4<sub>1</sub> (Authentication Round)

**Input round 1 :: (Round 3<sub>4</sub> output  $\oplus c_3$ )** = DC 2C E3 35 FF 78 A5 43 A2 7E A4 EF BB B4 7B 83

Round 1: AddRoundKey (Input round 1  $\oplus K_7$ ) = 77 09 7C 50 B3 8A 4C BA 29 24 73 3D 9B 96 16 41

Round 1: SubBytes = F5 01 10 53 6D 7E 29 F4 A5 A5 8F 27 14 90 47 83

Round 1: ShiftRows = F5 7E 8F 83 6D A5 47 53 A5 90 10 F4 12 01 29 27

Round 1: MixColumns = 7F 00 10 E8 3A A6 B3 F3 1E 5A 12 87 25 4A 2E 5A

---

#### **Round 4<sub>2</sub>**

Input Round 2 : 7F 00 10 E8 3A A6 B3 F3 1E 5A 12 87 25 4A 2E 5A

Round 2 : AddRoundKey (Input round 2  $\oplus K_8$ ) = BA 40 09 E3 8B 88 F9 7B B2 D2 EF D3 C9 D1 87 03

Round 2 : SubBytes = F4 09 01 11 3D C4 99 21 37 B5 DF 66 DD 3E 17 7B

Round 2 : ShiftRows = F4 C4 DF 7B 3D B5 17 11 37 3E 01 21 DD 09 99 66

Round 2 : MixColumns = 00 66 18 EA B8 64 95 C7 0C 69 68 24 45 19 57 20

---

#### **Round 4<sub>3</sub>**

Input Round 3 = 00 66 18 EA B8 64 95 C7 0C 69 68 24 45 19 57 20

Round 3 : AddRoundKey (Input round 3  $\oplus K_9$ ) = B7 90 4C 41 43 60 28 95 7C 37 02 A4 15 65 50 62

Round 3 : SubBytes = A9 60 29 83 1A D0 34 2A 10 9A 77 49 59 4D 53 AA

Round 3 : ShiftRows = a9 d0 77 aa 1a 9a 53 83 10 4d 29 2a 59 60 34 49

Round 3 : MixColumns = FF 21 72 08 51 43 B8 FA F4 DB 71 00 6F 8C 8A 2D

---

#### **Round 4<sub>4</sub>**

Input Round 4 = FF 21 72 08 51 43 B8 FA F4 DB 71 00 6F 8C 8A 2D

Round 4 : AddRoundKey (Input round 4  $\oplus K_{10}$ ) = 69 71 AE 2F 76 3D 2E 55 7F 2D 1A FB 08 E1 48 8F

Round 4 : SubBytes = F9 A3 E4 15 38 27 31 FC D2 D8 A2 0F 30 F8 52 73

Round 4 : ShiftRows = F9 27 A2 73 38 D8 52 15 D2 F8 E4 FC 30 A3 31 0F

Round 4 : MixColumns = 51 39 14 73 44 70 7B E8 B4 F2 E6 92 A0 31 E0 DC

---

Authentication Tag  $\tau$  (Output round 4<sub>4</sub>  $\oplus$  Authentication Key  $K_3$ )

= 2E 3C 80 2D 79 03 5D 17 2C F7 1B EC 5D C2 4F A3

<b>Ciphertext <math>C = IV/Cntr    C_1    C_2    C_3    \tau</math></b>
---

# II

## Appendix – Xilinx Sample Code and Waveforms

The appendix contains parts of the VHDL code implemented in parallel and iterative hardware architecture. Additionally simulation waveforms are all shown for ASE-128, Key expansion and ASC-1 encryption core.

Language	VHDL (VHSIC hardware description language)
VHDL Source Analysis Standard	VHDL-93
Product Category	
• Family	Spartan 3AN
• Device	XC3700AN
• Package	FGG484
• Speed	-4
Synthesis Tool	XST – VHDL/Verilog
Simulator	ISim
Software Version	Xilinx ISE – 12.4

### B.1 Advanced Encryption Standard (AES) – 128

The block cipher is used in initial phase for the calculation of Initialization Vector (IV) and keys used for encryption and authentication of data. For performance comparison AES-128 encryption is implemented in parallel and in basic iterative architectures.

