Selective Encryption of Video Data for IoT Environments

A Thesis Submitted to the
College of Graduate and Postdoctoral Studies
In Partial Fulfillment of the Requirements
For the Degree of Master of Science
In the Department of Electrical and Computer Engineering
University of Saskatchewan
Saskatoon

By

Amir Fotovvat

Permission to Use

In presenting this thesis in partial fulfillment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis/dissertation in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis/dissertation work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication, or use of this thesis/dissertation or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis/dissertation.

Requests for permission to copy or to make other uses of materials in this thesis/dissertation in whole or part should be addressed to:

Head of the Department of Electrical and Computer Engineering
57 Campus Drive
University of Saskatchewan
Saskatoon, Saskatchewan S7N 5A9 Canada

OR

Dean
College of Graduate and Postdoctoral Studies
University of Saskatchewan
116 Thorvaldson Building, 110 Science Place
Saskatoon, Saskatchewan S7N 5C9 Canada

Abstract

New advances in the Internet of Things (IoT) have raised lots of concerns regarding the reliability and security of devices within such environments since any vulnerability in the functionality of actuators or leak of data in IoT networks can have severe consequences. One reason for such cybersecurity threats in IoT devices is neglecting security issues during manufacturing due to required costs or expertise. However, IoT systems' progress and growth would not be stable without the development of security standards and rectification of its relevant privacy issues. Moreover, we should mention that the security issues in IoT environments are much more challenging compared to the security issues in the Internet. This is because of the inherent characteristics that come with the IoT, i.e., hardware resource constraints or conducting various physical actions over the Internet network through a wide range of devices.

Visual sensors serve as a critical component of the Internet of Things (IoT). There is an ever-increasing demand for broad applications and higher resolutions of videos and cameras in smart homes and smart cities, such as in security cameras. To utilize this large volume of video data generated from networks of visual sensors for various machine vision applications, it needs to be compressed and securely transmitted over the Internet. H.266/VVC, as the new compression standard that brings the highest compression rate for visual data. To provide security along with high compression, a selective encryption method for hiding information of videos is presented for this new compression standard. Selective encryption methods can lower the computation overhead of the encryption while keeping the video bitstream format which is useful when the video goes into untrusted blocks such as transcoding or watermarking. Syntax elements that represent considerable information are selected for the encryption, i.e., luma Intra Prediction Modes (IPMs), Motion Vector Difference (MVD), and residual signs., then the results of the proposed method are investigated in terms of visual security and bit rate change. Our experiments show that the encrypted videos provide higher visual security compared to other similar works in previous standards, and integration of the presented encryption scheme into the VVC encoder has little impact on the bit rate efficiency (results in 2% to 3% bit rate increase). The results indicate that

this method can serve as a suitable video encryption scheme in the latest video compression standard for most applications. Moreover, the advantages of such methods are further highlighted when considering the UHD videos, such as for 8k or 16k resolutions in Augmented Reality (AR) and Virtual Reality (VR) applications.

Acknowledgments

I would like to express my gratitude and appreciation to my supervisor, Professor Khan Wahid, for his guidance, support, and valuable lessons during my studies. Without his insightful advice and criticism, this thesis could not be completed. I also would like to extend my gratitude to Professors Ralph Deters, Liang Xiaodong, and Seok-bum Ko for serving as my defense committee members and for their insightful feedbacks.

I like to thank my labmates, especially Shahim and Ehsan, for the opportunity I had of working with them and sharing their knowledge and experiences.

I would like to express my sincere thanks to my parents for their efforts. I also would like to thank my siblings for their good wishes, and especially Meysam for his continuous guidance and support.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AEAD	Authenticated Encryption with Associated Data
AES	Advanced Standard Encryption
AI	Artificial Intelligence
AR	Augmented Reality
AMVR	Adaptive Motion Vector Resolution
AV1	AOMedia Video 1
AVC	Advanced Video Coding
CCM	Counter with Cipher Block Chaining
CPU	Central Processing Unit
CTB	Coding Tree Block
CTU	Coding Tree Unit
CU	Coding Unit
DOS	Denial of Service
DRM	Digital Rights Management
GCM	Galois/Counter Mode
GOP	Group of Pictures
HD	High Definition
HEVC	High Efficiency Video Coding
HM	HEVC Test Model
IoT	Internet of Things
IoVT	Internet of Video Things
IP	Internet Protocol
IPM	Intra Prediction Mode
LWC	Lightweight Cryptography
MMVD	Merge Motion Vector Differences
MVD	Motion Vector Difference
NIST	National Institute of Standards and Technology

OCB	Offset Codebook Mode
POC	Picture Order Count
RSA	Rivest-Shamir-Aldemen
RAM	Random Access Memory
RFID	Radio Frequency Identification
RPI-3B	Raspberry Pi 3B
RPI-ZW	Raspberry Pi Zero W
SE	Selective Encryption
SSH	Secure Shell
UART	Universal Asynchronous Receiver-Transmitter
UHT	Ultra High Definition
VR	Virtual Reality
VVC	Versatile Video Coding
VTM	VVC Test Model
WSN	Wireless Sensor Network

Chapter 1

Introduction

1.1 Internet of Things

Internet of Things (IoT) is a network of many objects such as sensors, actuators, wearable devices, cameras, and home appliances that are communicating with each other to share information or perform physical actions. It was in 1999 that the term IoT was coined by Kevin Ashton, who was at the time working on Radio Frequency Identification (RFID). IoT can also be considered an interdisciplinary subject since it is a very broad area and connects to several other fields such as computer networks, wireless technologies, Artificial Intelligence (AI), embedded devices, data analytics, etc.

In the coming years, such an interconnected network of objects will become very helpful as a key technology in the world. The purpose of IoT would be to connect a variety of end nodes and devices to the Internet in order to exchange data and to help many tasks become automated. IoT can make a wide range of tasks become automated and more intelligent. It brings the founding structure for the development of a new and more efficient generation of transportation systems, autonomous vehicles, agriculture automation, video surveillance, and smart cities. Currently, we are at the beginning of the IoT era and new advances in this field will bring many opportunities along with many challenges. There are abundant problems and challenges ahead of this technology that needs to be properly addressed prior to its ubiquitous utilization. For example, sometimes the IoT nodes need to act very fast such as in autonomous vehicles when the processing should be performed at the node and can not be transmitted to the cloud. Thus, edge computing was introduced to solve such issues. Another challenging problem in IoT is the lack of standards. Moreover, the maintenance, reliability, and firmware updates of the objects in the IoT environments are among other factors that require further attention.

The *things* used in IoT systems can contain a wide range of devices and can cover a variety of applications, among which we can mention visual sensors. Internet of Video Things (IoVT) is

an essential part of IoT and is defined as the internetworking of visual sensors [1]. Cameras generate versatile and richer data; with the advances in machine vision and artificial intelligence, cameras are becoming a favorite component in IoT systems. For example, automated retail stores such as Amazon Go are employing cameras and sensors instead of cashiers and workers for giving services to the shoppers. Other broad areas where IoVT can be applied to are home security, video surveillance, and smart city. On the other hand, the video data is huge, and its transmission, storage, processing, and security need further attention.

1.2 Privacy and Security in IoT

With the increasing number of IoT devices in the market, it is important to make sure about the privacy and reliability of the functioning of devices and data transmission. The importance of security in IoT devices becomes more obvious when we look at the variety of similar devices that people are using in their homes for day-to-day tasks. For example, lack of data privacy in smart lights can alone expose people's homes to a higher risk of robbery since, by analysis of such data, an adversary can find out what hours the targeted house is empty. Maintaining security in IoT would be more complicated compared to the Internet since IoT is connecting many physical actions with the Internet using a wide range of devices; thus, considering all the necessary aspects

Table 1-1 Overview of various vulnerabilities in IoT.

Layer	**possible vulnerabilities for RFID**	**possible vulnerabilities for WNS**
Application Layer	Buffer overflow, Injection	Injection, buffer overflows, unauthorized tag reading, tag modification
Network/Transport	Sinkhole, unfairness, false routing, hello and session flooding, eavesdropping.	Tag attacks: Cloning, spoofing Reader attacks: Impersonation, eavesdropping Network protocol attacks
Physical/Link	Jammers, replay attacks, Sybil, selective forwarding, synchronization attack.	Passive interference, active, jamming of temporarily disabling, the device, Sybil, destruction of RFID readers, replay attacks
Multi layer attack	Side channel attack, replay attacks, traffic analysis, crypto attack	Side channel attack, replay attacks, traffic analysis, crypto attack

might not be very easy. Also, many IoT devices are considered resource-constrained devices and might not necessarily have the required processing power, battery, or RAM to perform the cryptography algorithms because of their resource limitations. Some of the important vulnerabilities in IoT environments as was mentioned in [2], [3] are 1) insufficient authentication or authorization, such as lack of high- and low-level access controls, 2) insecure network services, such as exposure against Denial of Service (DOS) attacks 3) lack of transport encryption/integrity verification (usually no encryption is used in transmission of information), 4) insecure software and firmware, such as lack of management systems for updating the software of devices, and 5) poor physical security, such as removable memories and access to the firmware via ports. As the IoT market is relatively new, such issues have not been addressed so far. Moreover, because there are many companies and manufacturers involved in designing and producing IoT devices, the high priority concern so far was to release cheaper and more competitive products. Thus, many security issues are still not considered carefully. Also, a summary of attacks over different layers in IoT gathered from [4] is shown in Table 1-1.

Compared to other IoT sensors and devices, ubiquitous application of visual sensors will lead to more challenges since the storage, computation, transmission, and privacy of large volumes of videos would require significant considerations[1]. Considering the growing demand for IoVT

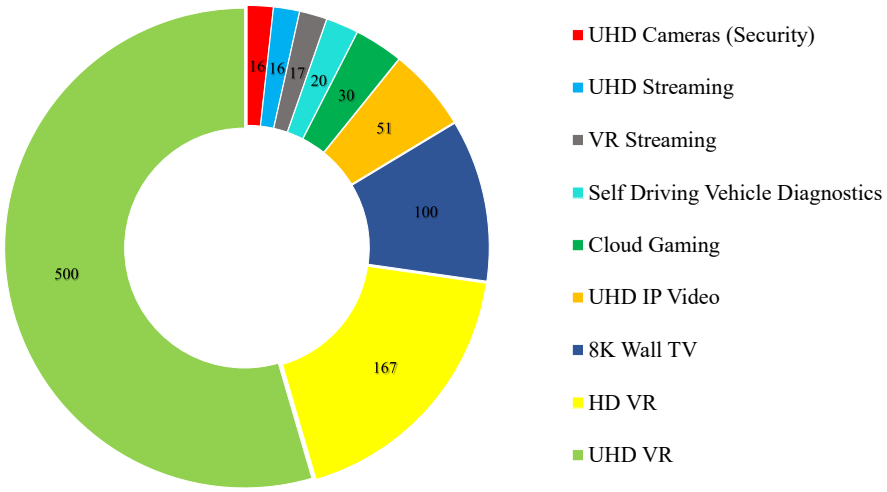


Figure 1-1 Bandwidth demand for future connected homes (Mbps).

and video streaming, the amount of video data on the Internet is rapidly increasing. In [5], It is predicted that by 2030, 13 billion cameras will be around the world from which a huge amount of data will be generated. Cisco predicts that in 2022, 82% of IP traffic will be consumed by video content [6]. Another report indicates a significant demand for video content in future smart homes [7]; Figure 1-1 shows the estimated bandwidth requirement from the results of this report (applications sorted based on their predicted appearance time in future, the upper it is the earlier appearance time would be). With the upcoming video technologies and the demand for higher bandwidth, it is necessary to provide efficient solutions for privacy, storage, and transmission of video data. The encryption of video data is important since the volume of video data can sometimes become large enough that makes the regular encryption process relatively slow. Also, since the video data is compressed using various codecs, we should also take into account the compressed format of video bit streams. H.266/VVC as the new compression standard is employed as the compression algorithm upon which the encryption block will be added to.

1.3 Thesis Objectives

With the upcoming video technologies and the demand for higher bandwidth, it is necessary to provide efficient solutions for privacy, storage, and transmission of video data. There are two general methods to encrypt a video prior to transmission, naive encryption and selective encryption. Naive encryption is defined where the whole video bitstream is encrypted. Using a secure encryption algorithm, this type of video encryption would completely hide any information within the video. On the contrary, selective encryption is when only some of video elements are being encrypted. Figure 1-2 shows an overview of a simplified IoT environment with visual sensors. The connections between cameras and edge server can be either wired or wireless. If the cameras are equipped with powerful processors, they can perform the encoding and encryption. Otherwise, the data can be sent to the edge server with a secure wire link and then the video can be transcoded and encrypted. Depending on the application and different scenarios, some of the computations, such as encryption and compression, can be processed at the edge. Then, the video data can be transmitted to the cloud for other processing, such as machine vision applications and storage. So, while a video is being compressed at the edge, for end-to-end encryption of video data some of the most important elements of the encoder can be encrypted to hide valuable information,

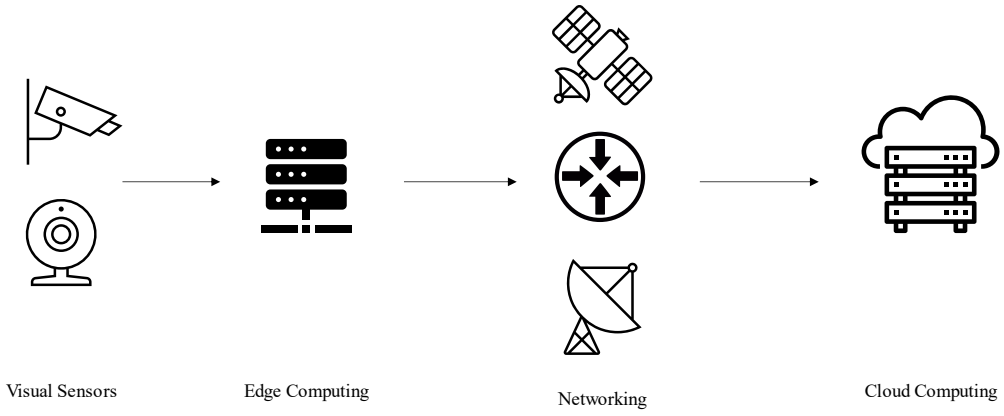


Figure 1-2 IoT environment with visual sensors.

then the encrypted video can be transmitted to the cloud servers. Naive encryption is defined where the whole video bitstream is encrypted. Using a secure encryption algorithm, this type of video encryption would completely hide any information within the video. On the contrary, selective encryption is when only some video elements are being encrypted. Figure 1-2 shows an overview of a simplified IoT environment with visual sensors. Depending on the application and different scenarios, some of the computations, such as encryption and compression, can be processed at the edge. Then, the video data can be transmitted to the cloud for other processing, such as machine vision applications and storage. So, while a video is being compressed at the edge, some of the most important elements of the encoder can be encrypted to hide the information of the video, then the encrypted video can be transmitted to the cloud servers.

Naive encryption is preferred in applications where complete confidentiality of videos is the top priority since it hides all the video information. Selective encryption can lower computation complexity and also is a suitable choice for situations where the video stream is going into other untrusted blocks (for applications such as transcoding, watermarking, and cutting) while being transmitted through the network [8]. This is due to the fact that, unlike naive encryption, videos encrypted by selective encryption approaches are decodable and can be treated like a normal video stream [9]. These features would make Selective Encryption (SE) a suitable choice for Digital Right Management (DRM) services as well [10]. Such selective encryption algorithms can provide a sufficient protection level with relatively small computation cost and delay. The level of output visual deterioration in SE algorithm is completely dependent on the number and type of syntax

elements that are selected for the encryption. Thus, to hide video information using these methods, the most important elements of the encoder that carry significant information of the video such as Motion Vector Differences (MVD), MVD signs, and Intra Prediction Modes (IPM) are usually encrypted. The encryption of syntax elements in the encoder should not cause the decoder to behave strangely, or in other words, the encryption should not change the format compliancy of the video bitstream so that it can be decoded by the receiver with no problem. Thus, for the selection of syntax elements in the selective encryption methods, format compliance of encoder and decoder is of high importance. This paper proposes a selective encryption algorithm for the H.266/VVC which is the latest video compression standard. This next generation of compression standard provides up to 50% more compression rate than the H.265/HEVC [11], thus it can play an important role to reduce the bandwidth consumption or storage requirement of video contents. To provide security along with high compression, a selective encryption method for hiding information of videos is presented for the H.266/VVC standard. Syntax elements that represent considerable information are selected for the encryption, i.e., luma Intra Prediction Modes (IPMs), Motion Vector Difference (MVD), and residual signs., then the results of the proposed method are investigated in terms of visual security and bit rate change. We also investigated the performance of several Authenticated Encrypted with Associated Data (AEAD) lightweight cryptography algorithms and compare them to currently common algorithms. These lightweight cryptography algorithms can lower the computation complexity and for example they can be used in resource constrained sensors.

A summary of the objectives of this thesis are as follows:

1- To provide an overview of security issues in IoT environments and to discuss some of the solutions.
2- To analyse some of promising lightweight symmetric cryptography algorithms that can be used in resource-constrained devices such as sensors.
3- Discussing the applications and challenges of the Internet of Video Things (IoVT) and why they need more considerations.
4- Discussing some of the new technologies in the H.266/VVC and comparing its performance with other popular video compression algorithms.

5- Proposing a selective encryption algorithm for the H.266/VVC standard and comparing the results with other similar works in previous standards.

1.4 Thesis Organization

The structure of the thesis is as follows:

Chapter 2 discusses the future of video applications in smart homes and smart cities. Also, the importance of visual sensors and corresponding challenges regarding their implementation are discussed.

Chapter 3 contains a comparative performance measurement of 35 lightweight Authenticated Encryption with Associated Data (AEAD) cryptography algorithms for the IoT sensors nodes.

Chapter 4 is about the video compression algorithms. It provides a brief overview of the techniques and new tools in the H.266/VVC and its performance compared to other video codecs such as AV1, VP9, and HEVC.

In Chapter 5, a selective encryption method for H.266/VVC is presented. The methodology is presented, and experimental results are examined based on the visual security, bit rate change, and encryption space.

Finally, Chapter 6 contains the conclusions and suggestions for future works.

Chapter 2

Visual Sensors in IoT

2.1 Introduction

Internet of things connects many devices and objects, among which we can mention visual sensors (cameras). Cameras provide us with richer information that is crucial in many applications. This internetworking of cameras are mostly referred to as IoVT. In this field, there are other challenges such as transmission, analyzing, and storage of video data that are becoming more and more important. New advances in the internet networks and IoT has made the ubiquitous application of networks of cameras more possible. This network of visual sensors can be utilized in areas such as security surveillance, smart city applications, smart vehicles, and transportation. There are also many other applications for various industrial applications. Continuous generation of video data and their transmission needs significant networking capabilities. In [5], It is predicted that by 2030, 13 billion cameras will be around the world from which a huge amount of data will be generated. With the popularity and progress of machine learning and deep learning methods in computer vision problems, analyzing videos will become much easier, and it can help in the automation of many tasks such as in autonomous driving vehicles. For high accuracy, many of these applications will demand higher definitions of videos in the future. So, it is necessary to prepare for transmission, storage, and processing of abundant video data. Most of the video data are transmitted to the cloud for processing. Many homes nowadays use multiple security cameras, and people have real-time access to the cameras from the internet. Cisco predicts that in 2022, 82% of IP traffic will be consumed by video content [6]. It is also predicted that in 2030 the video data traffic over the internet will be around 1 ZB per year [5]. Chen indicates the followings as the challenges in the IoVT systems [1]:

1) Energy efficiency, processing, storage, and networking issues in the embedded devices.