Top Module for AES- Encryption basic iterative architecture is implemented using the case statements. Each statement is executed when one specific case of an expression equal to a choice. Following shows the sample code for the case statements used in AES encryption module.



```
case algState is
```

#### **IDLE State:**

```
when IDLE =>
    ctValidDly   <= '0';
    if cnt < RND1KEYDELAY then
        cnt <= cnt + 1;
    else
        cnt      <= 0;
        sInNxt   <= sInNR(1);
        algState <= rnd1;
    end if;
```

#### **Round 1:**

```
when rnd1 =>
    W0nxt <= W(4);
    W1nxt <= W(5);
    W2nxt <= W(6);
    W3nxt <= W(7);

    if cnt < RNDDELAY then
        cnt <= cnt + 1;
    else
        cnt      <= 0;
        algState <= rnd2;
        sInNxt   <= sOutNxt;
    end if;
```

#### **Round 2:**

```
when rnd2 =>
    W0nxt <= W(8);
    W1nxt <= W(9);
    W2nxt <= W(10);
    W3nxt <= W(11);

    if cnt < RNDDELAY then
        cnt <= cnt + 1;
    else
        cnt      <= 0;
        algState <= rnd3;
        sInNxt   <= sOutNxt;
    end if;
```

```
.
.
.
.
```

#### **Round 9:**

```
when rnd9 =>
    W0nxt <= W(36);
    W1nxt <= W(37);
    W2nxt <= W(38);
    W3nxt <= W(39);

    if cnt < RNDDELAY then
        cnt <= cnt + 1;
    else
        cnt      <= 0;
        sInNR(10) <= sOutNxt;
    end if;
```

#### **Round 10 (Final Round):**

```
finalRoundSelect : process (W, sInNR) is
begin
    WFinal0 <= W(40);
    WFinal1 <= W(41);
    WFinal2 <= W(42);
    WFinal3 <= W(43);
    sInFinal <= sInNR(10);
end process;

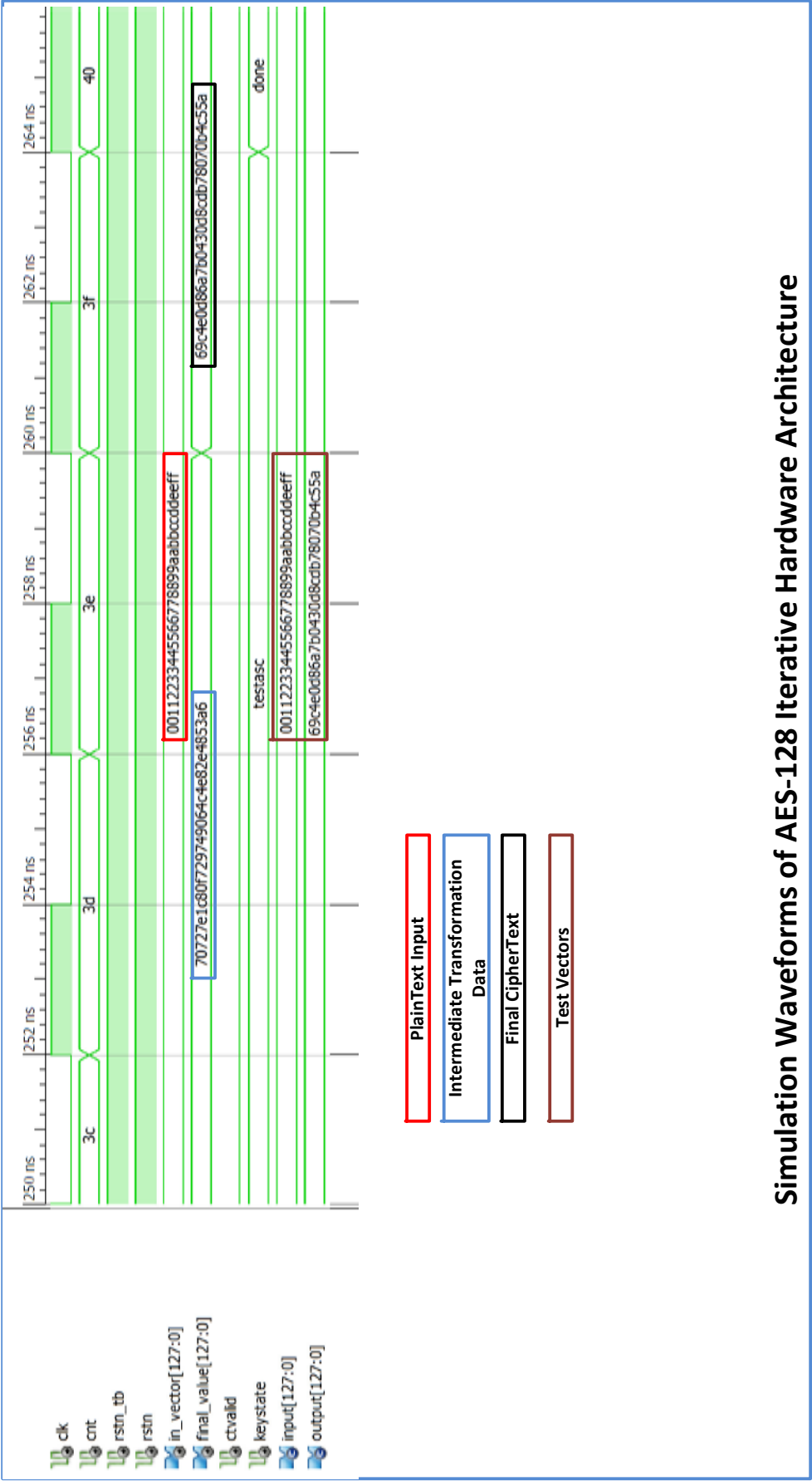
finalround_1 : finalround
port map (
    clk      => clk,
    rstn     => rstn,
```

The case is IDLE when the top module is waiting for the Key Schedule to be completed. Once all the round keys are calculated using the key scheduling the state moves to next case, i.e., rnd1

From round 1 to round 9, round keys and output of previous state is fed into the intermediate states. Except the final round same operation is performed in each round, i.e., SubBytes, ShiftRows, MixColumns and AddRoundKey.

```
W0      => WFinal0,  
W1      => WFinal1,  
W2      => WFinal2,  
W3      => WFinal3,  
stateIn => sInFinal,  
stateOut => final);
```

---



Simulation Waveforms of AES-128 Iterative Hardware Architecture

### Sample code for AES-128 parallel Architecture:

```
addroundkey_1: addroundkey
  port map (
    clk      => clk,
    rstn     => rstn,
    W0       => W(0),
    W1       => W(1),
    W2       => W(2),
    W3       => W(3),
    stateIn  => sIn,
    stateOut => sInNR(1));
```

At Round 0, key is Xor-ed with the initial state i.e., Plaintext

#### **Generating Rounds 1 to 9**

```
genRounds: for Nr in 1 to 9 generate
  round_NR: round
    port map (
      clk      => clk,
      rstn     => rstn,
      W0       => W(4+((Nr-1)*4)),
      W1       => W(5+((Nr-1)*4)),
      W2       => W(6+((Nr-1)*4)),
      W3       => W(7+((Nr-1)*4)),
      stateIn  => sInNR(Nr),
      stateOut => sOutNR(Nr));
```

All 9 rounds are generated using parallel pipeline architecture

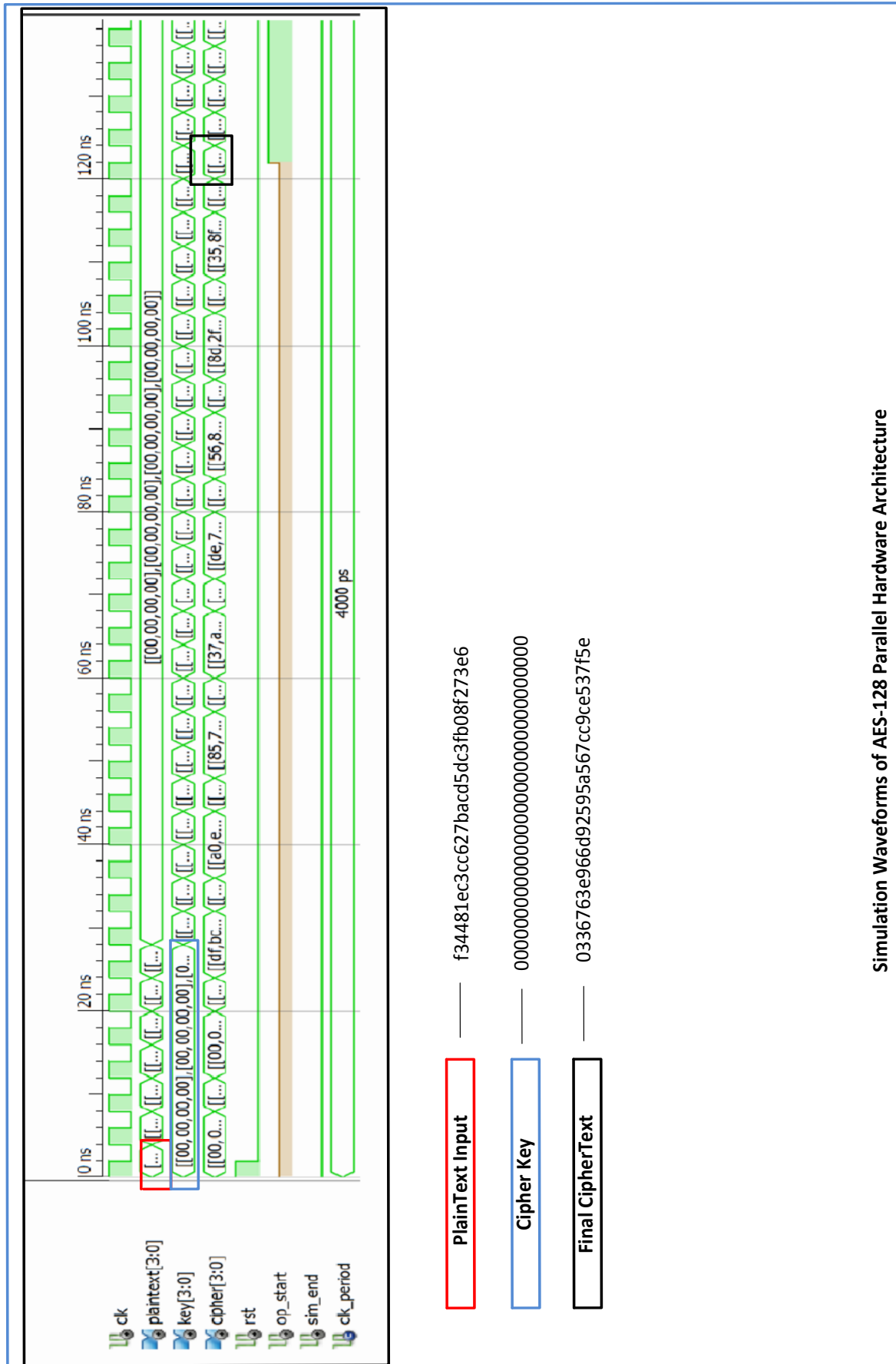
```
    sInNR(Nr+1) <= sOutNR(Nr);
  end generate genRounds;
```