2) Reliable high bandwidth networks with capabilities to handle the timely transmission of large volumes of visual data.

3) Integrating compression algorithms with data analytics or even encryption for more efficient computations.

4) Maintaining privacy and security of visual data is crucial in both the cloud and edge.

5) Standardized systems and networks for IoT environments to handle the coordination of data among IoVT data owners.

2.2 Future IoVT Applicatoins

2.2.1 Autonomous Vehicles and Traffic Monitoring

Most of the autonomous vehicles use multiple cameras to gather information about the surrounding environment for the computers to make driving decisions. There are, in general, two types of visual cameras that can be employed for these purposes. First, we have the cameras inside and around the vehicle itself that are used to make real-time decisions about cars' direction, speed, and other objects. The second type of camera is the ones that are installed on roads and at intersections. These visual sensors used in roads and intersections can provide further traffic intelligence for autonomous vehicles and for real-time maps to have online information about accidents, weather, and other information about traffic and cars. One of the challenges in this area is how to connect side road cameras with autonomous vehicles so that the combined data can give more intelligence regarding the environment, which would help the AI in autonomous vehicles to make better decisions. Even the autonomous cars themselves can use their own cameras to help another vehicle’s perception, which is called perception [12].

2.2.2 Medical and Healthcare Applications

One of the major applications of IoT has always been for healthcare-relevant devices, which was discussed at the beginning of IoT emergence. Devices that continuously gather data from users and patients can help to provide better diagnoses for the consumers. The data can be accessible for a doctor on a real-time basis for medical advice, and diagnosis or even simply the device can itself report health metrics and advice to the user based on their algorithms. For example, there are

currently many non-invasive glucose measurement devices in the market that continuously reports back to the patients of their glucose level and can alarm them when the glucose level is very high. Visual sensors also can be used for the health tracking and monitoring of people's activities. For example, with recent innovative ideas in IoT devices, there are mirrors that teach the users how to exercise, and with built-in cameras, the device can get feedbacks about health metrics and if the person is doing the exercise right. It also can remotely connect people together for group activities. Another application can be the use of a network of cameras in hospitals and senior citizen houses so that it can analyze the videos, and in case of an emergency, it can contact other people who can help. The challenging point about these applications that require continuous monitoring is the security and privacy of videos. Because these types of data are sensitive information, relevant considerations and procedures must be taken into account to prevent vulnerabilities.

2.2.3 Industrial Applications and Automatization

IoT can be a very promising solution for many industrial applications, from mining to agriculture to different factories. Since years ago, cameras have been employed for product line inspection in manufacturing processes. With the integration of IoT in industrial applications, the accuracy, efficiency, and safety in the factories are expected to be improved [13]. Moreover, IoVT with the help of machine vision, can help to make various product lines become completely automated. Such processes require secure, reliable, and real-time communications to effectively aid the automation. Other applications in industrial environments also include detection of suspicious activities and security purposes [14]. In agriculture, robots and drones can be used in the early future to automate the harvesting, diagnosis of plants' diseases, and spraying. Moreover, IoVT can provide affordable automation solutions for retail stores as well. There is no need for cashiers in future stores where cameras have already checked out what shoppers have picked. For example, Amazon Go stores are already using such technologies.

2.2.4 Video Surveillance

Many people are already using security cameras in their homes. Governments and different places also implement their own network of security cameras. Most of the data generated from these cameras are locally processed and analyzed. However, with IoVT, a network of security cameras can bring many interesting applications into the real world. Using machine vision, the data can be

used for the detection of smoke, fire, robbery, etc. In advanced situations, some countries are implementing security cameras for face detection. Such applications are highly useful for a variety of applications; however, they also impose lots of challenges as well. Privacy and security of generated data, accurate analysis of videos with AI, for what purposes the data can be used for, and who should have access to such data are among the issues and challenges of wide video surveillance applications.

2.3 Technical Challenges in IoVT

Artificial Intelligence (AI) will be a part of almost any future technology. AI requires an abundant amount of data for learning, and future IoT/IoVT environments can provide this data for the next generation of AI models. This large volume of rich visual data provides the opportunity for various analytics [15]. Transmission of usual IoT data to the internet would not impose significant challenges. However, video data generated from IoVT networks can become extremely large and its processing and transmission would require lots of processing power and networks capable of moving this large amount of data. With the rapid growth in the number of security cameras, especially for video surveillance in cities and industrial applications, there is a need to automatically detect the activities and behaviors in the videos. Because spending time watching recorded or real-time videos in order to know about events can be extremely inefficient. Moreover, with smart implementations, parts of data can be processed at the edge, so that to only send the necessary information is sent over the internet in order to prevent taking unnecessary internet bandwidth. This will bring other questions and challenges regarding the kinds of computations that need to be performed at the edge, the structure of the network, and new standards. Thus, proper implementation of edge processing in the future IoVT systems can have a significant effect on the bandwidth consumption, energy efficiency, and security of videos (as the processing gets closer to the end-user, the fewer security concerns will be imposed). Moreover, there are applications that are affected by the delay and can not be only addressed by cloud computing. All these issues are promoting the distributed edge computing instead of centralized cloud computing approaches [16].

In IoVT systems, each application might require different video pixel resolution and different framerates. Since the IoVT videos are not mostly for entertainment and the videos are generated for machine vision applications, the end goal is no longer high quality for human eyes. Thus, the quality of the video should be compared based on the features and syntaxes needed for

the specific machine vision application. Hereby, the compression algorithms also can change, and new compression methods need to be proposed and be optimized for specific machine communications [17]. Also, in a network of adjacent cameras, there are lots of abundant information, which would make their transmission and processing inefficient. Instead, it can be better to first remove the redundancies between video data and then do other computations. The main value of IoVT comes when aggregating the information from all visual sensors together. To do so, at least some portion of data from each node needs to be transmitted to the cloud. Such transmission of large volumes of video data might not get to its real potential without high-performance pervasive networking that has its own challenges such as application-specific protocol management, caching, and hardware radio components [1].

Chapter 3

Lightweight Cryptography Algorithms for IoT Sensor Nodes

Some parts of this chapter are published in [3]. I was responsible for conducting the experiments and codes along with writing.

3.1 Introduction

Edge nodes in IoT systems are constantly exchanging data with each other. Now, with the advancement of IoT applications, the security and privacy aspects of data and functionalities are becoming more and more important. There are many IoT devices that do not use proper encryption or authentication [2] which can lead to eavesdropping attacks. In healthcare and medical areas, there are many resource-constrained devices such as peacemakers with small RAM and processing power resources [18]. The risk of data leaks or malfunctioning of IoT devices will have unimaginable consequences for the manufacturing companies and users. Also, it is worth pointing out that rectifying security issues in IoT environments is more challenging compared to the Internet since IoT connects the physical actions (through actuators and other devices) with the Internet through a wide range of devices. Thus, recently there has been lots of works to propose secure communications and standards in IoT nodes. On one hand, since IoT devices are mostly designed for efficiency and their competitive price, security and reliability issues have not been the top priority. Moreover, because of the lower computation power and resource constraints in IoT devices, previously standardized cryptography algorithms such as Advanced Encryption Standard (AES) and RSA might not be a good choice for such devices. Also, not to mention that energy consumption is also another important factor since many IoT devices work with batteries. Thus, in such environments, there is a need for cryptography algorithms that require smaller resources for the operation which are called Lightweight Cryptography (LWC) algorithms. These cryptography algorithms are designed to provide authentication and encryption in computers with higher Random Access Memory (RAM) and CPU resources such as Personal Computers (PCs),

smartphones, and servers. In [19], a more comprehensive discussion of LWC algorithms and their importance in different layers of IoT systems is presented. To compare cryptography implementations, we can either compare characteristics in terms of software or hardware performance. For hardware-oriented purposes, comparison can be conducted based on the required gate area, energy consumption, number of logic blocks, or gate equivalents. For software implementations, the performance can be determined by considering RAM, Read-Only Memory (ROM), and CPU usage [20], [21]. Moreover, latency, power consumption, and throughput are other important factors. Since the rate of data transmission in resource-constrained devices is relatively low, throughput is not of high priority for LWC algorithms [21].

No standardized encryption or authentication algorithm is yet decided for IoT devices. However, since 2015 there have been lots of efforts from NIST to find suitable LWC solutions. In this chapter, we attempt to analyze the performance of 35 Authenticated Encryption with Associated Data (AEAD) cryptography algorithms when implemented at the network layers of IoT edge nodes. Among the 35 chosen algorithms, 32 were selected from the second round of lightweight cryptography standardization process of the National Institute of Standards and Technology (NIST) as shown in Table 3-1. The table contains the category of the cryptography algorithms and also identifies what version of the algorithm was used for the tests. The three other algorithms are AES with CCM, GCM, and OCB modes of operation. These AEAD algorithms take plain text, key, nonce, and associated data as inputs. The associated data is not encrypted, but its authenticity can be verified. In terms of security, in general, AEAD algorithms should guarantee security as long as the nonce is unique for each encryption with the same key [21]. To compare performance of these algorithms, their RAM requirement, CPU usage, and execution time are measured using three different IoT devices. The selected IoT nodes are Raspberry Pi 3, Raspberry Pi Zero W, and iMX233. The results are useful for designing lightweight cryptography algorithms suitable for resource constrained IoT devices.

Table 3-1 Selected cryptography algorithms.

No.	Algorithm name	Category	Ref
1	ACE (aceae128v1)	Permutation	[47]
2	ASCON (ascon128v1)	Permutation	[48]
3	COMET (comet128aesv1)	Block Cipher	[49]
4	DryGASCON (drygascon128)	Permutation	[50]
5	Elephant (elephant160v1)	Permutation	[51]
6	ESTATE (sestatetweaes128v1)	Tweakable Block Cipher	[52]
7	ForkAE (saefforkskinnyb128t256n120v1)	Tweakable Block Cipher	[53]
8	GIFT-COFB (giftcofb128v1)	Block Cipher	[54]
9	Gimli (gimli24v1)	Permutation	[55]
10	Grain-128AEAD (grain128aead)	Stream Cipher	[56]
11	HYENA (hyenav1)	Block Cipher	[57]
12	ISAP (isapa128av20)	Permutation	[58]
13	KNOT (knot128v1)	Permutation	[59]
14	LOTUS and LOCUS (twegift64locusaeadv1)	Tweakable Block Cipher	[60]
15	mixFeed (mixfeed)	Block Cipher	[61]
16	ORANGE (orangezestv1)	Permutation	[62]
17	Oribatida (oribatida192v12)	Permutation	[63]
18	PHOTON-Beetle (photonbeetleaead128rate128v1)	Permutation	[64]
19	Pyjamask (pyjamask128aeadv1)	Block Cipher	[65]
20	Romulus (romulusm1v12)	Tweakable Block Cipher	[66]
21	SAEAES (saeaes128a64t128v1)	Block Cipher	[67]
22	Saturnin (saturninctrcascadev2)	Block Cipher	[68]
23	SKINNY (skinnyaeadtk29664v1)	Tweakable Block Cipher	[69]
24	SPARKLE (schwaemm128128v1)	Block Cipher	[70]
25	SPIX (spix128v1)	Permutation	[71]
26	SpoC (spoc128sliscplight256v1)	Permutation	[72]
27	Spook (spook128mu384v1)	Tweakable Block Cipher	[73]
28	Subterranean (subterraneanv1)	Permutation	[74]
29	SUNDAE-GIFT (sundaegift128v1)	Block Cipher	[75]
30	TinyJambu (tinyjambu128)	Block Cipher	[76]
31	WAGE (wageae128v1)	Permutation	[77]
32	Xoodyak (xoodyakv1)	Permutation	[78]

3.2 Methodology

For the first step, the hardware platforms should be configured to minimize background processing. Raspbian was selected for the Raspberry Pi 3B (RPI-3B) and Raspberry Pi Zero W (RPI-ZW), and Debian for the iMX233 board. More information regarding the selected hardware platforms and our test procedures are shown in Table 3-2. Since all the codes are in C language, we use the gcc with no optimization or sanitization in order to compile the codes. The monitoring software is in python and runs on a separate device to prevent extra processing on the device under test. The monitoring software stores the CPU, RAM usage, and execution time while the tests are executing. For the power measurement, we use a separate device with its own automation software, which runs independently from other devices. An overview of the testing platform is shown in Figure 3-1 where the measurement unit, digital wattmeter, and PC (which controls both the IoT platform and the measurement units)
are specified. The IoT platform at any test can be one of RPI-3B, RPI-ZW, or iMX233.

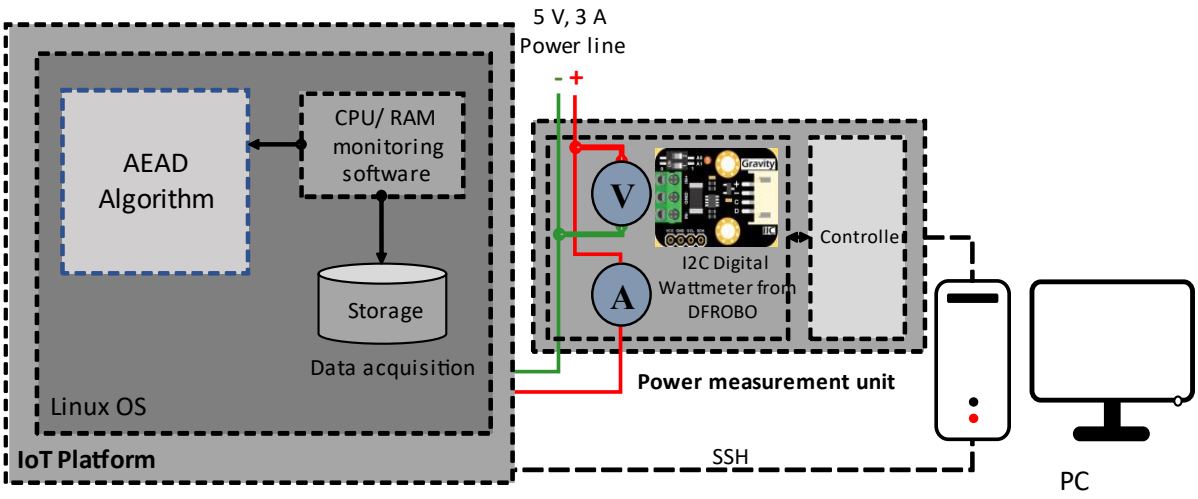


Figure 3-1Illustration of experimental setup.

Universal Asynchronous Receiver-Transmitter (UART) communication was used for the Secure Shell (SSH) connection. The devices are also powered with 5 V and 3 A adaptor which passes through the DFROBO power measurement unit. This wattmeter is capable of voltage resolution of 4 mV and its relative error is less than ±0.2%. The data will be stored and its processing will be

done after the tests. The experiment process is as follows: 1) the PC sends a request to the measurement unit to start sampling the voltage and current; 2) the PC sends another request to the target hardware to run the automation program written in python; 3) the python program executes the algorithm and gets its Process Identification (PID); 4) the automation program monitors the CPU and RAM usage for the known PID; 5) at the end of the algorithm, the automation program sends back the halt message to the PC; 6) the PC requests the measurement unit to stop the sampling and kill all related processes, and 7) finally, all data are stored on the PC for post-processing. Moreover, the sampling time for CPU, RAM usage, and power is set to 10 ms.

Table 3-2 AEAD algorithm performance measurement matrices.

	Used Platform	**Core**	**Similar Architecture**
Hardware Environment	1. RPI-3B (1.2GHz), 1GB SDRAM) [22]	ARM A53, Quad core	Pine A64, Rock64 (RK3328)
	2. RPI-ZW (1GHz), 512MB SDRAM) [23]	ARM 1176, Single core	iMX35, iMX31
	3. iMX233 (454MHz), 64 MB SDRAM) [24]	ARM 926, Single core	Arduino DUE (AT91SAM), LPC3220/40/50, OMAP 1
Parameter measured	1. CPU utilization 2. Maximum RAM requirement 3. Processing time 4. Total energy consumption 5. Transmission time		
Procedure	1. Variable hardware environment 2. Variable data size 3. LoRa wireless transmission		

3.3 Results

The data collection procedure is divided into four phases: 1) reference measurement, 2) test case 1 (different hardware platforms), 3) test case 2 (different message sizes), and 4) test case 3 (IoT sensor node application). In test case 1, all 32 AEAD algorithms are executed using the same benchmark to measure their performance in terms of execution time, CPU load, RAM usage, and energy requirement. For each algorithm, 1082 executions with different combinations of the message and associated data size (from 0 to 32 bytes) are performed. In test case 2, the execution

time and RAM requirement are measured using three different message sizes (100 B, 1KB, and 10 KB) to evaluate the effect of message size and memory management. Test case 3 is performed on an IoT sensor node with a LoRa wireless module. In this experiment, five AEAD algorithms are used to encrypt the sensor data (30 bytes) before transmitting, using the LoRa interface. The main objective of this experiment is to measure the contribution of the AEAD algorithms in terms of time and energy usage while transmitting sensor data over the LoRa interface. At first, power measurement is started by determining the base power consumption of each platform (average power consumption for the OS and automation software without running the AEAD algorithms). Then each test for each embedded device is conducted as explained in the following parts.

3.4 Tests with Different Hardware Platforms

For the final data collection, all the AEAD codes were used as received from the NIST without any optimization. From previously common authenticated encryption algorithms, we selected GCM [25], OCB [26], and CCM [27] modes of operation. Unlike OCB, GCM and CCM are both patent-free algorithms. For these algorithms, we used the OpenSSL library with AES as the block encryption. The RPI-3B, RPI-ZW, and iMX233 were employed as the platforms under test and the same structured test for each algorithm was used, in which encryption and decryption were performed 1082 times for different combinations of message and associated data sizes (from 0 to 32 bytes). Then, the overall execution time, RAM, and energy consumption were reported. After post-processing of the acquired data, it was found that some of the algorithms showed a relatively similar processing time. Figure 3-2 shows the algorithms' total execution time on three different hardware platforms arranged from the fastest to the slowest algorithm with the non-LWC algorithms (GCM, OCB, and CCM) at the beginning. It is worth mentioning that six of the last seven algorithms with the highest execution time are permutation-based ciphers. Among the seven algorithms with the lowest execution time, four of them are block cipher-based, two are permutation-based, and one is tweakable block cipher-based. In Figure 3-3, the maximum RAM usage for the LWC algorithms and three non-LWC algorithms are shown. All the algorithms use the lowest amount of RAM while executed in the RPI-3B and the highest while executed in the RPI-ZW. According to the results, the LWC algorithms use 308 KB to 368 KB of RAM in the RPI-3B. For the IMX board, they use 424 KB to 500 KB RAM, which is slightly higher than the RPI-3B. However, all the LWC algorithms use much more RAM in the RPI-ZW, which is above 968 KB and up to 1180 KB. SPARKLE is

found to be the lowest RAM-consuming LWC algorithm for all three hardware platforms. The Oribatida, ASCON, and Grain-128AEAD are the highest RAM-consuming algorithms in the RPI-3B, IMX, and RPI-ZW platforms, respectively. However, GCM, OCB, and CCM required almost 2.5 times RAM compared to the LWC algorithms for all three platforms. Therefore, these algorithms may not be suitable for IoT sensor nodes.