#### **Round 10 (Final Round) :**

```
finalRoundSelect: process (W, sInNR) is
  begin
    WFinal0 <= W(40);
    WFinal1 <= W(41);
    WFinal2 <= W(42);
    WFinal3 <= W(43);
    sInFinal <= sInNR(10);
  end process finalRoundSelect;
```

Finally, round 10 is transformed separately

```
finalround_1: finalround
  port map (
    clk      => clk,
    rstn     => rstn,
    W0       => WFinal0,
    W1       => WFinal1,
    W2       => WFinal2,
    W3       => WFinal3,
    stateIn  => sInFinal,
    stateOut => sOut);
```



## B.2 Key Expansion

Two sets of key scheduling is performed for ASC-1 encryption core, one for the initial phase generation where 10 rounds keys are derived from the key scheduling algorithm for AES-128 and 14 round keys are derived for ASC-1 encryption/decryption. Similar to AES implementation, for performance analysis both the key scheduling are implemented in Parallel and iterative architectures. However, results are shown only for AES-128 key expansion.

### Sample code for AES-128 key expansion iterative architecture .

```
w(0) <= keyInReg(31 downto 0);  
w(1) <= keyInReg(63 downto 32);  
w(2) <= keyInReg(95 downto 64);  
w(3) <= keyInReg(127 downto 96);
```

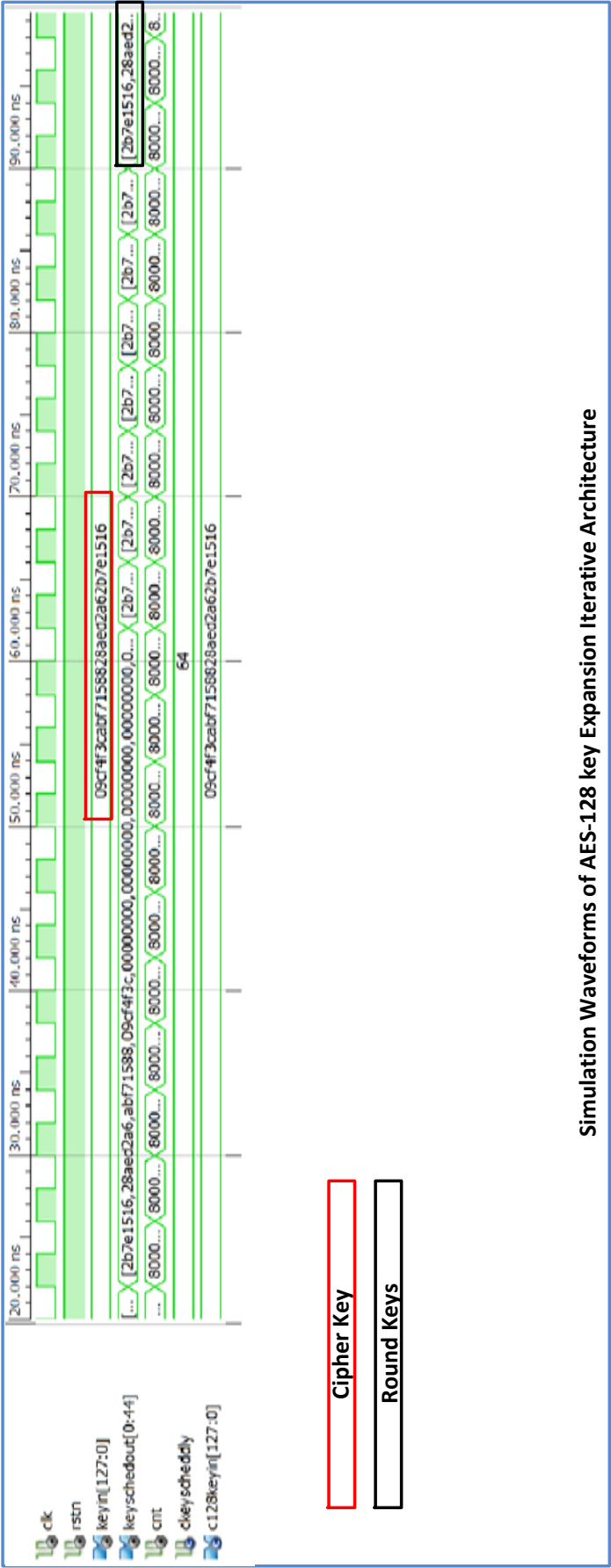
128 bit cipher keys is loaded into the registers w(0) ... w(4)

```
j <= 0;  
r <= 1;  
i <= 3;  
  
if cnt < RND1KEYLOAD + 3 then  
    cnt <= cnt + 1;  
else  
    cnt <= 0;  
    keyState <= itr1;  
end if;
```