Figure 3-4 shows the CPU utilization for the LWC and three AES algorithms. Although there are some variations for different algorithms, most of them were in a specific range, depending on the employed hardware platform. The iMX233 shows the lowest CPU utilization ranging from 61% to 64.9%, and the RPI-ZW shows between 70.9% to 78.7%. The RPI-3B shows the maximum among the three, which is from 79.9% to 94.9%. The iMX233 is a single-core processor and has the lowest speed. It was able to dedicate lower CPU resources to execute the algorithms while performing the other tasks of the OS itself along with the data display. At the same time, the RPI-3B having a quad-core CPU, was able to dedicate one of the cores for the AEAD algorithms and had the fastest processing time by utilizing almost 95% of the CPU. Among the LWC algorithms, Saturnin occupied the lowest and Elephant occupied the highest CPU utilization in IMX. Whereas ASCON got the lowest and SpoC got the highest CPU utilization in the RPI-ZW; SAEAES utilized the least and Romulus utilized the most CPU in the RPI-3B platform. All three non-LWC algorithms utilized the CPU less than some of the LWC algorithms, like Subterranean in all three hardware platforms. The same experiment was performed on a PC with a corei7 intel and 8 GB of RAM, and execution time was found to be faster compared to the RPI-3B. However, the relative performance results among the algorithms remained almost the same.

The energy consumption also depended on the execution time as shown in Figure 3-6. Among the three hardware platforms, the RPI-ZW always demonstrated the lowest energy consumption for almost all the LWC and AES algorithms. The IMX showed the highest energy consumption for all the LWC algorithms; however, the three AES algorithms consumed less energy compared to the consumption for the RPI-3B platform. All the LWC algorithms followed a similar sequence in terms of energy consumption for all three hardware platforms. Figure 3-6 shows that almost two-thirds of the LWC algorithms consume less than one Joule in the IMX. One-third of them consume as low as 500 mJ for the same hardware platform. Among the ten high-energy consuming LWC algorithms, Grain-128, Elephant, and Oribatida consume more than 5 J in the IMX platform. Among the LWC algorithms, TinyJambu is the lowest energy-consuming algorithm in the IMX and

RPI-3B, whereas SPARKLE consumes the lowest in the RPI-ZW board. All three non-LWC algorithms consume the loswest energy in the RPI-ZW when comparing all the LWC algorithms. Among the hardware platforms, the energy consumption was highest for all the algorithms when using the iMX233 board, but we chose this platform as the most preferred option for the IoT end-node due to its low power consumption during both the AEAD execution and other tasks. It shows that the iMX233 has the lowest power consumption of 260 mW; on the other hand, the RPI-ZW and RPI-3B consume 420 mW and 2200 mW of power, respectively. It also shows the power consumption during the execution of the AEAD algorithms.

3.5 Performance over Different Message Lenghts

In the next phase, experiments were designed to examine the dependency of execution time and RAM requirement versus message size; 100 bytes, 1 KB, and 10 KB message sizes were used. Here, we used the iMX233 platform since the goal of this experiment is to convey only the effect of the message size on the LWC algorithms. We repeated the encryption and decryption of the messages ten times and measured the execution time and RAM usage for each one. Then, to derive a single time iteration, the execution time was divided by a factor of ten. For better accuracy, we repeated this process several times for each algorithm and then reported the average as the execution time Figure 3-5 shows the algorithms arranged according to their execution time for single encryption and decryption of a 10 KB message (with non-LWC algorithms at the beginning). By comparing the execution times to the ones in Figure 3-2, there were some changes in the order of algorithms compared to test case 1. These variations may be due to the differences in test cases 1 and 2 (differences in message sizes and repetition times). Comparing the performance of all 32 algorithms in terms of message size, we found increasing execution time with the increase of text input length. Table 3-3 shows the RAM requirement for three different message sizes. It can be seen that the RAM requirement for algorithms is almost the same for input messages shorter than 1 KB but increases from 5% to 12% for a message size increase of 10 times. The only exception was found for the Grain-128 algorithm, which showed an increase of 29%.

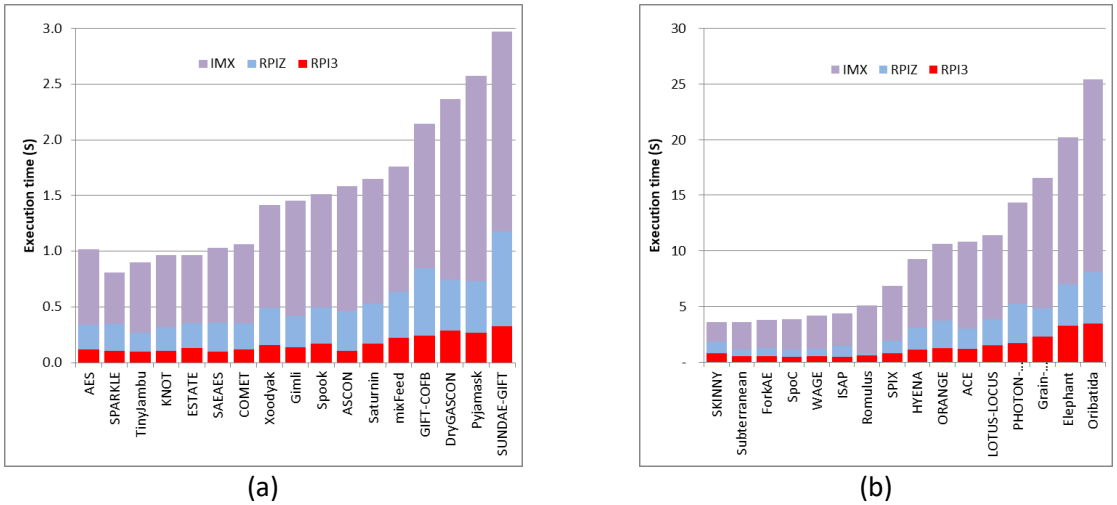


Figure 3-2 AEAD execution time on three different hardware platforms (arranged from low to high execution time, and keeping non-LWC algorithms at the beginning).

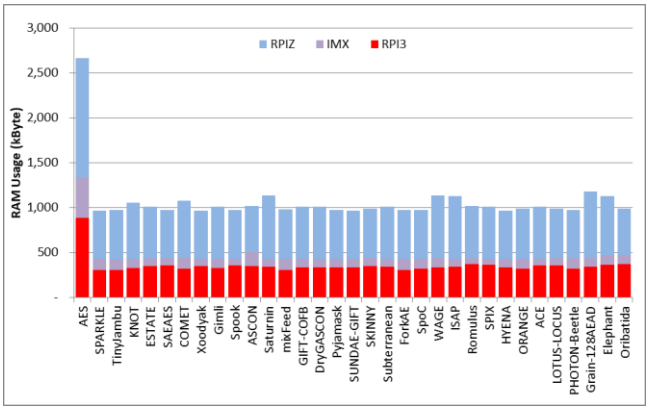


Figure 3-3 AEAD maximum RAM requirement (same sequence as previous figure).

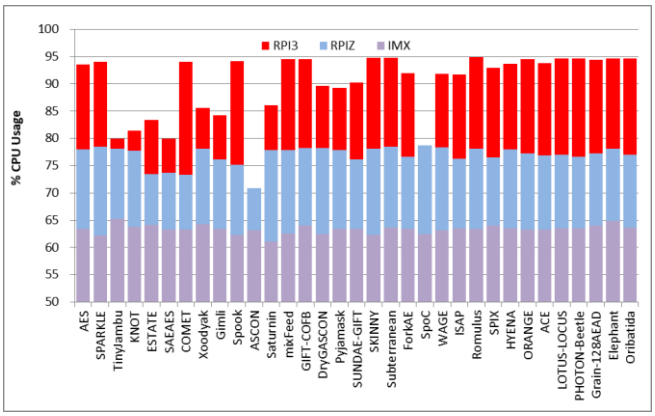


Figure 3-4 AEAD CPU utilization in three different hardware platforms (same sequence as previous figure).

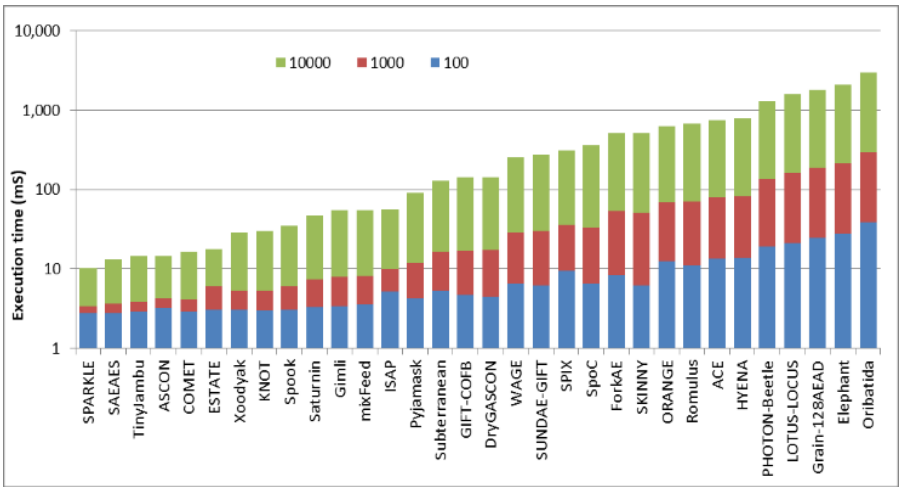


Figure 3-5 AEAD algorithm comparison for three different text lengths on iMX233 hardware platform in terms of processing time, arranged from fastest to slowest algorithm according to the 10 KB message size with non-LWC algorithms at the left (same sequence as previous figure).

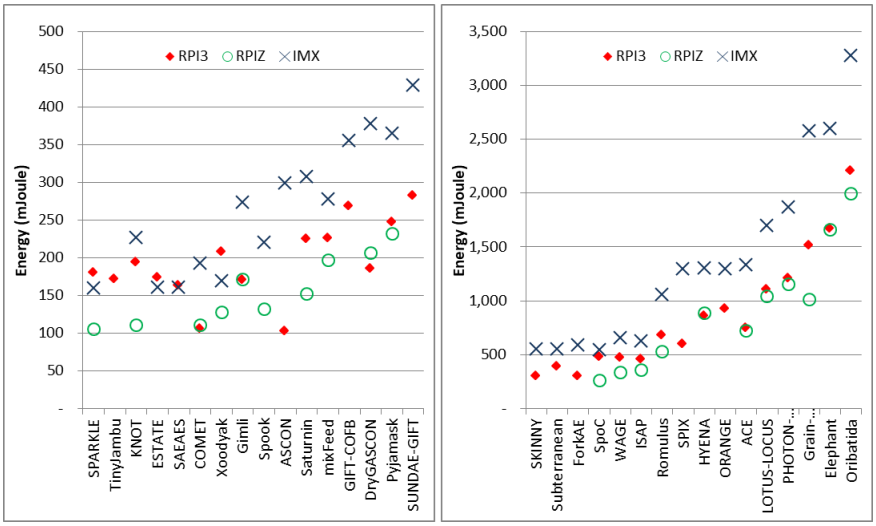


Figure 3-6 Energy consumption of the AEAD algorithms.

Table 3-3 Memory usage versus message size.

Algorithm	RAM usage (KB)		
	Data Message Size		
	0.1 KB	1 KB	10 KB
ACE	336	336	352
ASCON	348	348	372
COMET	344	344	400
DryGASCON	288	288	312
Elephant	324	324	344
ESTATE	288	288	312
ForkAE	292	292	324
GIFT-COFB	288	288	308
Gimli	284	284	312
Grain-128AEAD	340	348	448
HYENA	288	288	312
ISAP	288	288	312
KNOT	288	288	324
LOTUS-LOCUS	288	288	312
mixFeed	292	292	316
ORANGE	288	288	308
Oribatida	372	372	396
PHOTON-Beetle	288	288	308
Pyjamask	288	288	312
Romulus	296	296	312
SAEAES	290	290	320

Algorithm	RAM usage (KB)		
	Data Message Size		
	0.1 KB	1 KB	10 KB
Saturnin	288	288	312
SKINNY	280	280	292
SPARKLE	288	288	312
SPIX	336	336	352
SpoC	336	336	352
Spook	288	288	312
Subterranean	292	292	312
SUNDAE-GIFT	336	336	352
TinyJambu	284	284	308
WAGE	340	340	356
Xoodyak	288	288	312

3.6 Summary

In this chapter a comparative performance analysis of 35 AEAD algorithms was presented. Among the 35 chosen algorithms, 32 were selected from the second round of lightweight cryptography standardization process of the National Institute of Standards and Technology (NIST). The three other algorithms are AES with CCM, GCM, and OCB modes of operation. These AEAD algorithms take plain text, key, nonce, and associated data as inputs. To compare performance of these algorithms, their RAM requirement, CPU usage, and execution time are measured using three different IoT devices. The selected IoT platforms that the tests were performed on are Raspberry Pi 3, Raspberry Pi Zero W, and iMX233. The results are useful for designing and employing lightweight cryptography algorithms suitable for resource constraint IoT devices such as sensor nodes.

Chapter 4

Versatile Video Coding (VVC/H.266)

4.1 Introduction

Video compression techniques allow us to lower the size of videos without losing quality, i.e., same video quality with a lower bitrate. Video codecs can be lossy or lossless. In lossy encoding algorithms, some of the information is completely lost through the process, but in lossless, the output is completely reversible without losing any information. Almost all of the current videos we see or send over different platforms use compression to lower the bandwidth for transmission and storage. Depending on the video compression algorithm, the compression ratio is different. For example, in Advanced Video Coding (AVC) technology, or H.264, the raw video is 20 to 200 times larger compared to the compressed video. H.264/AVC was released in 2003 and still is considered a popular compression standard because of its simplicity. H.265/HEVC is the next standard that was released in 2013, which has 50% more compression efficiency over H.264/AVC (of course with more computation cost). Other famous compression algorithms are AV1 and VP9, which both are considered as royalty-free codecs, unlike H.264/AVC or H.265/HEVC. For example, whenever watching a video on Youtube, if the video is 4k, it is compressed with AV1, and for other resolutions, VP9 is used. AV1 and VP9 have been more suitable options for the industry because of the complexity of licensing and price in H.265/HEVC and H.264/AVC standards. However, with the increasing demand for higher resolution videos and a variety of video applications, 90 percent of internet traffic will be videos content; thus, there is a need for better video compression algorithms.

Raw videos contain a lot of redundant information and simply using the raw video data for transmission and storage is not efficient, especially when we take into account the large volume of video data. Compression algorithms allow us to save bits require for a video without losing quality. This happens by removing the extra bits used for spatial redundancy and temporal redundancy with various techniques and algorithms. Spatial redundancy represents the redundancies within

the same frame, like the background pixels of a frame that all have the same color. For such information, we do not need to repeatedly store the pixels' RGB values; instead, for example, we can store the RGB value and indicate the regions that have that color, which would reduce the number of bits. Temporal redundancy refers to redundant storage bits for the same region or objects within different frames. In video frames, since all the frames are connected, a lot of information is the same among adjacent frames with just a little movement and change. Thus, it is more efficient to only store the vector values that tell what objects or regions are moved from the previous (next) frame to the current frame. Video compression standards use hybrid video encoders and decoders that are a combination of transform coding and predictive coding. A simplified version of a hybrid encoder for video compression can be seen in

Figure 5-2 (please note that the selective encryption block is not a part of hybrid encoders and is only used for video encryption). The compression process starts with dividing video frames into Group of Pictures (GOP), e.g., IBBB. This helps to find relations between upcoming and previous frames in order to find their relations with the current frame. Depending on the video compression technology and requirements, various structures can be used. Figure 4-1 shows an example GOP and its Picture of Count Order (POC) and decoding order. Not to mention that the suitable structure of GOP is important for high output quality and high compression at the same time.

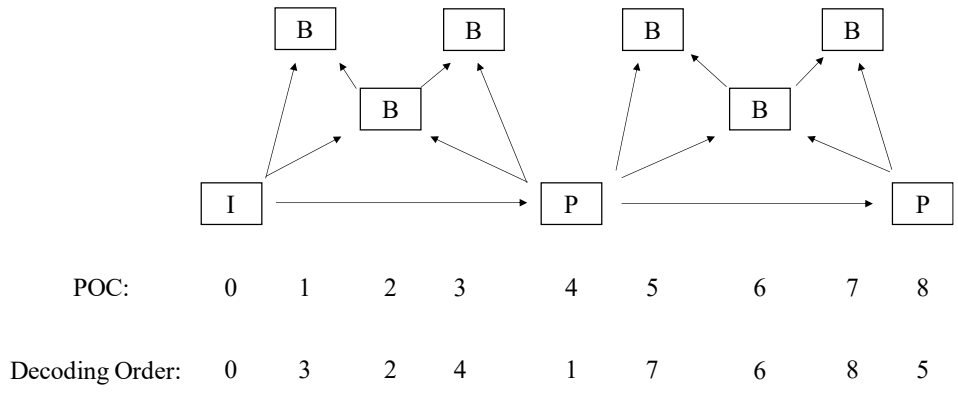


Figure 4-1 A sample GOP structure and the corresponding decoding order.

Next, the space of RGB is transferred to YCbCr space. The reason is that in this space, the information in luminance channel is much more important for human eyes, on the contrary to the RGB space that all the channels carry the same amount of information. Thus, YCbCr space allows employing different subsampling rates for chrominance and luminance channels which will help compression with a higher rate. Many other images and video compression algorithms also use the same YCbCr space. Then each frame is divided into smaller blocks, and the process of motion estimation and motion compensation is performed. After computing motion vectors, the residual coefficients in each subblock are calculated with the estimated motion compensations. Then, the residual coefficients go through the transformation, like integer Discrete Cosine Transform (DCT), and the outputs are quantized based on values that are optimized for humans' vision. Then after scanning or the element (e.g., using zigzag scanning), the bitstream goes into entropy coding.

The decoding is almost similar but in reverse order. A simplified decoder side of hybrid codecs is shown in Figure 4-2. At the decoder side, the input would be the video bitstreams of the encoded video. After entropy decoding, the bit stream needs to go into the inverse of quantization and retransformation operations to get to the reconstructed parameters and coefficients. Then similar to the encoder side, the actual parameters are found by adding the intra or inter prediction values to the residual coefficients. After filtering, the output decoded video would be ready.

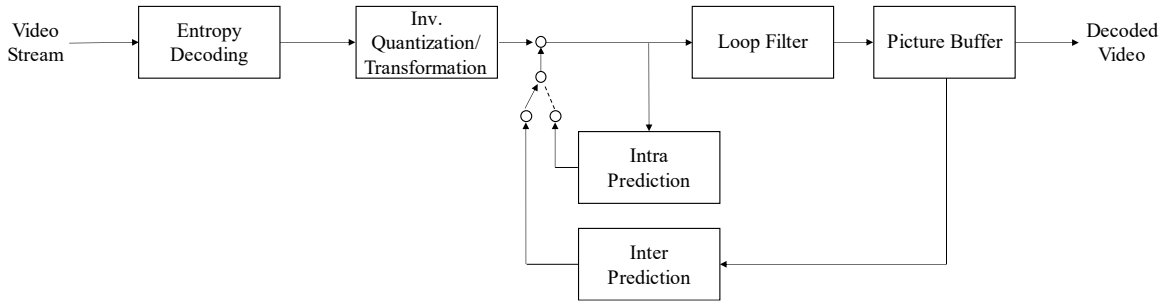


Figure 4-2 Diagram of a simplified hybrid decoder.