---

```
when itr1 =>  
    w(j + 4) <= Wout;  
    w(j + 5) <= Wout xor w1;  
    w(j + 6) <= Wout xor w1 xor w2;  
    w(j + 7) <= Wout xor w1 xor w2 xor w3;  
  
    i <= i + 4;  
    j <= j + 4;  
    r <= r + 1;  
  
    if cnt < 9 then  
        cnt <= cnt + 1;  
    else  
        cnt <= 0;  
        keyState <= DONE;  
    end if;
```

One single round of block is implemented as a combinational logic and is repeated 9 times .



Simulation Waveforms of AES-128 key Expansion Iterative Architecture

## Sample code for AES-128 Parallel Architecture

```
w(0) <= keyInReg(31  downto 0);  
w(1) <= keyInReg(63  downto 32);  
w(2) <= keyInReg(95  downto 64);  
w(3) <= keyInReg(127 downto 96);
```

### Cycle 1

---

```
vW(4) := (subWord(rotWord(w(3))) xor Rcon(1)) xor w(0);  
w(4)  <= vW(4);  
w(5)  <= vW(4) xor w(1);  
w(6)  <= (vW(4) xor w(1)) xor w(2);  
w(7)  <= ((vW(4) xor w(1)) xor w(2)) xor w(3);
```

### Cycle 2

---

```
vW(4+4) := (subWord(rotWord(w(3+4))) xor Rcon(2)) xor w(0+4);  
w(4+4)  <= vW(4+4);  
w(5+4)  <= vW(4+4) xor w(1+4);  
w(6+4)  <= (vW(4+4) xor w(1+4)) xor w(2+4);  
w(7+4)  <= ((vW(4+4) xor w(1+4)) xor w(2+4)) xor w(3+4);
```