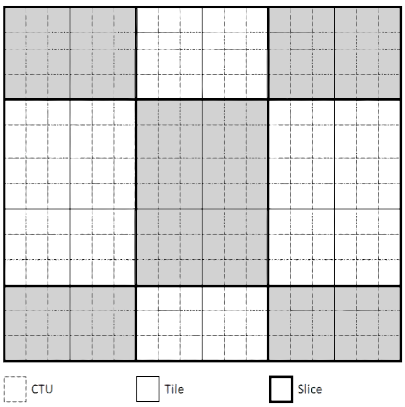
4.2 Versatile Video Coding (H.266/VVC)

Since 2015, there have been many efforts and activities to develop a new compression standard. It was until 2020 that the Joint Video Exploration Team (JVET) of the ITU and ISO/IEC Moving Pictures Expert Group (MPEG) announced the finalization of the H.266/VVC (otherwise known as ISO/IEC 23090-3). This new compression standard provides up to 50% more compression rate compared to the H.265/HEVC standard and supports video resolutions from 4K to 16K in addition to 360-degree videos. The official reference software of H.266/VVC standard is VVC Test Model (VTM) [28]. However, recently a faster and more optimized software named VVenC was released [29]. The new standard is, in fact, very similar to the HEVC since it is still a hybrid video codec. However, H.266/VVC is powered with many new coding tools along with lots of improvements and refinements over the previously implemented technologies. In terms of performance, Laude *et al.* [30] conducted a comprehensive performance comparison of VTM, AV1, HM (HEVC software), x264, and x265 video coding software. Their results show that, on average, VTM gives 5% better BD-rate performance compared to AV1. For the 4K videos, VTM outperforms AV1 with a 20% gain. However, the VTM encoding time is around two to three times of AV1 depending on the tested video sequences. However, we should mention that since the H.266/VVC is announced very recently, there will be more optimizations over its features and implementations in the near future. H.266/VVC is a hybrid codec and very similar to previous standards, thus, we only discuss some of the new features and optimizations of the H.266/VVC over H.265/HEVC that will affect the process of selective encryption.

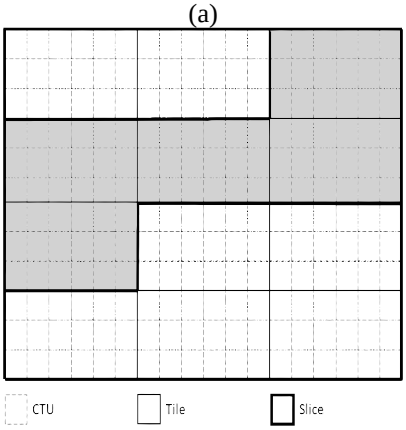
4.2.1 Tiles, Slices, and Partitioning

During the process of compression, similar to H.265/HEVC, each frame is first split into smaller units which are named Coding Tree Units (CTUs). The size of each CTU in the H.265/HEVC standard is 64x64 pixels, but in the H.266/VVC its dimension is increased to 128x128 pixels. Then, a sequence of CTUs is grouped together in rectangular shapes to form tiles. Also, in VVC two modes of slicing are considered, rectangular slice mode and raster scan slice mode. After dividing frames into CTUs, each CTU is further divided into Coding Units (CUs). In the H.265/HEVC, during prediction and transformation, CUs will also be respectively divided into smaller Prediction Units (PUs) and Transformation Units (TUs). However, the concept of separating CU, TU, and PU are no longer in use (except for CUs with a larger size than the maximum transform length).

In fact, VVC uses CUs as the basic processing units. Also, in VVC, two modes of slicing are considered, rectangular slice mode and raster scan slice mode. Figure 4-3 (a) is an image with rectangular slice partitioning where it is divided into 24 tiles and nine rectangular slices. Figure 4-3 (b) is an example of roster scan slice partitioning where the image is divided into three roster scan slices. Moreover, for the purpose of splitting CTUs, two types of hierarchical trees are used: Quadtree (QT) and Multi-Type Tree (MTT). First, CTUs are divided with a quaternary tree then, if needed, the leaf nodes are further partitioned with a multi-type tree. Figure 4-4 shows the QT and MTT splitting along with an example of a sample partitioned frame in VVC. multi-type splitting types are shows in). As a result, the VVC encoder has a wide range of rectangle blocks and shapes for a CTU to split into, which would bring higher compression (as well as higher computation cost). Also, the tiles, slices, and subpictures are separated in the bitstream, which is suitable for parallel processing in the encoder and decoder. This technique would also be helpful in the 360-degree videos since it allows the receiver to only decode the regions of the video that the user is seeing.



(a)



(b)

Figure 4-3 (a) and (b) show two different partitionings [46].

Coding tree splitting in VVC has the capability to assign different block tree structures to chroma and luma components. In one Coding Tree Unit (CTU) of P and B frames, the luma and chroma Coding Tree Blocks (CTBs) are the same structure, unlike in I frames where the structure for chroma and luma might be different. Thus, the partitioning of luma and chroma CTBs can have different coding tree structures. When the number of intra blocks increases in the compression algorithm, the computation cost in both the encoder and the decoder increases. This is due to the fact that there are smaller surrounding units as neighbors, and the processing to find the dependencies between them would become more computationally expensive. During decoding and encoding procedures, the intra prediction is done in sequential manner because it requires the reconstructed modes from thee up and left neighboring blocks.

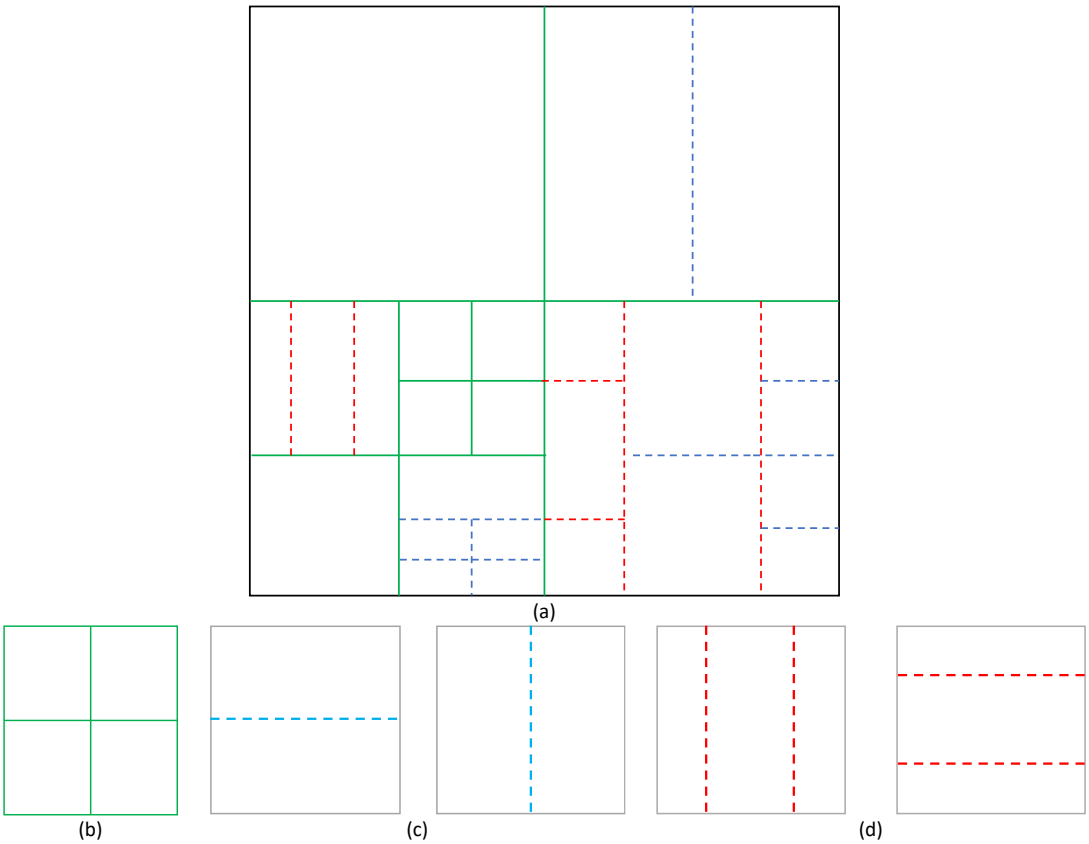


Figure 4-4 (a) shows a sample block partitioning in VVC. (b) is quadtree splitting. (c) is vertical and horizontal binary multi-type tree. (d) is vertical and horizontal ternary multi-type tree.

4.2.2 Intra Prediction

For Intra prediction, there are 67 Intra modes in the H.266/VVC which makes the prediction more accurate compared to H.265/HEVC standard, which has 35 modes. Two of the 67 modes are planar and DC modes. The rest belong to the angular predictions. The new directional modes not in HEVC are depicted as red dotted arrows in Figure 4-5, and the planar and DC modes remain the same. In the new coding standard, the intra prediction is similar to H.265/HEVC. Depending on the upper and left blocks of the current coding block, a list of Most Probable Modes (MPMs) with size of six will be constructed. Then, if the current prediction mode is one of MPM elements, the encoder binarizes its index using truncated unary coding, otherwise index of the remaining 61 modes will be binarized. In the H.265/HEVC, the prediction direction angles were between 45 degrees to -135. However, in the H.266/VVC, because of the employed non-rectangular blocks, the wide-angle intra prediction modes are introduced. For this purpose, no other new modes are used, i.e., the total prediction modes are still 67. So, the wide-angle prediction directions are assigned using the regular modes and are further remapped to specify the wider modes.

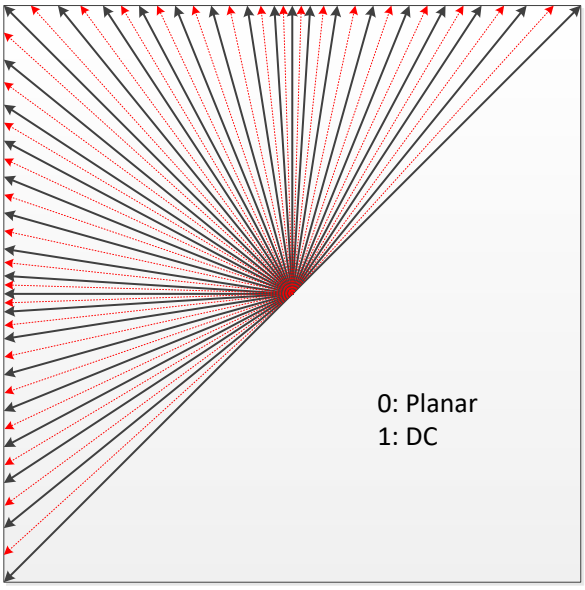


Figure 4-5 Intra prediction modes [31].

Cross Component Linear Model (CCLM) is another new intra prediction tool in the H.266/VVC. CCLM reduces the redundancy and reconstructs the chroma samples based on the reconstructed luma prediction modes of the current coding unit. The chroma intra prediction uses 8 modes. The modes maintain three cross-component linear model modes and five regular intra modes. Chroma mode coding is directly influenced by the luma prediction mode of the corresponding blocks (since multiple luma blocks might have one corresponding chroma block). Table 4-1 shows the binary strings of each intra chroma mode.

Table 4-1 Unified binarization table for chroma prediction mode.

Value of *intra_chroma_pred_mode*	Bin string
4	00
0	0100
1	0101
2	0110
3	0111
5	10
6	110
7	111

One other new feature is Matrix Weighted Intra Prediction (MIP). For predicting a rectangular unit of width *W* and height *H*, MIP takes the above and left surrounding line. In case the calculated components are unavailable, regular intra prediction methods are used for the prediction of samples. MIP consists of three steps, as shows in Figure 4-6.

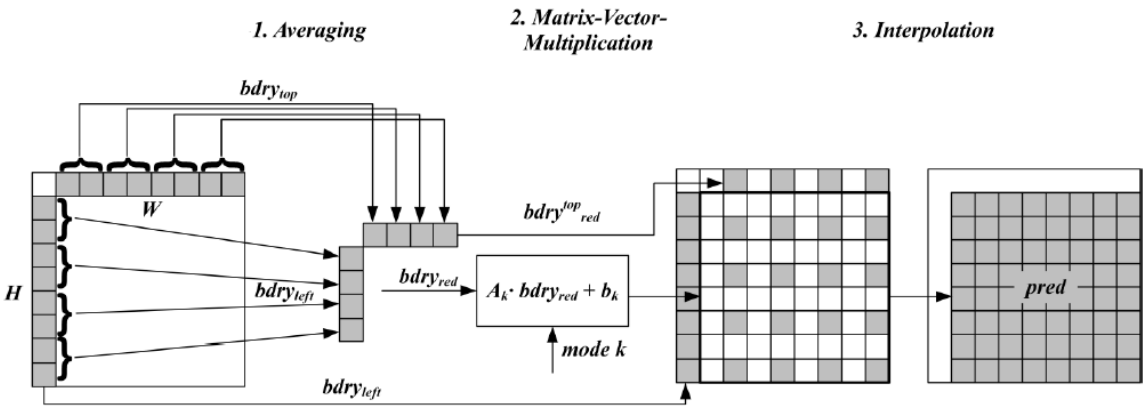


Figure 4-6 Matrix weighted intra prediction [31].

4.2.3 Inter Prediction

In H.266/VVC the inter prediction is performed by parameters such as indices of reference pictures, reference picture list indices, index of reference picture list, motion vectors, and other information. If the skip mode flag is turned on for a coding unit, only a prediction unit is assigned to the coding unit and no index of reference picture or motion vectors are used. Also, if the merge mode is used for inter prediction, the motion parameters of the coding unit are calculated using neighboring coding units (with spatial and temporal candidates and other implemented methods). This is called implicit transmission. The other method is explicit, where the information of motion vectors and reference pictures list and other needed parameters are used for each coding unit. A merge mode is a new tool compared to H.265/HEVC. Moreover, the Merge mode with Motion Vector Differences (MMVD) is also a newly introduced feature in H.266/VVC. In MMVD the merge mode is again refined by the MVD components. After the regular merge flag, the MMVD flag is sent in order to determine if the merge mode with motion vector differences is used. Other components include a merge candidate flag and two indices to determine the motion and indication of the motion vector.

For increasing the accuracy of motion vector merge mode, Bilateral Matching (BM) is introduced which is a new tool of H.266/VVC decoder. This is called the Decoder side Motion

Vector Refinement (DMVR), Figure 4-7 shows an overview of DMVR. The BM is used to measure the distortion between two candidate blocks. For Inter Prediction, VVC has several other new tools. One of the interesting new features is affine motion compensation. In previous codecs like H.265/HEVC, the movement of objects are only in 2D dimensional directions. However, in real videos, we barely see only planar motions, and movements of objects between video frames might accompany rotation or scale (zooming in or out). Thus, affine motion compensation is introduced to represent these more complex motions. There are a couple of other newly introduced tools for inter prediction, among which we can mention Merge Mode with MVD (MMVD), Symmetric MVD Coding, Adaptive Motion Vector Resolution (AMVR), etc. For more information and details of these features readers can refer to VTM algorithm descriptions [31].

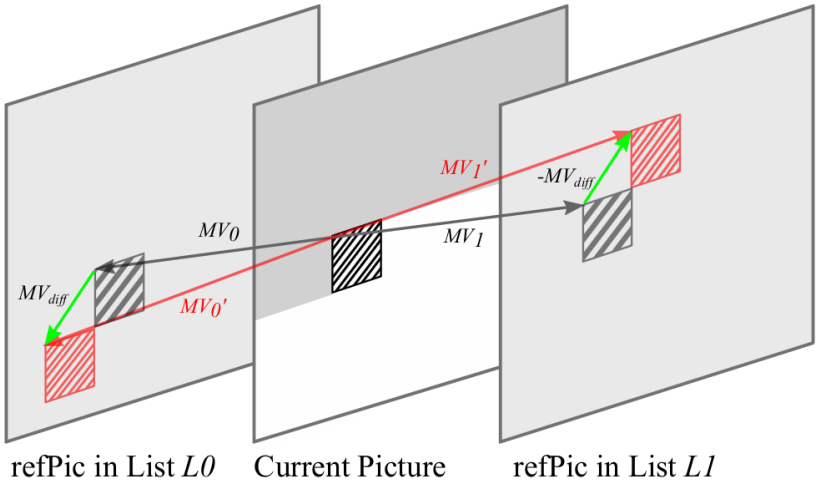


Figure 4-7 Decoder side motion vector refinement [31].

4.3 Summary

This chapter presented the basic description of hybrid video codecs and how they work. It discusses the major steps in video compression algorithms, then the performance of several popular video codecs such as AV1, H.266/VVC, and H.265HEVC are presented. The rest of the chapter is about some of the major new features of H.266/VVC and new enhancements compared to the H.265/HEVC that will be important for Chapter 5, which is about selective encryption of syntax elements in the H.266/VVC compression algorithm.

Chapter 5

Selective Encryption in H.266/VVC

5.1 Related Works on Video Encryption

Among the related literature, there are a variety of proposed methods that discuss how to securely hide video content information. The encryption process can be performed after video encoding where we encrypt some or all of the encoded bitstream, it can be integrated inside the compression algorithm (selective encryption), or the encryption can be performed by scrambling the pixels with chaos-based algorithms. Thus, the performance and characteristics of these methods essentially depend on the domain we apply the encryption to [9]. In [32], for the encryption of MPEG-1 bitstreams, it is proposed to encrypt the complete bitstream or important parts such as headers depending on the desired security level. Since this is considered naive encryption, the format compliance will not be maintained. Muhammad *et al.* [33] present a method for encryption of visual data in IoT systems. They use probabilistic algorithms to detect frames with a required level of abnormality, then the pixels of these keyframes are encrypted using chaotic maps and Pseudorandom Number Generators (PRNGs). A sample of one frame in both original and encrypted format when implementing Muhammad *et al.* [33] scheme is shown in Figure 5-1. Based

(a) (b)

Figure 5-1 Original and encrypted frame when using chaos-based image/video encryption.

on our experiments, we observed that depending on the number of encrypted frames, chaotic encryption algorithms can become quite expensive in terms of computation cost because of the processes used for the scrambling of pixels. Also, in another paper, Preishuber *et al.* show some other problems with chaos-based encryption algorithms [34].

Yang *et al.* [35], propose discrete sine and cosine transforms for the purpose of video encryption in the H.264/AVC. In [36], residual component, intra and inter prediction modes, and MVD components are used for the encryption in H.264/AVC standard. Wallendael *et al.*[37] investigated the encryption of syntax elements in the HEVC encoder, such as MVD values, delta QP, and residual components. They also discussed the effect of encryption on the encoder bit rate and visual scrambling of output frames. In [9], the authors presented an SE algorithm for The Context-Aware Binary Arithmetic Coding (CABAC) in the H.264/AVC and H.265/HEVC standards. They consider encryption of a wider range of elements in the aforementioned standards, including Luma IPM components. The presented experimental results of their work show the impact of selective encryption on the encoder bit rate and visual distortion of video frames. Thiyagarajan *et al.* [38] proposed a more efficient SE algorithm for the Internet of multimedia things. They used a method to estimate the energy level of frames, then based on that energy level, the algorithm decides what syntax elements of the H.265/HEVC standard be used for the encryption. Although it should be noted that the estimation of texture and motion energy levels in each frame is adding extra overhead, thus the real efficiency of the proposed system will get lower. Obviously, because their method is not encrypting all of the syntax elements in the encryption process, the reported metrics used to compare the visual distortion indicate the visual deterioration to be lower compared to other methods. Xu [39], proposes a similar encryption approach using MVD, IPM, and Quantized Transform Coefficient (QTC) elements along with a method for data embedding using QTC values in the H.265/HEVC standard. Authors of [40] tried to encrypt almost all of the important syntax elements of the H.265/HEVC standard and presented their work as a tunable approach. Also, another scrambling method is used to further distort the edges since regular SE algorithms might not completely hide edge regions; off course, this added scrambling process will increase the computation complexity of the proposed method. The authors' results also indicate that the bitrate increase would be between 2% to 10% depending on whether TU coefficients are scrambled or not. Moreover, in Table 5-1, a summary of some of related works on selective encryption are provided. In this paper, we are addressing the selective encryption in

the H.266/VVC to see how the new improvements in this new compression algorithm affects the SE methods. Since no other work is available for SE algorithms on H.266/VVC, we will be comparing the results with previous similar works where authors were using previous compression standards.

Table 5-1 Some of related works for video encryption.

Video Encryptin Scheme	Compression Standard	Cipher	Elements Used for Encryption	Bitrate Increase	Format Compliancy
[9]	H.264/AVC - H.265/HEVC	AES-CTR	Luma IPM, MVD sign and values, MV reference idx, Merge idx, MVP idx, Residual sign and values, SAO filter	Yes	Yes
[32]	MPEG-1	DES-CBC	Headers, DCT coefficients, or whole bitstream (based on security levels)	No	No
[33]	N/A	PRNG	Pixels	No	Yes
[37]	H.265/HEVC	AES	MVD sign and values, Residual sign and values, delta QP, reference pic idx, Merge idx, MVP idx, SAO parameter	Yes	Yes
[38]	H.265/HEVC	AES-CTR	Luma and chroma IPM, QTC, MVD signs and values	Yes	Yes
[39]	H.265/HEVC	RC4	QTC, MVD sign and values, IPM luma	Yes	Yes
[40]	H.265/HEVC	AES-CTR	Chroma and Luma IPM, Residual Sign and values, MVD sign and values, Merge Idx, MVP idx, Reference frame idx, SAO parameter	Yes	Yes

5.2 Methodology

5.2.1 Configuration, Software, and Test Video Sequences

A wide range of video sequences, from low resolution to 4K videos with different frame rates and bit depths, are selected for our tests in order to investigate the results of the proposed SE algorithm (selected test videos are shown in Table 5-2). The software for H.266/VVC is VTM [28]; currently, its latest version is 10.2. There is another software for VVC coding named VVenC [29], which is actually much faster compared to VTM. We can look at VTM as the complete software with all of the tools and options, while VVenC provides a faster implementation of H.266/VVC. However,

unfortunately, the VVenC only has the encoder, and the decoder side is not available yet. Thus, for our simplicity during the tests, we used the VVenC for the encoding and VTM for decoding video streams. We have used the *randomaccess_faster* configuration of the VVenC software, which has *GOPsize* of 32 and *intraPeriod* of 32. In this case, there will be one I frame with 31 following B frames. The base Quantization Parameter (QP) is also selected to vary from values 8, 24, and 32 depending on the performed tests. The *level* value which determines the tire that the encoded bitstream compiles to is chosen from Table 5-3 and Table 5-4 of [41] for each corresponding test video. Its value is dependent to video resolution (luma height and width) and frame rate of the video under test. In Figure 5-2, we can see diagram of the VTM software with a selective encryption block added prior to the CABAC.

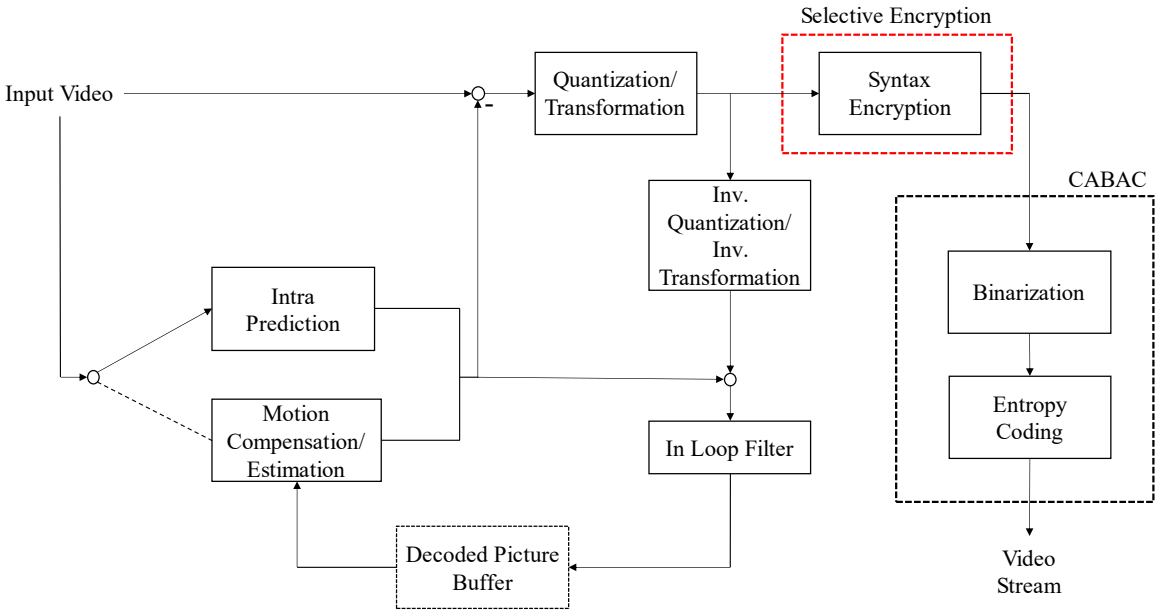


Figure 5-2 Block diagram of VTM software with the selective encryption block added before CABAC.

Table 5-2 Video sequences used for experiments.

Video	Resolution	Frame Rate	Bit depth
Mobile	352x288	24	8
BasketballPass	416x240	50	8
BQMall	832x480	60	8
Johnny	1280x720	60	8
FourPeople	1280x720	60	8
BasketballDrive	1920x1080	50	8
PeopleOnStreet	2560x1600	30	8
Traffic	2560x1600	30	8
RaceNight	3840x2160	50	10
HoneyBee	3840x2160	120	10

Table 5-3 Maximum frame rate for main level in VTM. level 1.0 to 4.1.

Level:				**1.0**	**2.0**	**2.1**	**3.0**	**3.1**	**4.0**	**4.1**
Max luma picture size (samples):				36 864	122 880	245 760	552 960	983 040	2 228 224	2 228 224
Max luma sample rate (samples/sec)				552 960	3 686 400	7 372 800	16 588 800	33 177 600	66 846 720	133 693 440
Format nickname	**Luma width**	**Luma height**	**Luma picture size**							
SQCIF	**128**	**96**	16 384	33.7	225.0	450.0	960.0	960.0	960.0	960.0
QCIF	**176**	**144**	36 864	15.0	100.0	200.0	450.0	900.0	960.0	960.0
QVGA	**320**	**240**	81 920	-	45.0	90.0	202.5	405.0	816.0	960.0
525 SIF	**352**	**240**	98 304	-	37.5	75.0	168.7	337.5	680.0	960.0
CIF	**352**	**288**	122 880	-	30.0	60.0	135.0	270.0	544.0	960.0
525 HHR	**352**	**480**	196 608	-	-	37.5	84.3	168.7	340.0	680.0
625 HHR	**352**	**576**	221 184	-	-	33.3	75.0	150.0	302.2	604.4
Q720p	**640**	**360**	245 760	-	-	30.0	67.5	135.0	272.0	544.0
VGA	**640**	**480**	327 680	-	-	-	50.6	101.2	204.0	408.0
525 4SIF	**704**	**480**	360 448	-	-	-	46.0	92.0	185.4	370.9
525 SD	**720**	**480**	393 216	-	-	-	42.1	84.3	170.0	340.0
4CIF	**704**	**576**	405 504	-	-	-	40.9	81.8	164.8	329.6
625 SD	**720**	**576**	442 368	-	-	-	37.5	75.0	151.1	302.2
480p (16:9)	**864**	**480**	458 752	-	-	-	36.1	72.3	145.7	291.4
SVGA	**800**	**600**	532 480	-	-	-	31.1	62.3	125.5	251.0
QHD	**960**	**540**	552 960	-	-	-	30.0	60.0	120.8	241.7
XGA	**1 024**	**768**	786 432	-	-	-	-	42.1	85.0	170.0
720p HD	**1 280**	**720**	983 040	-	-	-	-	33.7	68.0	136.0
4VGA	**1 280**	**960**	1 228 800	-	-	-	-	-	54.4	108.8
SXGA	**1 280**	**1 024**	1 310 720	-	-	-	-	-	51.0	102.0
525 16SIF	**1 408**	**960**	1 351 680	-	-	-	-	-	49.4	98.9
16CIF	**1 408**	**1 152**	1 622 016	-	-	-	-	-	41.2	82.4
4SVGA	**1 600**	**1 200**	1 945 600	-	-	-	-	-	34.3	68.7
1080 HD	**1 920**	**1 080**	2 088 960	-	-	-	-	-	32.0	64.0
2K×1K	**2 048**	**1 024**	2 097 152	-	-	-	-	-	31.8	63.7
2K×1080	**2 048**	**1 080**	2 228 224	-	-	-	-	-	30.0	60.0
4XGA	**2 048**	**1 536**	3 145 728	-	-	-	-	-	-	-
16VGA	**2 560**	**1 920**	4 915 200	-	-	-	-	-	-	-
3616×1536 (2.35:1)	**3 616**	**1 536**	5 603 328	-	-	-	-	-	-	-
3672×1536 (2.39:1)	**3 680**	**1 536**	5 701 632	-	-	-	-	-	-	-
3840×2160 (4*HD)	**3 840**	**2 160**	8 355 840	-	-	-	-	-	-	-
4K×2K	**4 096**	**2 048**	8 388 608	-	-	-	-	-	-	-
4096×2160	**4 096**	**2 160**	8 912 896	-	-	-	-	-	-	-
4096×2304 (16:9)	**4 096**	**2 304**	9 437 184	-	-	-	-	-	-	-
7680×4320	**7 680**	**4 320**	33 423 360	-	-	-	-	-	-	-
8192×4096	**8 192**	**4 096**	33 554 432	-	-	-	-	-	-	-
8192×4320	**8 192**	**4 320**	35 651 584	-	-	-	-	-	-	-

Table 5-4 Maximum frame rate for main level in VTM. level 5.0 to 6.2.

Level:				**5.0**	**5.1**	**5.2**	**6.0**	**6.1**	**6.2**
Max luma picture size (samples):				8 912 896	8 912 896	8 912 896	35 651 584	35 651 584	35 651 584
Max luma sample rate (samples/sec)				267 386 880	534 773 760	1 069 547 520	1 069 547 520	2 139 095 040	4 278 190 080
Format nickname	**Luma width**	**Luma height**	**Luma picture size**						
SQCIF	**128**	**96**	16 384	960.0	960.0	960.0	960.0	960.0	960.0
QCIF	**176**	**144**	36 864	960.0	960.0	960.0	960.0	960.0	960.0
QVGA	**320**	**240**	81 920	960.0	960.0	960.0	960.0	960.0	960.0
525 SIF	**352**	**240**	98 304	960.0	960.0	960.0	960.0	960.0	960.0
CIF	**352**	**288**	122 880	960.0	960.0	960.0	960.0	960.0	960.0
525 HHR	**352**	**480**	196 608	960.0	960.0	960.0	960.0	960.0	960.0
625 HHR	**352**	**576**	221 184	960.0	960.0	960.0	960.0	960.0	960.0
Q720p	**640**	**360**	245 760	960.0	960.0	960.0	960.0	960.0	960.0
VGA	**640**	**480**	327 680	816.0	960.0	960.0	960.0	960.0	960.0
525 4SIF	**704**	**480**	360 448	741.8	960.0	960.0	960.0	960.0	960.0
525 SD	**720**	**480**	393 216	680.0	960.0	960.0	960.0	960.0	960.0
4CIF	**704**	**576**	405 504	659.3	960.0	960.0	960.0	960.0	960.0
625 SD	**720**	**576**	442 368	604.4	960.0	960.0	960.0	960.0	960.0
480p (16:9)	**864**	**480**	458 752	582.8	960.0	960.0	960.0	960.0	960.0
SVGA	**800**	**600**	532 480	502.1	960.0	960.0	960.0	960.0	960.0
QHD	**960**	**540**	552 960	483.5	960.0	960.0	960.0	960.0	960.0
XGA	**1 024**	**768**	786 432	340.0	680.0	960.0	960.0	960.0	960.0
720p HD	**1 280**	**720**	983 040	272.0	544.0	960.0	960.0	960.0	960.0
4VGA	**1 280**	**960**	1 228 800	217.6	435.2	870.4	870.4	960.0	960.0
SXGA	**1 280**	**1 024**	1 310 720	204.0	408.0	816.0	816.0	960.0	960.0
525 16SIF	**1 408**	**960**	1 351 680	197.8	395.6	791.2	791.2	960.0	960.0
16CIF	**1 408**	**1 152**	1 622 016	164.8	329.6	659.3	659.3	960.0	960.0
4SVGA	**1 600**	**1 200**	1 945 600	137.4	274.8	549.7	549.7	960.0	960.0
1080 HD	**1 920**	**1 080**	2 088 960	128.0	256.0	512.0	512.0	960.0	960.0
2K×1K	**2 048**	**1 024**	2 097 152	127.5	255.0	510.0	510.0	960.0	960.0
2K×1080	**2 048**	**1 080**	2 228 224	120.0	240.0	480.0	480.0	960.0	960.0
4XGA	**2 048**	**1 536**	3 145 728	85.0	170.0	340.0	340.0	680.0	960.0
16VGA	**2 560**	**1 920**	4 915 200	54.4	108.8	217.6	217.6	435.2	870.4
3616×1536 (2.35:1)	**3 616**	**1 536**	5 603 328	47.7	95.4	190.8	190.8	381.7	763.5
3672×1536 (2.39:1)	**3 680**	**1 536**	5 701 632	46.8	93.7	187.5	187.5	375.1	750.3
3840×2160 (4*HD)	**3 840**	**2 160**	8 355 840	32.0	64.0	128.0	128.0	256.0	512.0
4K×2K	**4 096**	**2 048**	8 388 608	31.8	63.7	127.5	127.5	255.0	510.0
4096×2160	**4 096**	**2 160**	8 912 896	30.0	60.0	120.0	120.0	240.0	480.0
4096×2304 (16:9)	**4 096**	**2 304**	9 437 184	-	-	-	113.3	226.6	453.3
4096×3072	**4 096**	**3 072**	12 582 912	-	-	-	85.0	170.0	340.0
7680×4320	**7 680**	**4 320**	33 423 360	-	-	-	32.0	64.0	128.0
8192×4096	**8 192**	**4 096**	33 554 432	-	-	-	31.8	63.7	127.5
8192×4320	**8 192**	**4 320**	35 651 584	-	-	-	30.0	60.0	120.0

5.2.2 Encryption of Syntax Elements

The encryption is applied prior to the binarization of syntax elements (as shown in Figure 5-3). There are many non-binary syntax elements that are required to be binarized prior to encoding, such as MVD values, residual values, etc. Thus, binarization can be seen as a preprocessing step that maps each value to a stream of zeros and ones. In hybrid codecs, the binarization coding of each syntax element might be different is selected in a way to ensure syntax elements' suitable probability distribution and efficient implementation. Some of the binarization methods are Fixed Length (FL), *k*-th other Exp-Golomb (EGK), *k*-th order Truncated Rice (TRK), and unary coding. Also, Table 5-5 from [42] shows examples of different binarization codings.

Table 5-5 Examples of different binarizations used in video codings with n=8.

N	Unary	Truncated Unary	Truncated Rice (K=1)	Exp-Golomb (K=0)	Fixed Length
0	0	0	00	1	000
1	10	10	01	010	001
2	110	110	100	011	010
3	1110	1110	101	00100	011
4	11110	11110	1100	00101	100
5	111110	111110	1101	00110	101
6	1111110	1111110	1110	00111	110
7	11111110	1111111	1111	0001000	111

With unary coding binarization, a symbol *n* is binarized with *n* zeros (or ones) and a single one (zero) at the end. Truncated binary coding is similar to unary coding, but the maximum value of *n* does not have the final single one or zero bin. Golomb codes are composed of prefix and suffix binary streams, where prefix is a unary coding of quotient *q* and suffix is the remaining *r* which is truncated binary coded. Rice coding is a subset of Golomb coding where *m* is the power of two. Rice coding is suitable in computers since the multiplication and division of two can be performed more efficiently with shift operations. *k*-th order Exp-Golomb coding is a robust and near-optimal

coding which is suitable for geometrically distributed syntax elements with varying or unknown distribution parameters [42]. Similar to HEVC, Context-Aware Binary Arithmetic Coding (CABAC) in VVC has two *regular* and *bypass* modes for encoding. In *bypass* mode, all symbols are considered as equiprobable for the encoding.

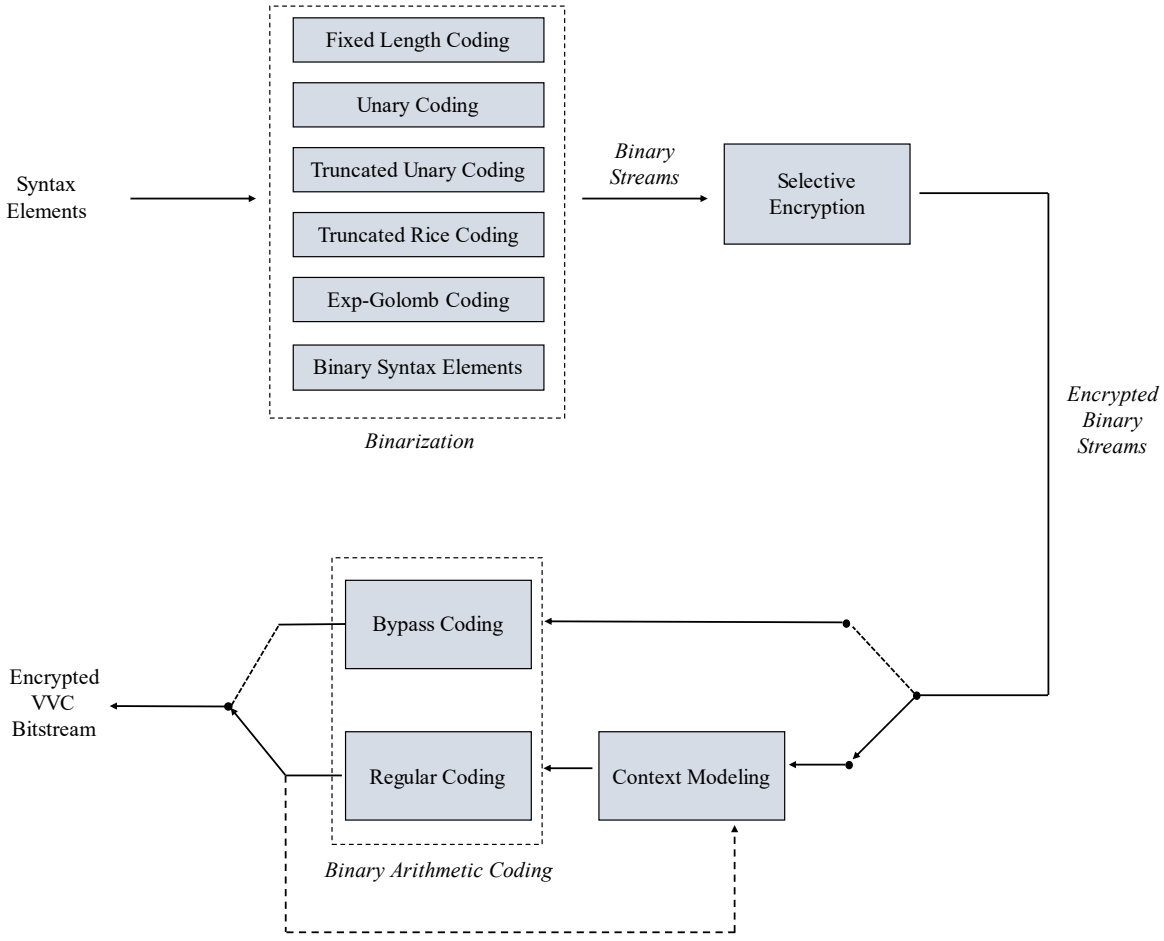


Figure 5-3 Context Aware Binary Arithmetic Coding (CABAC) and selective encryption.