### Cycle 9

---

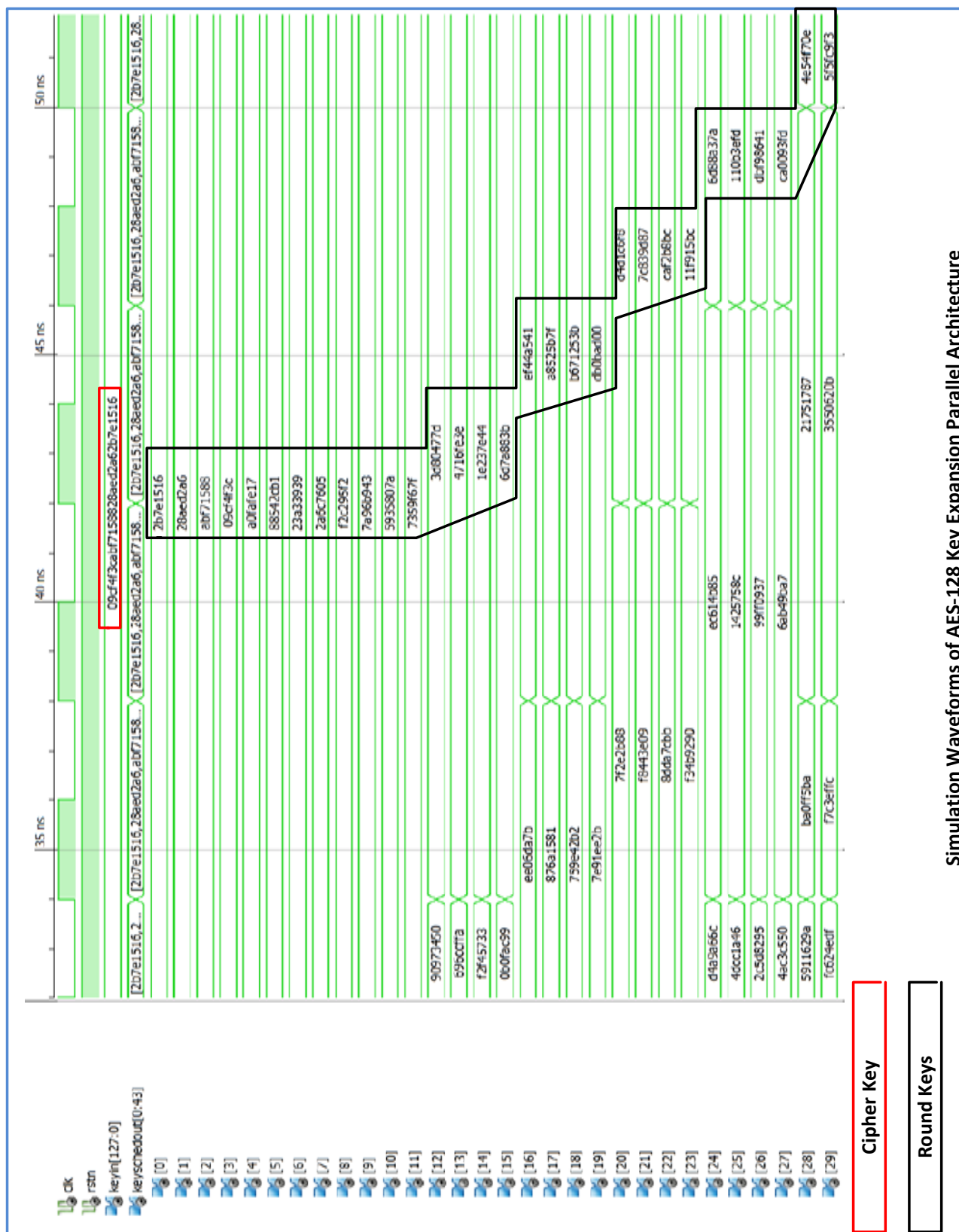
```
vW(4+32) := (subWord(rotWord(w(3+32))) xor Rcon(9)) xor w(0+32);  
w(4+32)  <= vW(4+32);  
w(5+32)  <= vW(4+32) xor w(1+32);  
w(6+32)  <= (vW(4+32) xor w(1+32)) xor w(2+32);  
w(7+32)  <= ((vW(4+32) xor w(1+32)) xor w(2+32)) xor w(3+32);
```

### Cycle 10

---

```
vW(4+36) := (subWord(rotWord(w(3+36))) xor Rcon(10)) xor w(0+36);  
w(4+36)  <= vW(4+36);  
w(5+36)  <= vW(4+36) xor w(1+36);  
w(6+36)  <= (vW(4+36) xor w(1+36)) xor w(2+36);  
w(7+36)  <= ((vW(4+36) xor w(1+36)) xor w(2+36)) xor w(3+36);
```





## B.3 ASC-1 Encryption

Encryption in ASC-1 consists of 4 rounds of AES and operates in a variant of Cipher Feedback Mode (CFB). The encryption of next data block depends on the previous block in feedback mode, which the all block must be processed sequentially. Following shows the sample code for ASC-1.

```
case algState is
  when IDLE =>
    ctValidDly <= '0';
    if cnt < RND1KEYDELAY then
      cnt <= cnt + 1;
    else
      cnt <= 0;
      sInNxt1 <= sIn;
      algState <= rnd1;
    end if;
```

The case is IDLE when the top module is waiting for the Key Schedule to be completed. Once all the round keys are calculated using the key scheduling the state moves to next case, i.e., rnd1

### Round 1

```
when rnd1 =>
  W0nxt <= W(4);
  W1nxt <= W(5);
  W2nxt <= W(6);
  W3nxt <= W(7);
  W_key <= W(40);

  if cnt < RNDDELAY then
    cnt <= cnt + 1;
  else
    cnt <= 0;
```

From round 1 to round 4, round keys and output of previous state is fed into the intermediate states. Leak of bytes depends on odd or even rounds. AES round output certain four bytes from the intermediate variable.

```
Leak <= sOutNxt1(0)(0) & sOutNxt1(2)(0) & sOutNxt1(0)(2) & sOutNxt1(2)(2);
sInNxt <= sOutNxt1;
algState <= rnd2;
end if;
```

```
when rnd2 =>
  W0nxt <= W(8);
  W1nxt <= W(9);
  W2nxt <= W(10);
  W3nxt <= W(11);
  W_key <= W(41);

  if cnt < RNDDELAY then
    cnt <= cnt + 1;
  else
    cnt <= 0;
    Leak1 <= sOutNxt(0)(1) & sOutNxt(2)(1) & sOutNxt(0)(3) & sOutNxt(2)(3);
    sInNxt1 <= sOutNxt;
    algState <= rnd3;
  end if;
```

```
when rnd3 =>
  W0nxt <= W(12);
  W1nxt <= W(13);
  W2nxt <= W(14);
  W3nxt <= W(15);
  W_key <= W(42);

  if cnt < RNDDELAY then
    cnt <= cnt + 1;
```

```

else
    cnt      <= 0;
    Leak2    <= sOutNxt1(0)(0) & sOutNxt1(2)(0) & sOutNxt1(0)(2) & sOutNxt1(2)(2);
    sInNxt   <= sOutNxt1;
    algState <= rnd4;
end if;

```

---

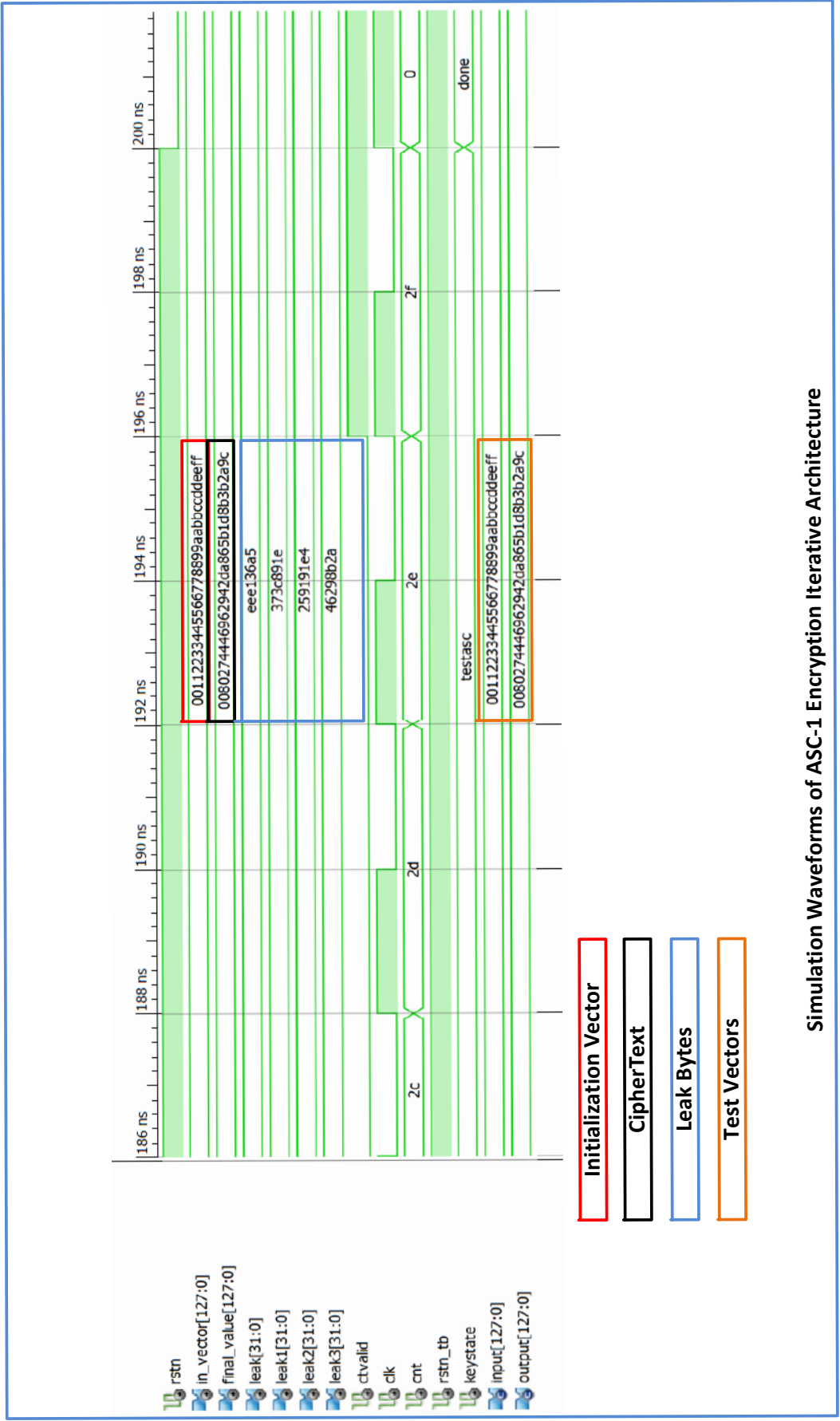
```

when rnd4 =>
    W0nxt <= W(16);
    W1nxt <= W(17);
    W2nxt <= W(18);
    W3nxt <= W(19);
    W_key <= W(43);

    if cnt < RNDDELAY then
        cnt <= cnt + 1;
    else
        cnt      <= 0;
        ctValidDly <= '1';
        Leak3    <= sOutNxt(0)(1) & sOutNxt(2)(1) & sOutNxt(0)(3) & sOutNxt(2)(3);
        final    <= sOutNxt;
    end if;
when others => null;
end case;

```

Finally, the output of the 4<sup>th</sup> round is XOR-ed with Authentication key to calculate the Tag ( $\tau$ ).





## Co-author statement in connection with submission of PhD thesis

With reference to Ministerial Order no. 18 of 14 January 2008 regarding the PhD Degree § 12, article 4, statements from each author about the PhD student's part in the shared work must be included in case the thesis is based on already published or submitted papers.

---

Paper title: **Implementation of Diffie-Hellman Key Exchange on Wireless Sensor Using Elliptic Curve Cryptography**

Place of publication:

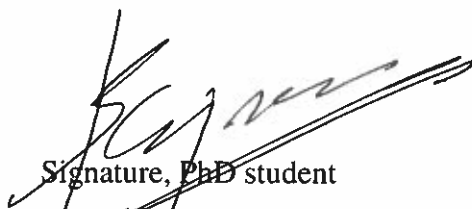
**Wireless VITAE 2011** - Samant Khajuria and Henrik Tange, *Implementation of Diffie-Hellman Key Exchange on Wireless Sensor Using Elliptic Curve Cryptography* : Wireless Communication Society, Vehicular Technology, Information Theory and Aerospace and Electronic Systems Technology, Wireless VITAE 2011

List of authors: **Samant Khajuria and Henrik Tange**

PhD student: **Samant Khajuria**

Scientific contribution (short description (4-5 lines) of all authors' contribution to the article, text):

Both the authors have equal contribution to the paper. The paper is divided into two parts – first part covers the mathematical aspects of the Elliptic Curve Cryptography and the second part is based on the design and implementation of Diffie-Hellman Key Exchange for low-cost low-power wireless sensors.



Signature, PhD student



Signature, co-author

## **Co-author statement in connection with submission of PhD thesis**

With reference to Ministerial Order no. 18 of 14 January 2008 regarding the PhD Degree § 12, article 4, statements from each author about the PhD student's part in the shared work must be included in case the thesis is based on already published or submitted papers.

---

Paper title:

**Authenticated Encryption for Low-Power Reconfigurable Wireless Devices**

Place of publication:

- Journal of Cyber Security and Mobility, 2012.

List of authors:

**Samant Khajuria and Birger Andersen**

PhD student:

**Samant Khajuria**

Scientific contribution (short description (4-5 lines) of all authors' contribution to the article, text):

Samant contributed with design details of ASC-1 authenticated encryption cipher and implementation details and results analysis.

Birger contributed as supervisor and helped getting the paper into shape



Signature, PhD student

Signature, co-author



## Co-author statement in connection with submission of PhD thesis

With reference to Ministerial Order no. 18 of 14 January 2008 regarding the PhD Degree § 12, article 4, statements from each author about the PhD student's part in the shared work must be included in case the thesis is based on already published or submitted papers.

---

Paper title: **ASC-1 : An Authenticated Encryption Stream Cipher**

Place of publication:

**Selected Areas of Cryptography (SAC) – 2011.**

- Goce Jakimoski and Samant Khajuria, *ASC-1 : An Authenticated Encryption Stream Cipher*: Selected Areas of Cryptography (SAC), LNCS 7118, Springer, pp. 356-372, 2012.

List of authors: **Goce Jakimoski and Samant Khajuria**

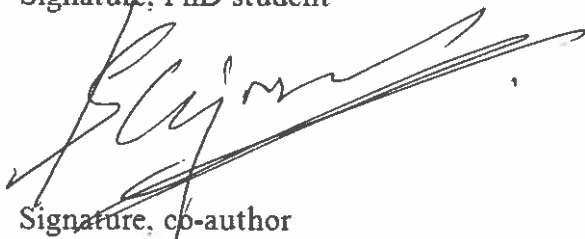
PhD student: **Samant Khajuria**

Scientific contribution (short description (4-5 lines) of all authors' contribution to the article, text):

The above mentioned paper is an outcome of Ph.D student's (Samant Khajuria) six months visit to Stevens Institute of Technology, Hoboken, NJ, USA under the guidance Goce Jakimoski.

Samant contributed with the ASC-1 specification and proof in the paper. Goce contributed as a guide and helped with some mathematical aspects for proving the security of the design.

Signature, PhD student



Signature, co-author

GOCE JAKIMOSKI 