For *Regular* mode, a probability model is determined using the context of elements and the encoding is performed based on this probability model. Thus, encryption of syntax elements encoded with bypass mode would be more suitable since it does not affect the probability model, which would lower the encoding efficiency. For the encryption of elements that are encoded in *regular* mode, we should also make sure that their probability model is also changed accordingly. In the proposed method, the key syntax elements that are considered for the encryption in the

H.266/VVC algorithm are luma IPM value, horizontal and vertical values/signs of motion vectors, and signs of residual values. The reason for the selection of these elements is that they have a very high effect on the final frame disturbance and also when considering all of them together for the encryption it yields an steady highly distorted output over all types of frames (e.g., I and B). For the encryption process, a binary stream needs to be generated using an encryption algorithm, and then the XOR operation is made between the stream and syntax elements. For this purpose, it is recommended to use the Advanced Encryption Standard (AES) in CTR mode of operation for encryption and decryption. The x and y position values of each unit can be used for the Initialization Vector (IV) of the encryption algorithm (e.g., AES-CTR), which will maintain the security of the key during the encryption of all syntax elements. Since some of the syntax elements have a specific range (e.g., luma most probable modes are from 0 to 5), the XOR operation should be performed carefully so that the final value is not outside of the available range of the corresponding syntax element. Without such considerations, because of the format compliancy of the video bitstream, the decoder might face some problems. Thus, we choose a desirable number of last bits in the binary stream generated from the AES-CTR, and then we do the XOR operation if its value after XOR is not beyond the range of available values. For the encryption of luma IPM modes, the encoder either encodes the index of Most Probable Mode (MPM) (predicted from the neighboring units) or it encodes the index of 61 remaining modes for the binarization (when the mode is not among the MPM modes). Encoding of the MPM uses *bypass* mode and requires five bits since it uses truncated unary coding for binarization. For the remaining IPM modes, the six bits fixed-length coding is used, and encoding is done in *bypass* mode. Thus, the encryption is applied to luma modes according to the selected mode for the luma intra prediction and their respective binarization. For the format compliance, when MPM modes are being used, the encrypted syntax should not be greater than 5. Also, when the prediction mode is not among the MPM modes, the encrypted remaining IPM model has a maximum range of 61 because six MPM modes are removed from all 67 available luma modes. Motion vectors constitute of coefficients and corresponding signs for horizontal and vertical directions. The absolute values are encoded using two flags in *regular* mode, (*abs_mvd_greater_0* and *abs_mvd_greater*_1) and *abs_mvd_minus_2* which is *bypass* coded with golomb-rice binarization. We only have chosen to encrypt the *abs_mvd_minus_2* and corresponding signs since they are encoded in *bypass* mode.

Another syntax element selected for encryption is *signPattern* which represents signs of residual coefficients and is encoded in *bypass* mode.

5.3 Results

5.3.1 Visual Security

The encryption of selected syntax elements results in highly distorted videos which indicates that the details and information of videos are securely hidden. Figure 5-4 shows the visual disturbance caused by the encryption of each syntax element. Based on these sample frames, we observe the

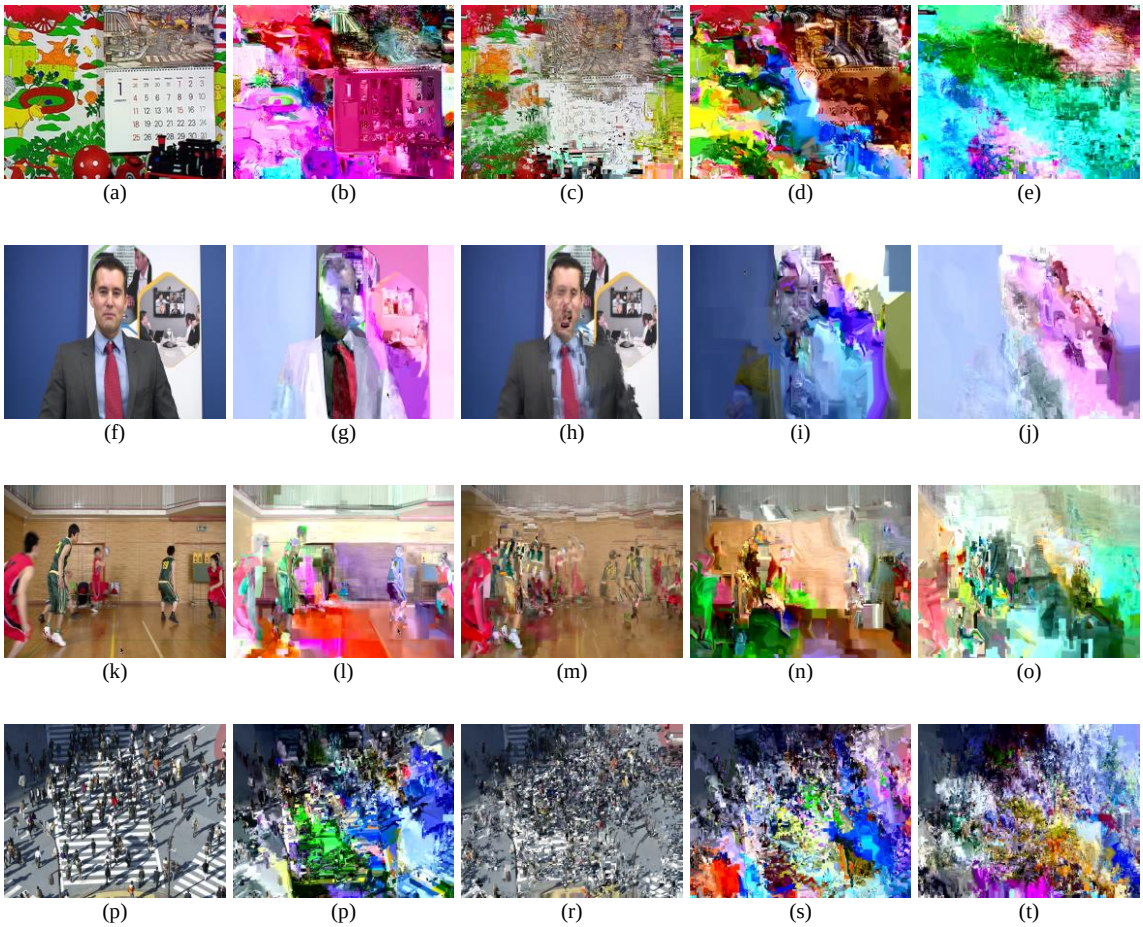


Figure 5-4 Visual results of experimental tests. First column shows the original frames, second column is when residual components are encrypted, third column is when MVD values and signs are encrypted, forth column is when luma modes are encrypted, and fifth column is when all the mentioned syntaxes are encrypted.

importance of luma IPM modes in the H.266/VVC videos since their encryption causes most of the deterioration in the output video (as can be seen in column four of Figure 5-4). After luma modes, MVD modes are the next key syntax elements that their encryption is causing the highest deterioration in final videos (column three Figure 5-4). However, residual signs have a smaller impact on the output video, as can be seen from column two of Figure 5-4. The quantitative performance of the videos are compared using PSNR, SSIM [43], and VMAF [44], [45]. SSIM

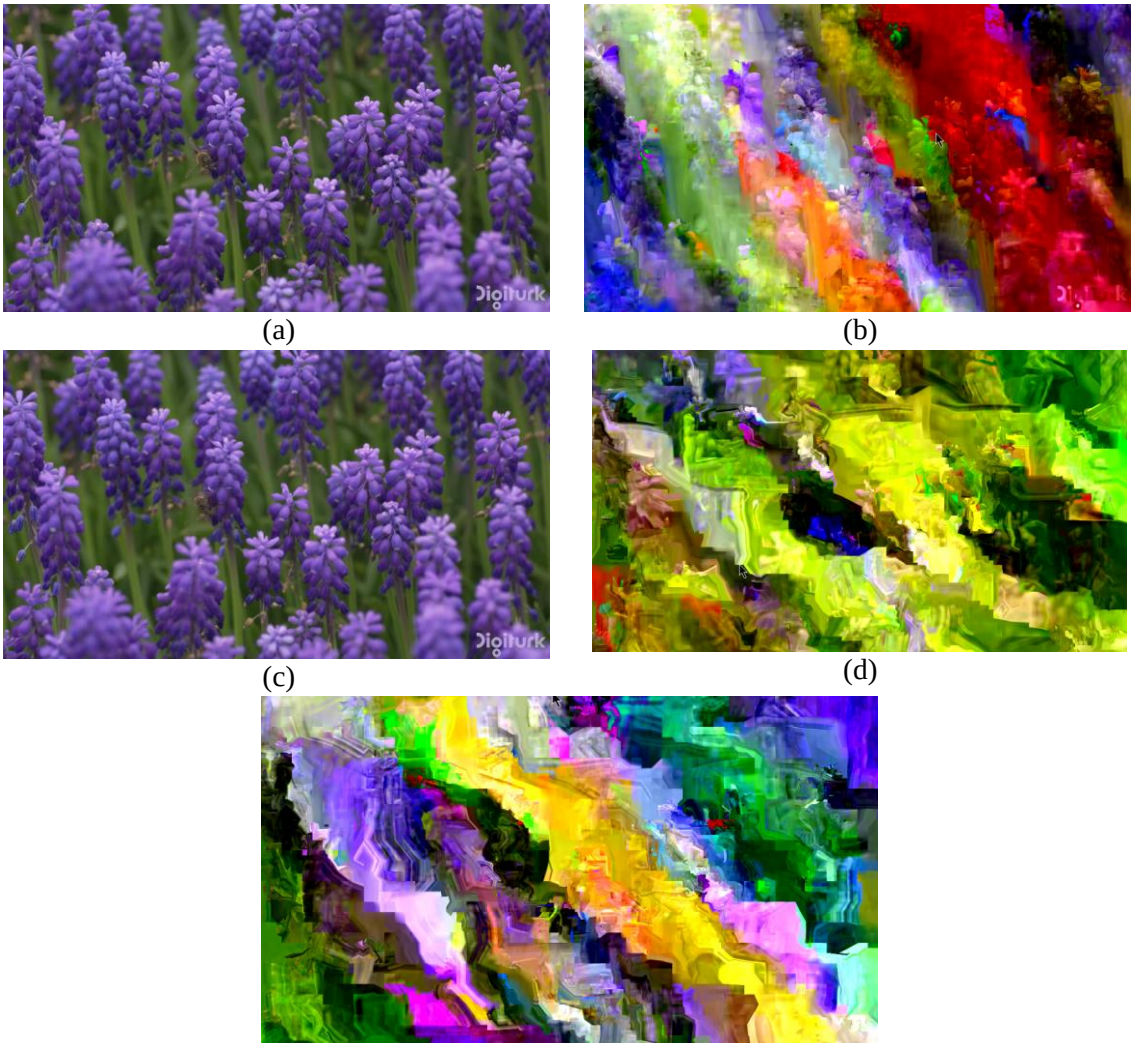


Figure 5-5 Visual results of experimental tests. (a) shows the original 3840x2160 frame of a moving bee with stationary background, (b) is when residual components are encrypted, (c) is when MVD values and signs are encrypted, (d) is when luma modes are encrypted, and (e) is when all the mentioned syntaxes are encrypted.

shows the structural similarity index which compares the processed images (the encrypted video frames for our application) and the original images (when the video is encoded with VVC but is not encrypted). To measure the performance of an encryption algorithm, the lower the SSIM the better the performance of the algorithm would be. PSNR is another metric that is not very good for our application; however, because of its popularity we have employed this metric as well. Video Multimedia Assessment Fusion (VMAF) is a relatively new metric used for video quality measurement, which is developed by Netflix. The VMAF returns a score between 0 to 100, and a higher score shows more resemblance between the two given original and processed videos. In Table 5-6, the performance results from the mentioned metrics are provided for selected test videos while using the encryption. We can see that the SSIM values for encrypted videos are much lower compared to the original values. The PSNR values also confirm the effect of encryption on visual deterioration of encrypted videos. In some cases, we observe mediocre PSNR scores even though the SSIM is very low and video quality is mostly disturbed. This shows the fact that PSNR might not be a very good metric here. Also, the reason that the VMAF scores for some original videos are not as high as it should be (close to 100) is that we have not used the 4K models. VMAF is, in fact, a metric that is trained with resolutions up to 1080p, as a result, we see mediocre scores for some of the original videos. However, these scores can still be used as a comparative metric between videos. The reported values in the Table 5-6 are achieved from the average of the first 64 frames in the selected test videos, however, In Figure 5-6, we can see the SSIM, PSNR, and VMAF results for each of the first 64 frames. When only encrypting the MVD elements (signs and values for horizontal and vertical directions), the I frames would not have enough visual disturbance (e.g., frame 32 in Figure 5-6). However, Encryption of Luma components results in a steady, highly deteriorated output for all frames. Moreover, because the residual signs are not highly important in hiding video information, only encrypting them will not yield acceptable disturbance in the encrypted videos. Because our work is the first one that looks at selective encryption in the H.266/VVC, we only compare the results with other similar methods that are used in H.265/HEVC and H.264/AVC compression standards.

In Table 5-7, we see a comparison with some of the other recent papers for SE algorithms in H.265/HEVC. From this comparison, we see the higher disturbance that is caused in the VVC encoded videos by presented selective encryption algorithms. In most of the cases, we notice the performance of selective encryption to be higher compared with state of the art works in HEVC

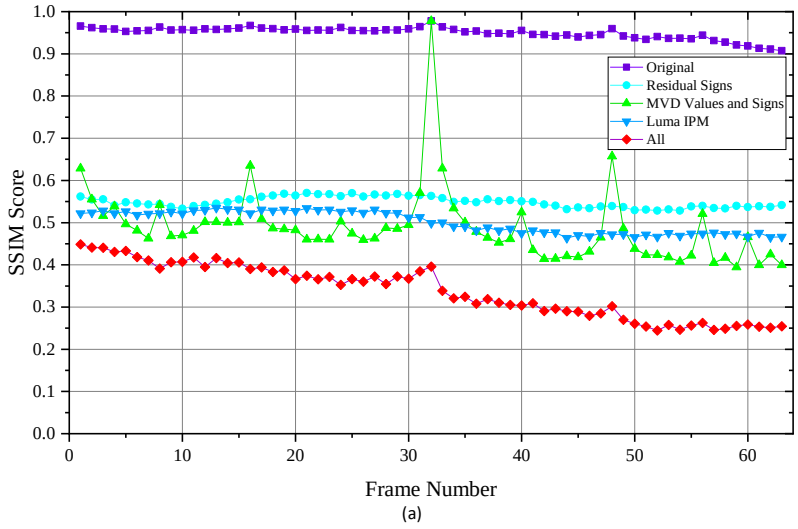
standard. In [40], authors select almost all of the key syntax elements for the encryption in the H.265/HEVC video encoder, but here, we are encrypting only some of the key syntax elements. If wider range of elements were considered in our scheme for the selective encryption, the quantitative results should even become better. Also, we should mention that the computation cost depends on how many syntax elements and how many units are selected for the encryption. The ideal case would be when the video disturbance is high, i.e., high security of video, while the computation cost remains low.

Table 5-6 Performance of the scheme in terms of SSIM, PSNR and VMAF for each video sequence.

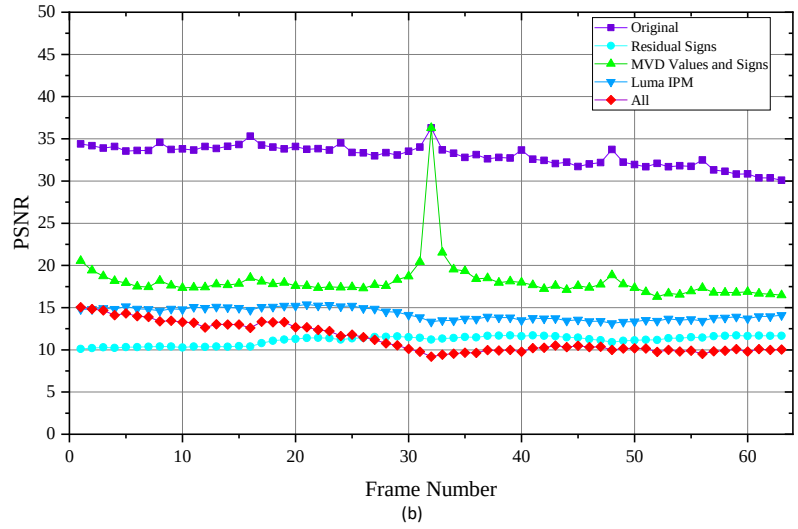
Video Sequence	QP	Original			Encrypted		
		SSIM	PSNR	VMAF	SSIM	PSNR	VMAF
Mobile	8	0.997	49.81	99.959	0.078	9.235	6.642
	24	0.98	35.78	99.234	0.075	8.819	6.190
	40	0.876	26.25	75.375	0.072	9.027	6.068
BasketballPass	8	0.995	50.52	99.339	0.362	13.21	2.877
	24	0.972	40.48	94.397	0.419	15.01	2.254
	40	0.834	30.51	52.556	0.43	10.17	2.2
BQMall	8	0.999	49.88	99.959	0.201	10.62	5.015
	24	0.989	38.47	98.745	0.216	10.81	5.457
	40	0.912	29.39	66.379	0.227	9.8	6.445
Johnny	8	0.998	50.12	98.248	0.397	8.596	0
	24	0.992	42.34	95.258	0.476	9.510	0
	40	0.976	36.47	79.389	0.557	11.74	0
FourPeople	8	0.999	50.09	98.339	0.227	8.75	0
	24	0.995	42.22	94.905	0.234	9.07	0
	40	0.974	34.71	75.771	0.266	9.93	0.005
BasketballDrive	8	0.999	50.18	99.958	0.292	12.95	1.342
	24	0.994	38.81	99.875	0.321	12.06	1.618
	40	0.949	33.08	65.037	0.340	11.39	1.826
PeopleOnStreet	8	0.999	50.25	99.959	0.085	9.90	6.658
	24	0.998	38.86	99.864	0.096	10.02	7.033
	40	0.977	29.48	62.668	0.099	9.79	6.973
Traffic	8	0.999	49.85	99.959	0.11	10.09	3.899
	24	0.998	40.65	95.48	0.126	9.28	3.815
	40	0.982	32.69	67.681	0.135	9.16	3.638
RaceNight	8	0.999	50.35	99.956	0.314	10.57	0
	24	0.997	37.37	96.456	0.355	7.54	0
	40	0.983	34.60	69.107	0.531	14.97	0
HoneyBee	8	0.999	50.26	97.536	0.212	10.14	8.888
	24	0.997	39.17	83.673	0.241	12.41	6.354
	40	0.991	37.67	65.464	0.259	9.72	6.445

Table 5-7 Comparison of experimental results with other proposed selective encryption algorithms.

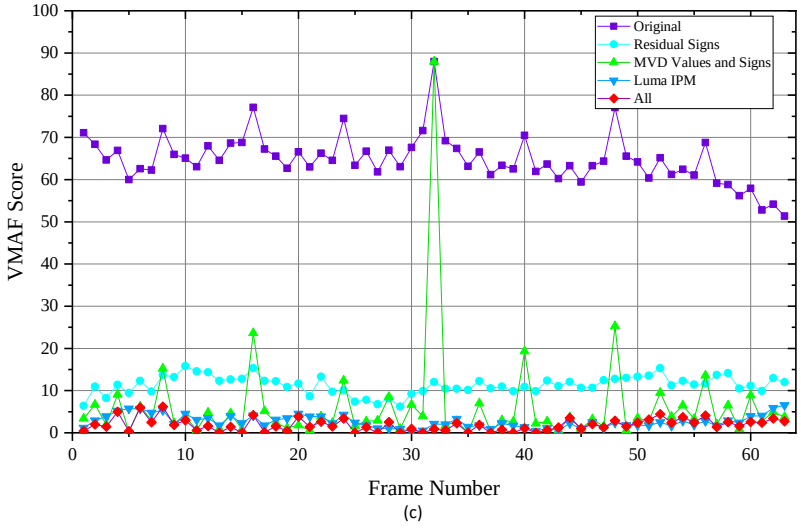
Video Sequence	QP	SSIM			PSNR		
		Boyadjis (for HEVC) [9]	State of the art (for HEVC) [40] – Enc	Presented Scheme (for VVC)	Boyadjis (for HEVC) [9]	State of the art (for HEVC) [40] - Enc	Presented Scheme (for VVC)
Mobile	8	0.070	**0.058**	0.078	10.81	10.89	**9.23**
	24	0.077	0.076	**0.075**	10.64	10.53	**8.82**
	40	0.110	0.097	**0.072**	11.46	10.59	**9.03**
BasketballPass	8	0.320	**0.260**	0.362	15.21	**14.89**	15.21
	24	0.408	**0.372**	0.419	15.48	**15.40**	15.51
	40	0.457	0.459	**0.43**	16.39	16.94	**14.17**
BQMall	8	0.238	0.215	**0.201**	13.96	14.56	**10.62**
	24	0.301	0.282	**0.216**	14.32	14.54	**10.81**
	40	0.332	0.312	**0.227**	14.82	14.40	**9.8**
Johnny	8	0.468	0.449	**0.397**	13.38	13.87	**8.60**
	24	0.569	0.552	**0.476**	13.77	13.68	**9.51**
	40	0.592	0.580	**0.557**	13.41	13.40	**11.746**
FourPeople	8	0.325	0.295	**0.227**	12.76	13.13	**8.75**
	24	0.402	0.381	**0.234**	13.61	13.55	**9.07**
	40	0.420	0.389	**0.266**	13.07	12.83	**9.93**
BasketballDrive	8	0.492	0.496	**0.321**	14.65	15.21	**12.955**
	24	0.545	0.509	**0.321**	14.72	15.00	**12.068**
	40	0.582	0.560	**0.340**	15.49	15.26	**11.391**
PeopleOnStreet	8	0.250	0.232	**0.085**	12.93	13.12	**9.90**
	24	0.294	0.262	**0.096**	13.23	13.10	**10.018**
	40	0.332	0.312	**0.099**	13.09	13.23	**9.79**
Traffic	8	0.260	0.240	**0.11**	12.31	12.53	**10.09**
	24	0.348	0.328	**0.126**	12.81	12.70	**9.28**
	40	0.372	0.361	**0.135**	13.52	13.34	**9.16**



(a)

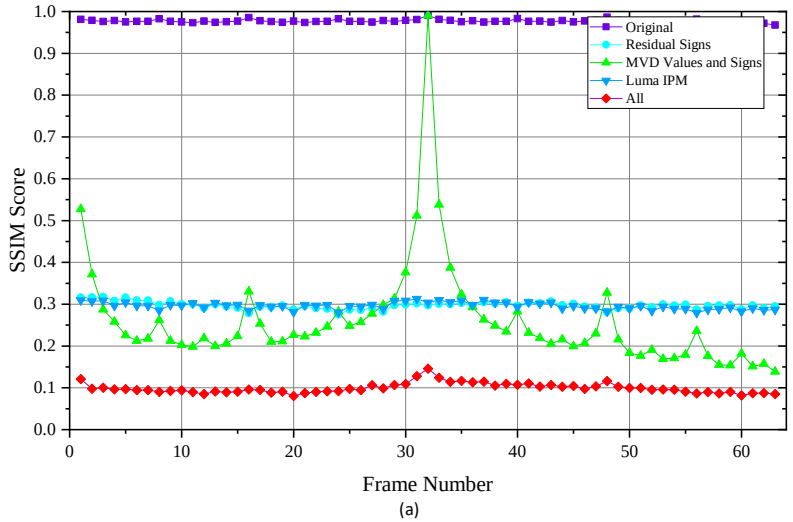


(b)

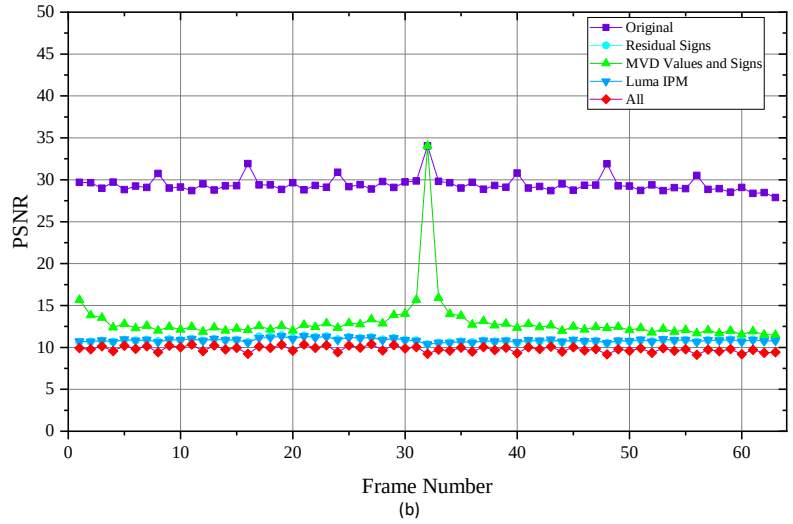


(c)

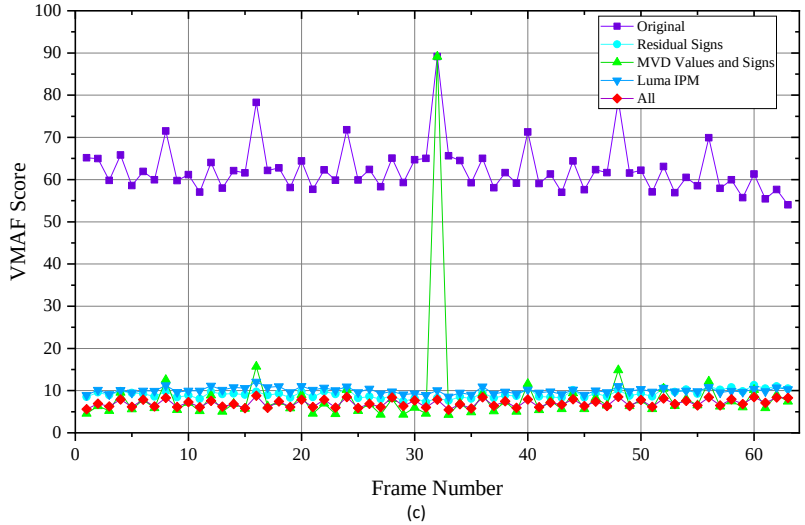
Figure 5-6 Visual quality results after encrypting selected syntax elements for first 64 frames in BasketballDrive video sequence.



(a)



(b)



(c)

Figure 5-7 Visual quality results after encrypting selected syntax elements for first 64 frames in PeopleOnStreet video sequence.

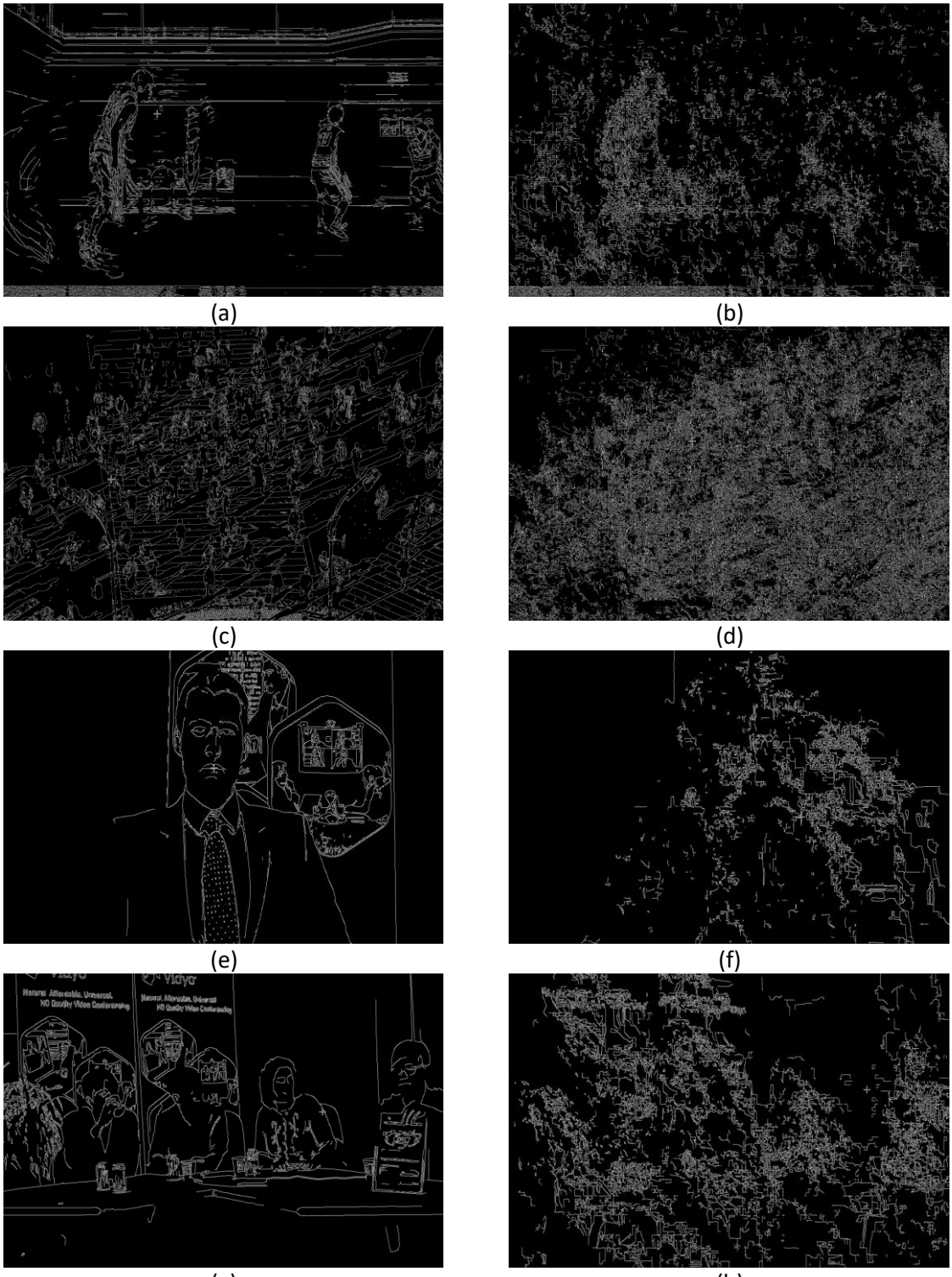


Figure 5-8 Highlighted edges in some of video test sequences after and before performing video encryption. Images in first column are the original frames and the second column contains the corresponding encrypted frames.

Edges of regions in the encrypted video frames play an important factor for the visual security of encrypted video using selective encryption methods. If the selective encryption scheme is not good enough edges of the encrypted videos might leak some information about the objects in frames. Figure 5-8 shows several of the sample frames in both original (row one) and encrypted forms (row two). We can observe that an adversary cannot get any important information by considering the edges in the encrypted video. To validate the security of edges in the proposed method, we have considered the Edge Differential Ratio (EDR) as was employed in [8]. The EDR is defined as:

$$\text{EDR}=\frac{\sum_{m,n=1}^{M}| P(m, n) - \bar{P}(m, n)|}{\sum_{m,n=1}^{M}| P(m, n)+\bar{P}(m, n)|} \qquad (5.1)$$

Where M is the number of edge pixels, $P(m,n)$ is the value of edge pixel in the original video frame, and $\bar{P}(m,n)$ is the value of edge pixel in the encrypted video. In Table 5-8, the average of EDR values for first 64 frames are listed for different QP values. Reported values confirm that the similarity of edges in encrypted and original frames are very small (low EDR indicate high edge similarity between given images). Note that this metric is completely dependent on the implemented edge detection algorithm. Also, from table V, we see that for QP=40, EDR values in

Table 5-8 Average EDR scores of encrypted and original frames.

Video Sequence	EDR					
	QP = 8		QP = 24		QP = 40	
	Org.	Enc.	Org.	Enc.	Org.	Enc.
Mobile	0.035	0.952	0.135	0.98	0.353	0.992
BasketballPass	0.108	0.91	0.255	0.928	0.433	0.968
BQMall	0.082	0.911	0.266	0.944	0.531	0.968
Johnny	0.239	0.911	0.357	0.964	0.449	0.967
FourPeople	0.194	0.931	0.35	0.946	0.535	0.938
BasketballDrive	0.141	0.781	0.296	0.985	0.452	0.986
PeopleOnStreet	0.192	0.946	0.316	0.98	0.557	0.991
Traffic	0.195	0.957	0.396	0.987	0.444	0.983

original values are relatively high, however, it does not indicate that the two compared videos are very different. This is due to the fact that while using large QP values, many of the edges strength will change, which results in a higher EDR value.

5.3.2 Bit Rate Change

Because SE methods are integrated into the compression algorithms, many of the coefficients and values will be changed, resulting in a different probability model for the syntax elements. For example higher MVD values have lower probability of, however after encrypting the syntax elements the distribution of components will change which results in a uniform probability distribution. This will increase the bit rate encoding (lowering the encoding efficiency). The number of encrypted elements and selected syntax elements directly affect the bit rate change. In Table 5-9, the bit rate change is reported for different video bitstreams and is compared with other important related works. The results show that the bit rate increase remains around the same value as was reported for other works in H.265/HEVC. However, the presented scheme for the H.266/VVC results in more visual deterioration.

Table 5-9 Bit rate change of our method compared to other works.

Video Sequence	Bit Rate Increase		
	Boyadjis (for HEVC) [9]	State of the art (for HEVC) [40] – Enc	Presented Scheme (for VVC)
Mobile	0.0177	0.0244	0.0143
BasketballPass	0.0263	0.0430	0.0266
BQMall	0.0216	0.0320	0.0204
Johnny	0.0200	0.0345	0.0204
FourPeople	0.0246	0.0348	0.022
BasketballDrive	0.0119	0.0290	0.0179
PeopleOnStreet	0.0151	0.0292	0.0313
Traffic	0.0164	0.0249	0.0174

5.3.3 Encryption Space

The available encryption space is an important factor in selective encryption algorithms since it determines the number of encrypted syntax elements. Encryption space should be large enough to make the brute force process difficult for an adversary who wants to find the keys used for the encryption. Through a brute force attack every possible values for each syntax element can be analyzed so that most of regions in the reconstructed image are recovered. The encryption space is also important since it is showing the components of the bitstream that we can encrypt without violating the format compliance rule. Table 5-10 shows the encryption space used in our method compared with two other methods proposed for HEVC. It can be seen that depending on the video sequence and assigned QP value, the number of syntax elements in the presented method is 10 to 20 times larger compared to selective encryption methods in previous standards. This is probably one of the reasons that selective encryption in H.266/VCC provides higher visual security (as we discussed earlier). Moreover, from Table 5-10, we notice a sharp change in encryption space as QP values change. The higher QP value results in higher compression and lower output quality, which causes to have fewer syntax elements to be encrypted (as more coefficients become zero).

5.4 Summary

a selective encryption method for videos encoded with H.266/VVC standard is presented. Key syntax elements of the encoder, such as MVD values and signs, IPM modes, and residual signs that carry important information, are selected for the encryption process. Then, the integrating of the presented selective encryption algorithm to the encoder is discussed both in terms of encoder performance (i.e., bitrate change), and visual security of the output video (i.e., SSIM, PSNR, VMAF). The results show that such methods can be a suitable solution while maintaining the privacy of videos for most of the applications. The importance of selective encryption algorithms is even more noticeable when dealing with videos that require large bandwidth like 4K/8K and 360-degree videos.

Table 5-10 Comparison of encryption space for one I and three B frames.

Video Sequence	QP	Selective Encryption Algorithms		
		Boyadjis (for HEVC) [9]	State of the art (for HEVC) [40] – Enc	Presented Scheme (for VVC)
Mobile	8	408280	420276	2970442
	24	91249	98087	1075159
	40	15129	16234	285866
BasketballPass	8	156381	162952	1747539
	24	25586	27982	525422
	40	2694	3207	69143
BQMall	8	1335708	1369545	12605551
	24	123706	133542	1887324
	40	15913	17993	341932
Johnny	8	1529922	1576207	16440069
	24	67579	72493	1502727
	40	9478	10669	188802
FourPeople	8	1581943	1639071	15812535
	24	109661	115938	1947921
	40	19176	20552	371282
BasketballDrive	8	5747621	5842386	74431080
	24	248473	268004	6289156
	40	25455	30993	428516
PeopleOnStreet	8	1.05×10^7	1.08×10^7	111624820
	24	1144257	1276675	19361562
	40	136658	169194	2799887
Traffic	8	9217222	9523708	100424218
	24	782504	844394	14478533
	40	103101	110918	1957091

Chapter 6

Conclusions and Suggestions for Future Works

6.1 Conclusion

Considering the growth of IoT, the importance of security and privacy issues is becoming more and more crucial. As the IoT market is relatively new, such issues have not been addressed so far. Moreover, because there are many companies and manufacturers involved in designing and producing IoT devices, the high priority concern so far was to release cheaper and more competitive products. Thus, many security issues are still not considered carefully. In this thesis, some of the security issues in the IoT environments were investigated, such as the need for lightweight cryptography algorithms and special considerations for encryption of video data. On one hand, since IoT devices are mostly designed for efficiency and their competitive price, security and reliability issues have not been the top priority. Moreover, because of the lower computation power and resource constraints in IoT devices, previously standardized cryptography algorithms such as Advanced Encryption Standard (AES) and RSA might not be a good choice for such devices. Also, not to mention that energy consumption is also another important factor since many IoT devices work with batteries. Thus in such environments, there is a need for cryptography algorithms that require smaller resources for the operation, which are called Lightweight Cryptography (LWC) algorithms. In Chapter 3, we compare 32 LWC algorithms and talk about their performance compared to previous common cryptography algorithms. Among the 35 chosen algorithms, 32 were selected from the second round of lightweight cryptography standardization process of the National Institute of Standards and Technology (NIST). The three other algorithms are AES with CCM, GCM, and OCB modes of operation. To compare the performance of these algorithms, their RAM requirement, CPU usage, and execution time are measured using three different IoT devices. The selected IoT nodes are Raspberry Pi 3, Raspberry Pi Zero W, and iMX233.

The second part of the thesis is mostly discussing video data compression and encryption. In Chapter 4, an overview of hybrid coding compression algorithms is provided. Also, this chapter talks about some of the new features and new changes in the H.266/VVC compared to previous compression standards. Currently, around 80% of IP traffic is video data. With the future applications of security cameras, smart homes, and AR/VR streaming, new schemes and considerations are necessary for the storage, compression, security, and analysis of video data. The necessity of UHD videos for AR/VR streaming and smart home/city applications are becoming more critical, and one of the available solutions is to employ video compression algorithms with high compression efficiency such as VVC. However, it comes with higher computation costs. With the future video technologies in smart cities and the employment of high-resolution videos, there is a need for works to address the security of videos. In Chapter 5, a selective encryption method for videos encoded with H.266/VVC standard is presented. Key syntax elements of the encoder, such as MVD values and signs, IPM modes, and residual signs that carry important information, are selected for the encryption process. Then, the integrating of the presented selective encryption algorithm to the encoder is discussed both in terms of encoder performance (i.e., bitrate change), and visual security of the output video (i.e., SSIM, PSNR, VMAF). The results show that such methods can be a suitable solution while maintaining the privacy of videos for most of the applications. The importance of selective encryption algorithms is even more noticeable when dealing with videos that require large bandwidth like 4K/8K and 360-degree videos.

6.2 Future Works

The presented scheme has several advantages although for improvement of this work and better implementations, there are following suggestions:

1- Experiments about the applicability of other syntax elements in the H.266/VVC for selective encryption.
2- Search for more efficient methods where encryption of syntax elements is only applied to a portion of CU units to reduce the computation cost while yielding the highest visual security.

3- Develop more dynamic methods where the encryption level is adjusted based on a particular application and information within frames.
4- More efficient integration of encryption algorithms into the compression algorithm, where if possible, the whole bit stream of considered syntax elements are encrypted at once. Currently, the encryption is applied repeatedly for small variables.

References

[1] C. W. Chen, “Internet of Video Things: Next-Generation IoT with Visual Sensors,” *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6676–6685, Aug. 2020, doi: 10.1109/JIOT.2020.3005727.

[2] E. Bertino and N. Islam, “Botnets and Internet of Things Security,” *Computer (Long. Beach. Calif).*, vol. 50, no. 2, pp. 76–79, Feb. 2017, doi: 10.1109/MC.2017.62.

[3] A. Fotovvat, G. M. E. Rahman, S. S. Vedaei, and K. A. Wahid, “Comparative Performance Analysis of Lightweight Cryptography Algorithms for IoT Sensor Nodes,” *IEEE Internet Things J.*, 2020, doi: 10.1109/JIOT.2020.3044526.

[4] M. binti Mohamad Noor and W. H. Hassan, “Current research on Internet of Things (IoT) security: A survey,” *Comput. Networks*, vol. 148, pp. 283–294, Jan. 2019, doi: 10.1016/j.comnet.2018.11.025.

[5] A. Mohan, K. Gauen, Y. H. Lu, W. W. Li, and X. Chen, “Internet of video things in 2030: A world with many cameras,” Sep. 2017, doi: 10.1109/ISCAS.2017.8050296.

[6] “2020 Global Networking Trends Report,” 2020. Accessed: Nov. 03, 2020. [Online]. Available: https://www.cisco.com/c/m/en_us/solutions/enterprise-networks/networking-report.html.

[7] Cisco, “Cisco Annual Internet Report (2018–2023) White Paper,” 2020. Accessed: Nov. 03, 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

[8] A. I. Sallam, O. S. Faragallah, and E. S. M. El-Rabaie, “HEVC Selective Encryption Using RC6 Block Cipher Technique,” *IEEE Trans. Multimed.*, vol. 20, no. 7, pp. 1636–1644, Jul. 2018, doi: 10.1109/TMM.2017.2777470.

[9] B. Boyadjis, C. Bergeron, B. Pesquet-Popescu, and F. Dufaux, “Extended Selective

Encryption of H.264/AVC (CABAC)-and HEVC-Encoded Video Streams," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 4, pp. 892–906, Apr. 2017, doi: 10.1109/TCSVT.2015.2511879.

[10] Z. Shahid and W. Puech, "Visual protection of HEVC video by selective encryption of CABAC binstrings," *IEEE Trans. Multimed.*, vol. 16, no. 1, pp. 24–36, Jan. 2014, doi: 10.1109/TMM.2013.2281029.

[11] A. Wieckowski *et al.*, "Towards A Live Software Decoder Implementation For The Upcoming Versatile Video Coding (VVC) Codec," Sep. 2020, pp. 3124–3128, doi: 10.1109/icip40778.2020.9191199.

[12] H. Liu, P. Ren, S. Jain, M. Murad, M. Gruteser, and F. Bai, "FusionEye: Perception Sharing for Connected Vehicles and its Bandwidth-Accuracy Trade-offs," in *Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks workshops*, Jun. 2019, vol. 2019-June, doi: 10.1109/SAHCN.2019.8824839.

[13] H. Xu, W. Yu, D. Griffith, and N. Golmie, "A Survey on Industrial Internet of Things: A Cyber-Physical Systems Perspective," *IEEE Access*, vol. 6. Institute of Electrical and Electronics Engineers Inc., pp. 78238–78259, 2018, doi: 10.1109/ACCESS.2018.2884906.

[14] T. Hussain, K. Muhammad, J. Del Ser, S. W. Baik, and V. H. C. De Albuquerque, "Intelligent Embedded Vision for Summarization of Multiview Videos in IIoT," *IEEE Trans. Ind. Informatics*, vol. 16, no. 4, pp. 2592–2602, Apr. 2020, doi: 10.1109/TII.2019.2937905.

[15] A. Nauman, Y. A. Qadri, M. Amjad, Y. Bin Zikria, M. K. Afzal, and S. W. Kim, "Multimedia internet of things: A comprehensive survey," *IEEE Access*, vol. 8, pp. 8202–8250, 2020, doi: 10.1109/ACCESS.2020.2964280.

[16] H. El-Sayed *et al.*, "Edge of Things: The Big Picture on the Integration of Edge, IoT and the Cloud in a Distributed Computing Environment," *IEEE Access*, vol. 6, pp. 1706–1717, Dec. 2017, doi: 10.1109/ACCESS.2017.2780087.

[17] L. Duan, J. Liu, W. Yang, T. Huang, and W. Gao, “Video Coding for Machines: A Paradigm of Collaborative Compression and Intelligent Analytics,” *IEEE Trans. Image Process.*, vol. 29, pp. 8680–8695, 2020, doi: 10.1109/TIP.2020.3016485.

[18] L. Yan *et al.*, “A 680 nA ECG acquisition IC for leadless pacemaker applications,” *IEEE Trans. Biomed. Circuits Syst.*, vol. 8, no. 6, pp. 779–786, Dec. 2014, doi: 10.1109/TBCAS.2014.2377073.

[19] M. N. Khan, A. Rao, and S. Camtepe, “Lightweight Cryptographic Protocols for IoT Constrained Devices: A Survey,” *IEEE Internet Things J.*, pp. 1–1, Sep. 2020, doi: 10.1109/jiot.2020.3026493.

[20] N. A. Gunathilake, W. J. Buchanan, and R. Asif, “Next Generation Lightweight Cryptography for Smart IoT Devices: : Implementation, Challenges and Applications,” in *IEEE 5th World Forum on Internet of Things, WF-IoT 2019 - Conference Proceedings*, Apr. 2019, pp. 707–710, doi: 10.1109/WF-IoT.2019.8767250.

[21] K. A. McKay, L. Bassham, M. S. Turan, and N. Mouha, “Report on lightweight cryptography,” Gaithersburg, MD, Mar. 2017. doi: 10.6028/NIST.IR.8114.

[22] “Buy a Raspberry Pi 3 Model B+ – Raspberry Pi.” https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/ (accessed Mar. 02, 2021).

[23] “Buy a Raspberry Pi Zero W – Raspberry Pi.” https://www.raspberrypi.org/products/raspberry-pi-zero-w/ (accessed Mar. 02, 2021).

[24] “iMX233-OLinuXino-MICRO - Open Source Hardware Board.” https://www.olimex.com/Products/OLinuXino/iMX233/iMX233-OLinuXino-MICRO/open-source-hardware (accessed Mar. 02, 2021).

[25] M. Dworkin, “Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC,” 2007. doi: 10.6028/NIST.SP.800-38d.

[26] P. Rogaway, M. Bellare, and J. Black, “OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption,” *ACM Trans. Inf. Syst. Secur.*, vol. 6, no. 3, pp. 365–

403, Aug. 2003, doi: 10.1145/937527.937529.

[27] M. Dworkin, “Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality.” doi: 10.6028/NIST.SP.800-38c.

[28] “jvet / VVCSoftware_VTM · GitLab.” https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM (accessed Nov. 09, 2020).

[29] “Fraunhofer Versatile Video Encoder (VVenC) – Fraunhofer Heinrich Hertz Institute.” https://www.hhi.fraunhofer.de/en/departments/vca/technologies-and-solutions/h266-vvc/fraunhofer-versatile-video-encoder-vvenc.html (accessed Nov. 09, 2020).

[30] T. Laude, Y. G. Adhisantoso, J. Voges, M. Munderloh, and J. Ostermann, “A Comprehensive Video Codec Comparison,” *APSIPA Trans. Signal Inf. Process.*, vol. 8, 2019, doi: 10.1017/atsip.2019.23.

[31] J. Chen, Y. Ye, and S. Kim, “JVET-T2002 Algorithm description for Versatile Video Coding and Test Model 11 (VTM 11),” 2021. [Online]. Available: https://jvet-experts.org/doc_end_user/current_document.php?id=10541.

[32] J. Meyer and F. Gadegast, “Security mechanisms for Mulimedia-Data with the Example MPEG-I-Video.” Accessed: Jan. 24, 2021. [Online]. Available: http://www.gadegast.de/frank/doc/secmeng.pdf.

[33] K. Muhammad, R. Hamza, J. Ahmad, J. Lloret, H. Wang, and S. W. Baik, “Secure surveillance framework for IoT systems using probabilistic image encryption,” *IEEE Trans. Ind. Informatics*, vol. 14, no. 8, pp. 3679–3689, Aug. 2018, doi: 10.1109/TII.2018.2791944.

[34] M. Preishuber, T. Hutter, S. Katzenbeisser, and A. Uhl, “Depreciating motivation and empirical security analysis of chaos-based image and video encryption,” *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 9, pp. 2137–2150, Sep. 2018, doi: 10.1109/TIFS.2018.2812080.

[35] S. K. A. Yeung, S. Zhu, and B. Zeng, “Design of new unitary transforms for perceptual video encryption,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 9, pp. 1341–

1345, Sep. 2011, doi: 10.1109/TCSVT.2011.2125630.

[36] S. Lian, Z. Liu, Z. Ren, and Z. Wang, “Selective video encryption based on advanced video coding,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Nov. 2005, vol. 3768 LNCS, pp. 281–290, doi: 10.1007/11582267_25.

[37] G. Van Wallendael, A. Boho, J. De Cock, A. Munteanu, and R. Van De Walle, “Encryption for high efficiency video coding with video adaptation capabilities,” *IEEE Trans. Consum. Electron.*, vol. 59, no. 3, pp. 634–642, 2013, doi: 10.1109/TCE.2013.6626250.

[38] K. Thiyagarajan, R. Lu, K. El-Sankary, and H. Zhu, “Energy-Aware Encryption for Securing Video Transmission in Internet of Multimedia Things,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 3, pp. 610–624, Mar. 2019, doi: 10.1109/TCSVT.2018.2808174.

[39] D. Xu, “Commutative Encryption and Data Hiding in HEVC Video Compression,” *IEEE Access*, vol. 7, pp. 66028–66041, 2019, doi: 10.1109/ACCESS.2019.2916484.

[40] F. Peng, X. Zhang, Z. X. Lin, and M. Long, “A Tunable Selective Encryption Scheme for H.265/HEVC Based on Chroma IPM and Coefficient Scrambling,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 8, pp. 2765–2780, Aug. 2020, doi: 10.1109/TCSVT.2019.2924910.

[41] Y.-K. W. Benjamin Bross, Jianle Chen, Shan Liu, “Versatile Video Coding text specification Draft 10,” 2020.

[42] V. Sze and D. Marpe, “Entropy Coding in HEVC,” 2014, pp. 5–6.

[43] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: From error visibility to structural similarity,” *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004, doi: 10.1109/TIP.2003.819861.

[44] R. Rassool, “VMAF reproducibility: Validating a perceptual practical video quality metric,” Jul. 2017, doi: 10.1109/BMSB.2017.7986143.

[45] “Netflix/vmaf: Perceptual video quality assessment based on multi-method fusion.” https://github.com/Netflix/vmaf (accessed Dec. 13, 2020).

[46] J. Chen, Y. Ye, and S. H. Kim, “Algorithm description for Versatile Video Coding and Test Model 5 - VTM 5,” 2019. [Online]. Available: http://phenix.it-sudparis.eu/jvet/doc_end_user/current_document.php?id=6641.

[47] M. Aagard, R. AlTawy, G. Gong, K. Mandal, and R. Rohit, “ACE: An Authenticated Encryption and Hash Algorithm,” Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available: https://csrc.nist.gov/events/2019/lightweight-cryptography-workshop-2019

[48] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer, “ASCON,” Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available: https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates

[49] S. Gueron, A. Jha, and M. Nandi, “COMET: COunter Mode Encryption with authentication Tag,” Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available: https://csrc.nist.gov /projects/lightweight-cryptography/round-2-candidates

[50] S. Riou, “DryGASCON,” Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available: https://csrc.nist.gov/projects /lightweight-cryptography/round-2-candidates

[51] T. Beyne, Y. L. Chen, and C. Dobraunig, “Elephant v1.1,” Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available: https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates

[52] A. Chakraborti, N. Datta, A. Jha, C. M. Lopez, M. Nandi, and Y. Sasaki, “ESTATE,” Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available: https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates

[53] E. Andreeva, V. Lallemand, A. Purnal, R. Reyhanitabar, A. Roy, and D. Vizár, “ForkAE v.1,” Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available:

https://csrc.nist.gov
/projects/lightweight-cryptography/round-2-candidates

[54] S. Banik, A. Chakraborti, T. Iwata, K. Minematsu, M. Mridul, T. Peyrin, Y. Sasaki, S. M. Sim, and Y. Todo, "GIFT-COFB v1.0," Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available: https://csrc.nist.gov/events/2019/lightweight-cryptography-workshop-2019

[55] D. J. Bernstein, S. Kölbl, S. Lucks, P. M. C. Massolino, F. Mendel, K. Nawaz, T. Schneider, P. Schwabe, F. Standaert, Y. Todo, and B. Viguier, "Gimili," Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available: https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates

[56] M. Hell, T. Johansson, W. Meier, J. Sönnerup, AND H. Yoshida, "Grain-128AEAD - A lightweight AEAD stream cipher," Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available: https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates

[57] A. Chakraborti, N. Datta, A. Jha, AND M. Nandi, "HyENA," Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available: https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates

[58] C. Dobraunig, M. Eichlseder, S. Mangard, F. Mendel, B. Mennink, R. Primas, and T. Unterluggauer, "ISAP v2.0," Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available: https:
//csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates

[59] W. Zhang, T. Ding, B. Yang, Z. Bao, Z. Xiang, F. Ji, and X. Zhao, "KNOT," Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available: https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates

[60] A. Chakraborti, N. Datta, A. Jha, C. M. Lopez, M. Nandi, and Y. Sasaki, "LOTUS-AEAD and LOCUS-AEAD," Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available: https://csrc.nist.gov/
projects/lightweight-cryptography/round-2-candidates

[61] B. Chakraborty and M. Nandi, “mixFeed,” Second Round Candidate of the NIST LWC Competition, 2019.

[62] B. Chakraborty and M. Nandi, “ORANGE,” Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available: https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates

[63] A. Bhattacharjee, E. List, C. M. López, and M. Nandi, “Oribatida v1.2,” Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available: https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates

[64] Z. Bao, A. Chakraborti, N. Datta, J. Guo, M. Nandi, T. Peyrin, and K. Yasuda, “PHOTON-Beetle,” Second Round Candidate of the NIST LWC Competition, 2019. [online]. Availale: Available: https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates

[65] D. Goudarzi, J. Jean, S. Kölbl, T. Peyrin, M. Rivain, Y. Sasaki, and S. M. Sim, “Pyjamask v1.0,” Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available: https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates

[66] T. Iwata, M. Khairallah, K. Minematsu, and T. Peyrin, “Romulus v1.2,” Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available: https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates

[67] Y. Naito, M. Matsui, Y. Sakai, D. Suzuki, K. Sakiyama, and T. Sugawara, “SAEAES,” Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available: https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates

[68] A. Canteaut, S. Duval, G. Leurent, M. Naya-Plasencia, L. Perrin, T. Pornin, and A. Schrottenloher, “Saturnin: a suite of lightweight symmetric algorithms for post-quantum security v1.1,” Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available: https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates

[69] C. Beierle, J. Jean, S. Kölbl, G. Leander, A. Moradi, T. Peyrin, Y. Sasaki, P. Sasdrich, and S. M. Sim, “SKINNY-AEAD and SKINNY-Hash v1.1,” Second Round Candidate of the

NIST LWC Competition, 2019. [online]. Available: https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates

[70] C. Beierle, A. Biryukov, L. C. Santos, J. Großschädl, L. Perrin, A. Udovenko, V. Velichkov, Q. Wang, "Schwaemm and Esch: Lightweight Authenticated Encryption and Hashing using the Sparkle Permutation Family," Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available: https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates

[71] R. AlTawy, G. Gong, M. He, K. Mandal, and R. Rohit, "Spix: An Authenticated Cipher," Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available: https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates

[72] R. AlTawy, G. Gong, M. He, A. Jha, K. Mandal, M. Nandi, R. Rohit, "SpoC: An Authenticated Cipher," Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available: https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates

[73] D. Bellizia, F. Berti, O. Bronchain, G. Cassiers, S. Duval, C. Guo, G. Leander, G. Leurent, I. Levi, C. Momin, O. Pereira, T. Peters, F. Standaert, and F. Wiemer, "Spook: Sponge-Based Leakage-Resistant Authenticated Encryption with a Masked Tweakable Block Cipher," Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available: https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates

[74] J. Daemen, P. Maat Costa Massolino, and Y. Rotella, "The Subterranean 2.0 cipher suite," Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available: https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates

[75] S. Banik, A. Bogdanov, T. Peyrin, Y. Sasaki, S. M. Sim, E. Tischhauser, and Y. Todo, "SUNDAE-GIFT v1.0," Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available: https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates

[76] H. Wu and T. Huang, "TinyJAMBU: A Family of Lightweight Authenticated Encryption AlgorithmsForkAE," Second Round Candidate of the NIST LWC Competition, 2019.

[online]. Available: https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates

[77] M. Aagaard, R. AlTawy, G. Gong, K. Mandal, R. Rohit, and N. Zidaric, "WAGE: An Authenticated Cipher," Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available: https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates

[78] J. Daemen, S. Hoffert, M. Peeters, G. V. Assche, and R. V. Keer, "a lightweight cryptographic scheme," Second Round Candidate of the NIST LWC Competition, 2019. [online]. Available: https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates