Security integration in IP video surveillance systems

NATALIA PARATSIKIDOU

KTH Information and
Communication Technology

Degree project in
Communication Systems
Second level, 30.0 HEC
Stockholm, Sweden

Security integration in IP video surveillance systems

Master thesis

Natalia Paratsikidou
natpar@kth.se

Examiner & Academic Supervisor:

Prof. Gerald Q. Maguire Jr.

Kungliga Tekniska Högskolan

Stockholm, Sweden

Industrial Supervisor:

Henrik Mau, Software Projects Manager

Veracity UK Ltd (henrik.mau@veracityuk.com)

Prestwick, UK

School of Information and Communication Technology (ICT)

KTH Royal Institute of Technology

Stockholm, Sweden

Abstract

Video surveillance systems are a rapidly growing industry. As with most systems, this technology presents both opportunities and threats. The wide adoption of video surveillance systems by various businesses and individuals has raised some vital security issues. Appropriately addressing these security issues is of great importance for video surveillance systems, as these systems may capture sensitive personal data and may attract numerous attacks. As of today nearly all devices have become networked (or are on their way to being connected to networks), hence eavesdropping is a common attack which can exploit a breach of a system's security and result in data disclosure to unauthorised parties, video stream alterations, interference, and reduction of a system's performance. Moreover, it is important that video surveillance systems are standardized by appropriate standardization organizations in order to assure high quality of the security services that utilize these systems and to facilitate interoperability.

In this master thesis project rules and regulations concerning personal data protection were studied in order to define the requirements of the proposed robust and high quality security scheme that is to be integrated into video surveillance systems. This security scheme provides United States Federal Information (FIPS)[*] compliant security services by securing the communication channel between the system's devices. The authentication of the system's devices is established by using certificates and key exchanges. The proposed security scheme has been scrutinized in order to analyze its performance (and efficiency) in terms of overhead, increased jitter, and one-way delay variations.

Our implementation of the proposed security scheme utilized OpenVPN to provide privacy, integrity and authentication to the video streaming captured by Veracity's clients and stored in Veracity's proprietary NAS device (COLDSTORE). Utilization of OpenSSL FIPS Object module develops our security scheme in a FIPS compliant solution. For testing purposes, we created different test scenarios and collected data about the total delivery time of a video file, delivered from the IPCamera/NVR/DVR devices to the COLDSTORE device, the network overhead and lastly the one-way delay between the two endpoints.

Another area of interest that we focus on is how to deploy certificates to new, existing, and replacement devices; and how this deployment may affect the system's security design. In addition, we investigate the problems arising when a secured video stream needs to be played back via another device outside of our system's network.The results of the thesis will be used as an input for product development activities by the company that hosted this thesis project.

Keywords: Video surveillance Systems, security, FIPS, OpenVPN

[*] This master thesis also considers and describes other security specifications such as British Standard Institution (BSI) 10012, and International Standards Organization (ISO) and International Electrotechnical Commission (IEC) standard (ISO/IEC 27001) to provide a complete picture of the rules and regulations concerning personal data protection.

Sammanfattning

Videoövervakningssystem är en växande industri. Precis som med de flesta systemen, har denna teknologi både möjligheter och risker. Den stora utspridningen av videoövervarkningssystemen har lett till essentiella säkerhetsrisker. Det ligger en stor vikt i att hantera säkerhetsrisker för videoövervakningssystem i och med att dessa system kan eventuellt fånga upp personlig data och därav attrahera attacker. Idag har nästan alla enheter blivit nätverksanslutna (eller är påväg att bli), vilket har lett till att avlyssning har blivit en vanlig attack. En avlyssnare kan exploatera en säkerhetsrisk och resultera i informationsläckor till obehöriga, videomanipulering, störningar, och reducerad prestanda i systemet. Det viktigt att videoövervakningssystem är standardiserade av lämpliga standardiseringsorganisationer för att säkra en hög kvalité i säkerhetstjänsterna som använder sig av dessa system och för att försäkra sig om kompatibilitet.

I den här examensarbetet studerade man regler och förordningar som har att göra med säkrandet av personlig data, för att kunna definiera kraven för det föreslagna robusta och högkvalitativa säkerhetsarkitekturen som skall integreras med videoövervakningssystemen. Säkerhetsarkitekturen erbjuder United States Federal Information (FIPS)* kompatibla säkerhetstjänster genom att säkra kommunikationskanalen mellan systemets enheter. Autentiseringen av systemets enheter sker genom att använda certifikat och nyckelutbyten. Det föreslagna säkerhetsarkitekturen har granskats för att analysera dess prestanda vad gäller ineffektiviteter, ökade störningar och fördröjningar i envägs variationer.

Vår genomförandet av den föreslagna systemet utnyttjas OpenVPN att tillhandahålla sekretess, integritet och autentisering till strömmande video fångades av Veracity kunder och lagras i Veracity egenutvecklade NAS-enhet (COLDSTORE). Utnyttjande av OpenSSL FIPS Objekt modulen utvecklar vår trygghet i ett FIPS-kompatibel lösning. För teständamål, skapade vi olika testscenarier och insamlade data om den totala leveranstiden för en videofil, som levereras från IPCamera / NVR / DVR-enheter till fryshus enhet, nätverket overhead och slutligen den enkelriktad fördröjning mellan de två ändpunkterna .

Ett annat område av intresse som vi fokuserar på är certifikat för nya, existerande och ersättningsenheter; och hur det kan påverka systemets säkerhetsarkitektur. Utöver detta undersöker vi problemen som uppstår när en säkrad videoström behöver spelas upp i en enhet utanför systemets nätverk. Insatsen gjord i det här examensarbetet kommer användas som grund för produktutvecklingen av företaget där examensarbetet gjordes.

Nyckelord: Videoövervakningssystem, säkerhet, FIPS, OpenVPN

* Examensarbetet tar också hänsyn till och beskriver andra säkerhetspecifikationer som British Standard Institution (BSI) 10012, och International Standards Organization (ISO) och International Electrotechnical Commission (IEC) standard (ISO/IEC 27001) för att ge en fullständig bild av regler och förordningar angående säkrandet av personlig data.

Acknowledgements

I would like to thank all the people that helped me to conduct this master thesis project. First and foremost, I would like to express my sincere gratitude to my academic supervisor professor Gerald Q. “Chip” Maguire Jr. for his consistent help and valuable feedback that without which this thesis project could not be completed. I would also like to thank my industrial supervisor Henrik Mau and his company Veracity that gave me the opportunity to investigate on the subject using real problem parameters. Last but not least, I would like to thank my family and friends for always giving me great support and encouragement to achieve my goals.

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

AES	Advanced Encryption Standard
BF	Blowfish
BSI	British Standard Institution
CA	Certificate Authority
CCTV	Closed-Circuit Television
CMVP	Cryptographic Module Validation Program
CODEC	Coder/decoder
CRL	Certificate Revocation List
CSE	Communication Security Establishment
CSR	Certificate Signing Request
CTR mode	Counter mode
DES	Digital Encryption Standard
DH	Diffie-Helman
DPA	Data Protection Act
DSA	Digital Signature Algorithm
DVR	Digital Video Recorder
EU	European Union
FIPS	Federal Information Processing Standards
ICO	Information Commissioner's Office
IDEA	International Data Encryption Algorithm
IP	Internet Protocol
IPsec	Internet Protocol Security
ISMS	Information Security Management System
LAID	Linear Array of Idle Disks
L2TP	Layer 2 Tunneling Protocol
NAS	Network-Attached Storage
NIST	National Institute of Standards and Technology
NTP	Network Time Protocol
NVR	Network Video Recorder
OS	Operating System
OSF	OpenSSL Software Foundation
PKI	Public Key Infrastructure
PoE	Power over Ethernet

PPP	Point-to-Point Protocol
PPTP	Point-to-Point Tunneling Protocol
PUB	Publication
RBG	Random Bit Generator
RTT	Round-trip time
RVEA	Real-time Video Encryption Algorithm
SFS	Sequential disk Filing System
SSL	Secure Socket Layer
S/MIME	Secure/Multipurpose Internet Mail Extensions
TLS	Transport Layer Security
UK	United Kingdom
US	United State
VBR	Variable Bitrate
VCR	Videocassette Recorder
VNI	Virtual Network Interface
VPN	Virtual Private Network

Chapter 1

Introduction

This chapter presents a brief introduction to the thesis project and its corresponding research area by stating the problem that this thesis addresses under the general context of securing IP video surveillance systems. Specifically, section 1.1 describes the general framework under which video surveillance systems operate and the security aspects that such infrastructures must address. Section 1.2 introduces the problems that exist in the area of video surveillance which this master's thesis project tries to find solutions to. Finally, section 1.3 presents the aims and objectives of this thesis project.

1.1 Overview

Our society has changed from an industrial to an information society where creation, collection, distribution, exploitation, and manipulation of information have become essential parts of our socioeconomic and cultural activities[1]. Digital information is now generated in various formats, and the internet protocol has become the predominant media for information exchange. In this new information society most businesses are becoming networked, by adopting simple and cost-effective IP based solutions in order to give users access to view and manipulate information collected by networked devices.

Security has always been an important business sector. For more than 35 years video surveillance systems, for example closed-circuit television (CCTV), have been used to protect critical infrastructures[2]. As a result of the currently emerging IP based systems, video surveillance systems are becoming fully integrated solutions by adapting new digital technologies. Reaping the benefits of the available high speed Internet infrastructures, video surveillance systems are upgrading from low resolution wired analogue cameras to high resolution wireless IP cameras and from slow, insecure, magnetic tapes (such as VHS) storing video streams to new, safer, and faster magnetic or optical disk storage solutions.

Most video surveillance systems are composed of three standard modules: video capture devices, digital video recorders (DVRs) or network video recorders (NVRs) accessing multiple cameras which temporarily store video streams, and a remote monitoring system[3]. Although DVRs and NVRs have become more and more powerful over the decades, they still have some inherent limitations based on their small storage capabilities and the computation burden, as they have to record, compress, encrypt, and authenticate the streams collected from all of the cameras in a video surveillance system, and then transfer the data through the network to a remote monitoring system[2]. Therefore, today most businesses with high video surveillance demands are using DVRs/NVRs to temporarily store the data, but then utilize a network-attached storage (NAS) device which is more reliable, simpler, and inexpensive to store the data for a longer period of time. Modern decentralized IP capable cameras can

compress, encrypt and/or authenticate, and temporarily store video streams to a network attached hard disk eliminating the need for DVRs/NVRs.

The data streams that we will be concerned with are comprised of two types of data: control data and media data. An IP video surveillance network is responsible for delivering both the control and the media data as quickly as possible so that the media data can be displayed in (near) real-time on remote monitoring devices. For example, if a video surveillance system is established in a mall, then the security guard who is monitoring the mall from a control room would like to quickly know if a customer is hurt or a shop is being robbed in order to act as quickly as possible, for example calling for an ambulance or calling the police, and thus minimizing the negative effects of the incident. If the network delivers the video stream with 5 minutes of delay, then an injured customer might not be able to be helped by the ambulance or the robber may have escaped; thus such a surveillance system would be considered unsuitable.

Control data are small packets generally transmitted in bursts (as control data are sent only on start-up, shut down, or if a change occurs in the communication between the parties). This control information may be highly correlated with other context data. The control data are important and may carry confidential or configuration data. Control data utilize a very small percentage of the available network bandwidth, called the control bandwidth. In contrast the media data conveys an immense amount of information. For example, a standard definition TV stream using MPEG2 encoding generates 1.8Mbps and MPEG4 generates approximately 0.9Mbps [4]. The media data are generally time sensitive as described in the above example. As a result, streaming data occupy the bulk of the available network bandwidth used by a video surveillance system.

Considering the nature of video surveillance systems, it is apparent that security issues arise when we examine the threats to the system. Video images are captured by surveillance cameras and stored in different devices, where most of the time this data falls under the category of sensitive personal data. All countries in the European Union (EU) are obliged to comply with the *Data Protection Directive* [5] which regulates the processing and possession of personal data within the EU borders. Competitors or malicious parties might capture the video (and possibly audio) by listening to the network's traffic. In addition to passive listening, they might also tamper with the data or generate fake streams as if the streams came from cameras, for example to trick security guards about what is happening at that moment in a specific area. Therefore, integration of security is a necessity for the efficiency and effectiveness of video surveillance systems.

Security in network systems is preserved by authenticating the devices transferring data to the network, ensuring that the transferred data are confidential and cannot be revealed to unauthorized parties, that data remain unmodified, and lastly proving that expected sender has really sent the transferred data (i.e., authenticating the source). The aforementioned criteria are summarized in the following security principles: authentication, confidentiality, integrity, and non-repudiation. Any video surveillance system should provide this security in order to be practical and usable. In this thesis project, we begin by examining the EU data

protection laws and later we implement a security scheme to comply with these laws. We also investigate and analyze the performance of the proposed security scheme in terms of added overhead, one-way delay, and jitter; and finally discuss various problems regarding certificate deployment and we propose some solutions taking into consideration the available resources.

1.2 Problem Statement

Security vulnerabilities can render a video surveillance system unreliable, hence it will not be useful to its customers and ultimately neither the industry nor the society would profit from it. Moreover, EU laws are strict concerning how data can be processed and stored by organizations, hence the appropriate security must be provided to prevent disclosure of data to unauthorized parties. Security mechanisms provide authentication, confidentiality, integrity, and non-repudiation of information by applying cryptographic algorithms and secrets to streaming data to secure the data against malicious actions. However, all of these techniques are computationally expensive and might utilize greater bandwidth (i.e., add overhead to the data that must already be transferred and stored). More specifically, adding extra computations on the media data stream increases the computational overhead impeding transmission if the hardware is not adequate (i.e., due to increased delay). Although the bandwidth consumed by security headers and control data might be negligible for most applications, video data already consumes a lot of the network bandwidth and the security overhead will add an additional burden due to the extra traffic which must be sent via the network.

Part of the encryption and/or authentication process can be facilitated by deploying a certificate to the system's devices, if public key encryption is implemented. These public key certificates can be signed by a third party or by the system operator or vendor. The certificates must be generated by a trusted party and must be delegated to (and deployed in) all the network attached devices. Another area of interest concerning encryption and/or authentication is playback on various different devices outside the organization's networked system, for example to provide playback on a device in a police station. Such a remote playback outside of the surveillance system poses a number of problems, due to the restrictions that must be met, hence making the meeting of the security requirements more difficult for the security designers.

This thesis project begins by examining the EU laws for data protection and will propose a suitable security scheme which meets these laws and is both performance and bandwidth efficient for IP video surveillance networks. Moreover, an investigation will be made of certificate deployment and its impact on device management, for example the introduction of new devices, device replacement, etc., and video streaming playback via different devices.

1.3 Aims, goals, and objectives

This master's thesis project is being conducted in cooperation with Veracity, a United Kingdom (UK) company which provides IP solutions in three main areas: transmission, storage, and display. The company provides digital video surveillance solutions and currently intends to become compliant with the European laws and United States Federal Information Processing Standards (FIPS) rules (more information about UK and U.S. FIPS standards are given in section 2.3). To achieve this goal a secure channel *must* be implemented between the different devices in the video surveillance system. This channel introduces encryption and authentication in the company's network protocol in order to secure their customer's generated video stream and video data storage. The company also utilizes their own proprietary NAS system, called COLDSTORE. COLDSTORE, unlike common DVRs and RAID disks arrays which can be unreliable due to a high rate of disk failures, provides an innovative solution for streaming storage, using Linear Array of Idle Disks (L.A.I.D™) technology combined with a unique Sequential disk Filing System (SFS™), which minimizes disk failure, power costs, and usage complexity. Along with COLDSTORE, the company also markets a disk player system, called DISKPLAY, which allows direct playback of an extracted COLDSTORE disk on any PC.

This thesis project's objectives are based on the company's current needs to investigate security integration in their clients' networks. The objects are:

a) Investigation of EU laws concerning protection of personal data;

b) Propose a security scheme compliant with the investigated EU directives;

c) Examine performance, throughput, and latency impacts introduced by the encryption and/or authentication integration on the company's video surveillance testing system;

d) Investigate certificate deployment to new, existing, and replacement devices and its effect on the design of the security scheme; and

e) Perform research leading to proposed solutions that facilitate video playback via devices other than those within the organization's own surveillance network.

The thesis deliverables consists of:

a) Documentation concerning the UK standards that a video surveillance system must comply with;

b) A security scheme compliant with these UK standards based upon implementing a secure channel using OpenVPN for protecting the communication between the video surveillance system's devices;

c) Documentation about the performance of the video surveillance system after security integration, which includes the following metrics:

    a. Overhead;

    b. Jitter; and

    c. End-to-end delay.

d) Documentation about efficient certificate deployment; and

e) Documentation concerning playback via different devices (especially those outside of the organization's network).

1.4 Thesis outline

The thesis is organized into 6 chapters. Chapters 1 and 2 give a brief introduction into the video surveillance systems field of study, some background information about the evolution of these systems from 1973 to now, and also cover the legal framework regarding the processing and use of personal data not only in Europe but also internationally.

To continue, chapter 3 will briefly introduce the existing security schemes and how they are used for protection of the video stream captured from video surveillance systems. A brief comparison is also included to help the reader discern the advantages and limitations of each scheme and understand the reasons behind the selection of the proposed scheme.

The methodology that was used to conclude in the proposed security scheme and the selection of the scenarios for testing this scheme, as well as the tools used for data capture and analysis are included in chapter 4.

In chapter 5, useful information about the analysed data are presented and various graphs are included to interpret the results. Finally, some conclusions are drawn and a discussion on the future work is included.

Chapter 2

Background

Video surveillance systems are widely accepted by the public in many countries and modern IP networks have revolutionized surveillance networks over the past decades. However, there are a lot of objections to the use of video surveillance systems due to concerns about privacy and disclosure of personal data to unauthorized parties. As a result, legislation concerning protection of personal data has been introduced in many countries and video privacy techniques must be implemented to comply with these laws. This chapter provides the background knowledge required for grasping the concepts that are utilized in this master's thesis project. The chapter begins, in section 2.1, by reviewing the history of video surveillance systems and how they have evolved during the past decades. Section 2.2 describes the existing rules and regulations for personal data protection, while section 2.3 describes three important standards for information management systems.

2.1 Evolution of Video Surveillance Systems

The growing demand for security from modern societies has led to an increasing demand for surveillance solutions in many diverse environments. Special attention is given to public places or crowded places, such a public transportation, airports, stadiums, and malls. This demand for better security systems has lead the video surveillance market to improve image quality, simplify use and maintenance, integrate security and reliability, provide greater storage capacities, introduce more cost efficient and scalable solutions, and make many other improvements. To meet the aforementioned requirements video surveillance systems have undergone a lot of technological changes and improvements during the past four decades, starting from analogue solutions to today's fully digital systems.

The first CCTV systems, which appeared in the beginning of 1970s, were completely analog, consisting of analogue cameras connected with separate coaxial cables to a videocassette recorder (VCR) device. Initially, these VCR devices utilized the same cassettes used for home VCR devices, and as the video was uncompressed, one tape lasted for approximately 8 hours. Eventually, in large systems, a multiplexer was used to combine multiple video signals from several cameras into a multiplexed video signal and this signal was sent to a VCR and/or monitor. An analogue monitor was used to display the captured video. Although analog systems operated well, they did not have good scalability and they were expensive in terms of installation and VCR maintenance costs.

By the middle of 1990s, VCRs were replaced by the newly introduced DVRs which digitized and compressed the analogue video, resulting in digital data being written to a hard disk drive. As the hard disk drives initially had limited storage and were too expensive, a number of companies introduced proprietary compression algorithms to reduce the amount of

data that needed to be stored. Over time these algorithms were replaced by standard compression algorithms, such as MPEG-4. Standardization led to cost reductions and increased performance (due to hardware implementation of the compression and decompression algorithms). DVRs offered several video inputs and thus replaced both the multiplexer and VCR devices, minimizing the number of system components. Lastly, due to digital video being available from the DVRs, this digitized data could be transmitted across longer distances, hence DVRs were equipped with embedded telephone modems to transfer video to remote monitoring locations. Unfortunately, these telephone lines had low bandwidth which meant low video frame rates, low resolution, and the necessity to use high compression, all of which significantly degraded the video quality.

Subsequently DVRs were equipped with an Ethernet port, making them network video devices (NVRs), which enabled them to be easily connected to faster networks. Consequently, NVRs became available on the market which provided remote monitoring capabilities using a remote PC. However, these NVRs had some inherent drawbacks; they had a high computation burden, since they had to perform compression, while recording and transferring multiple streams from different cameras, which rendered them slow when used in large configurations. Moreover, their maintenance was rigid and expensive as the hardware was proprietary due to preloaded software, forcing end users to stick with a given manufacturer's service. Scalability was also an issue, as most NVRs were built with 16 or 32 inputs making them difficult to use in systems that were not multiples of 16 cameras, for example when using 18 cameras one needed to add another NVR.

The first attempts to create a networked video system were due to the introduction of video encoders. A video encoder is a device which connects to analog cameras and then digitizes and compresses the video stream from these cameras. This stream can be sent through an IP network to a remote server. The server runs video management software, thus providing better scalability and better maintenance for the video surveillance system. Scalability was provided by either scaling up the performance of a single server or scaling out with the addition of more servers. Maintenance was improved through the use of commodity server and network technology. The introduction of NVRs and hybrid DVR devices including video management software prolonged the lifetime of DVR solutions, although the drawbacks of traditional DVRs were retained. [6]

Nowadays, analog cameras are being replaced with networked cameras, specifically IP cameras creating a fully networked video surveillance solution and eliminating the separate video encoders. IP cameras are connected through IP networks to network switches and routers, and then to servers where the digital streams are stored. Digitization occurs inside the cameras and the data remains in digital form throughout the network. Another benefit of network based solutions is that IP cameras can use the existing network infrastructure of a building or an area to transfer their video data. Also, surveillance capabilities can exploit the technology of the IP cameras, as some devices allow high resolution, zoom commands, etc. State-of-the-art network camera devices can also use Power over Ethernet (PoE) functionality to minimize the cost of providing power to the cameras and enabling cameras to be installed anywhere there is a network jack (connected to a PoE switch). Moreover, intelligent cameras

are the latest generation of networked cameras. These intelligent cameras make use of embedded motion detection, night vision, and alarm capabilities. Ultimately a network camera based solution is highly scalable and flexible, provides high video quality, and offers remote camera controls such as zoom, pan, tilt, etc. This preprocessing of video streams by cameras can increase infrastructure bandwidth, and reduce processing time and power consumption of the DVR/NVR devices, hence improving the system's accuracy and performance. [7]

2.2 Data Protection Rules and Regulations

Everyday vast amounts of personal data information are transferred through banking, business, public, or social networks all around the world. Different countries have their own (possibly conflicting) rules for protecting personal data. These rules impede the free movement of personal data internationally. The EU established the Data Protection Directive (officially known as Directive 95/46/EC) [5] to ensure that personal data are secured according to high standards in all the EU countries. This directive (and the national laws based upon it) also defines specific rules regarding the transfer of personal data outside the EU's borders.

The Data Protection Directive allows the processing and storage of personal data *only* if the following criteria are met [5]:

a) The data are processed if an explicit and legitimate purpose is specified;
b) The data subject has the right to be informed about their data that a party is holding, how the data are processed, and for how long the data will be kept; and
c) The data are accurately collected and are relevant in subject and amount based on the purpose for which the data are obtained and processed.

The EU directive also specifies that each member state has to set up an independent body having the responsibility to monitor that the rules and regulations are correctly followed and to start legal proceedings in the event of a suspected data protection violation. Sections 2.2.1 to 2.2.3 and section 2.3.1 describe the UK laws and regulations to support the reader's understanding of the legal framework under which Veracity and its clients operate in the UK.

In 2012, the EU released a proposal reforming the data protection directive to include important aspects such as strengthening personal rights, handling globalization, social networks, and cloud computing. These new proposals are planned to be adopted in 2014 in national laws and put into effect in 2016. Discussions about these proposed new laws are still on-going. [8]

2.2.1 Data Protection Act

The Data Protection Act (DPA) 1998 is an act of the UK Parliament defining the laws regarding processing and handling of individuals' information. DPA was enacted to comply

with the EU data protection directive of 1995 [5] requiring member states to protect individual's fundamental rights and freedoms and specifically the right of privacy regarding processing and storage of personal data. The act does **not** apply to domestic personal use, such as civilians keeping an address book, but anyone who is collecting and storing personal data for other purposes is legally obliged to conform to this Act[9].

2.2.2 Information Commissioner's Office

The official legal body responsible for monitoring and executing the EU directives in the UK is the Information Commissioner's Office (ICO). ICO is an independent authority responsible for ensuring that Act's regulations are enforced and provides guidance for conformity[10]. Specifically, ICO states that any organization collecting and/or processing personal data has to notify the ICO about these actions. This is due to a basic principle stated in DPA that the public should be aware of or be able to find out who is processing their personal data and for what reason[10].

2.2.3 UK personal data security enforcements

DPA does not require that personal data be encrypted, although it does require security measures in order to prevent personal data from disclosure to unauthorized parties, as mentioned in their guide to data protection[10]. Additionally, the Parliamentary Office of Science and Technology of the UK has published an article regarding CCTV systems, in which it clearly mentions that the authentication of CCTV images must be provided in order to avoid potential modification of recorded data[11]. Furthermore, a parliamentary report [12] referring to digital images states that a digital signature is an adequate way to authenticate data sent electronically. If data is transferred through the network or stored, then encryption protocols should be utilized.

2.3 EU and International standards

There is an abundance of EU, international standards, and corresponding organizations for ensuring information security and to help companies provide high quality personal data security to their clients. The following subsections describe three well known set of standards that are widely used in information management systems.

2.3.1 BS 10012

The British Standard Institution (BSI) is an international non-profit and government independent business service whose major responsibility is to produce standards and provide related services for them. In the UK, BSI is also responsible to provide British standards complying with international and EU standards. [13]

BS 10012 has been developed by BSI to help companies establish and maintain best practice personal information management systems complying with the DPA. This is the first standard that relates to the management of personal information. By following the framework set out within BS 10012, organizations can improve data storage protection and better management of data processing and data transfers, so that they comply with legislation. [14]

2.3.2 ISO/IEC 27001

The International Standards Organization (ISO) and International Electrotechnical Commission (IEC) standard, ISO/IEC 27001, is an international standard for information security management. The standard provides a framework which helps companies to build a concrete and secure Information Security Management System (ISMS) by providing the appropriate procedures and functions to secure organizations' information and minimize risks, litigation, and downtime. Adopting the international standard assures that a company is using a risk-based approach to gather and implement security controls. It also ensures that the company meets the relevant international and national laws and that the relevant regulations are identified and compliance is provided. [13]

2.3.3 FIPS

The United States (US) Federal Information Processing Standards (FIPS) publications (PUBs) are guidelines that set best practices for software and hardware computer security products and are developed by the US federal government. FIPS 140 is a series of standards specifying cryptographic modules and FIPS 140-2 is the current version of the standard. Although they are developed by the US federal government, they are widely used as standards to specify encryption and authentication requirements. [15]

FIPS PUB 140-2 was issued by the US National Institute of Standards and Technology (NIST). It has also been adopted by the Canadian government's Communication Security Establishment (CSE). The US and Canadian governments have established the Cryptographic Module Validation Program (CMVP) as a shared effort between the aforementioned bodies. FIPS PUB 140-2 specifies the security criteria that a cryptographic module must meet in order to be used by security systems protecting information inside IT systems. FIPS 140-2 consists of four increasing, qualitative levels of security which covers the diverse needs of potential applications and environments deploying cryptographic modules.

The basic reason for a company to acquire FIPS certification is to comply with international standards if it is selling products that implement cryptography to the US federal government. Moreover, FIPS nowadays is used as a quality mark and an indication that a product is certified by a prominent certification organization. The reason why we are interested in ensuring that the proposed security scheme is FIPS compliant is that a lot of Veracity's clients want to buy systems that have been validated according to the FIPS standards. This validation will ensure that Veracity's products and services meet the requirements standardized by a well-known and reputable organization, making these products and services competitive - as well as opening new markets for the company in the US.

Chapter 3

Security schemes for Video surveillance systems

Video surveillance systems handle sensitive personal information, therefore privacy of the images (and other data) that are captured is required. An important solution to ensure privacy in these systems has emerged through the development of virtual private networks (VPNs). VPNs can provide a secure communication channel between devices in a video surveillance system using encryption and authentication. Together with encryption of the stored data in a NAS device we can assure that the captured data are well protected inside our system. For this reason, section 3.1 describes the different available cryptographic approaches for securing video surveillance systems summarizing an existing comparison of their main characteristics. Section 3.2 presents the existing cryptography libraries and the reason we selected OpenSSL to support our cryptographic implementations. Finally, section 3.3 refers to VPN technologies and states the advantages of using the OpenVPN SSL/TLS application for building a secure tunnel between the system's devices.

3.1 Security Algorithms

The most prominent method for ensuring privacy for video surveillance systems is encryption. Encryption assures that the data are protected while being transferred and only authorized parties can access the stored data, as we assume that they are the only ones who have the correct key to decrypt it. A lot of research effort has been devoted to find the best algorithm or set of algorithms for encrypting video data. In the following sections we discuss two prevalent cryptographic algorithm approaches and we compare and contrast their effectiveness in encrypting video surveillance data.

3.1.1 Naive Algorithms

Conventional cryptographic methods include the common secret key, public key, and hash function algorithm types. Briefly, secret key cryptography uses one key for both encryption and decryption, public key cryptography uses one key for encryption and another key for decryption, and a hash function is a one way function which encrypts a message irreversibly. Each of the three types includes a variety of algorithms, and each of these algorithms has advantages and disadvantages.

We first describe secret key cryptography. The Digital Encryption Standard (DES) is considered an old fashion cryptographic algorithm due to its small key size, 56 bits. It was first published by NIST in 1977 and is now susceptible to brute force attacks. It is also relatively slow when implemented in software, but is more efficient if implemented in hardware. Triple-DES is more secure than DES as it performs multiple encryptions, but this enhances DES's disadvantage of being slow when implemented in software. The International Data Encryption Algorithm (IDEA), on the other hand, was designed for

efficient software implementations and it uses a larger key, but was protected by a patent and is considered relatively slow. Lastly, Advanced Encryption Standard (AES), which succeeded DES, accepts a variety of key lengths and is efficient for both software and hardware implementations. [16]

In public key cryptography, each algorithm is useful for key exchange, digital signing, and encryption, but these algorithms cannot be used interchangeably – as they have different properties. For example, the Diffie-Helman (DH) algorithm is useful for key agreement, but not for signing performance. The Digital Signature Algorithm (DSA) is used for creating digital signatures, but cannot be used for key exchange. RSA can be used for everything, from key agreement and signing to encryption capabilities. [17]

Hash function algorithms in contrast, are used to compute a digital fingerprint of messages to ensure that the message cannot be altered by a malicious party without this being detected. MD5 and SHA1 are the most famous hash functions. However, MD5 is now considered insecure due to some inherent cryptographic weaknesses. Both MD5 and SHA1 are susceptible to collision attacks, known as birthday attacks, which depend on the high probability of collisions occurring in random attacks. Refer to Kaufman, Perlman, and Speciner [16] for more information about cryptography and cryptographic algorithms.

As each type of cryptographic algorithms may benefit different applications, combinations of them can be used to exploit all their diverse capabilities. Public key cryptography is ideal for key exchange in an unsecure network and also for ensuring user authentication and non-repudiation (verifying that the sender is actually the one who sent the message). But public key cryptography is much slower than secret key cryptography with respect to encryption. Secret key cryptography, on the other hand, is ideal for encryption to ensure data privacy and confidentiality. Furthermore, hash functions are widely used for ensuring quick data integrity checking. Combining the aforementioned characteristics of the different types of cryptographic algorithms, we can perform data encryption utilizing the most appropriate type of algorithm. This cryptography method is called a digital envelop and it works as follows:

1) Session keys are created periodically and these session keys are used to encrypt the actual message(s);
2) The receiver's public key is used to encrypt the session keys when key establishment is needed;
3) A hash function is used to create a relatively small fixed length message digest;
4) The sender's private key is used to encrypt the message digest in order to achieve authentication and non-repudiation;
5) The receiver's private key is used to decrypt the session key and
6) The session key is used to decrypt the message;
7) The sender's public key is used to decrypt the message digest and finally
8) The receiver calculates a message digest of the decrypted data using the same hash function that the sender used and compares this with the received digest to ensure authentication and non-repudiation of data.

3.1.2 Video Encryption Algorithms

Due to real-time data constrains (real-time display or replay) encryption of real-time video is hard to achieve, hence video specific encryption techniques have been proposed. The naive algorithm approaches encrypt the entire compressed multimedia stream using one encryption method. Research efforts in the mid-1990s proposed the development of several algorithms in order to increase the speed of these naive algorithms. Some of those algorithms are known as selective encryption algorithms because encryption is applied to only part of the video data, for example to the most important coefficients from the compression process and encrypting only those coefficients. In contrast some researchers developed new mechanisms completely different from the selective encryption approach. As noted by Singh and Manimegalai in [18], with careful consideration, these alternative techniques can have similar security benefits as the naive algorithm approach and can also be significantly faster by reducing the computational cost of naive algorithms. To better understand the notion behind these video encryption algorithms, we classify the encryption algorithms based on their relation with compression schemes.

3.1.2.1 Joint Compression and Encryption Algorithms

The principle behind joint compression and encryption algorithms is to apply the encryption procedure in one of the compression stages, for example after transformation, after quantization, or within entropy coding, to achieve a scrambled output stream which cannot be accessed by third parties without access to the encryption key. Some of the most prominent joint compression and encryption algorithms are RVEA (Real-time video encryption algorithm, Shi et al., 1999) adding only 10% encryption overhead, saving up to 90% as compared to naive algorithms. Furthermore, RVEA does not reduce the compression efficiency. REC/RPB [19] save up to 50% compared to naive algorithms' computational cost, but are a lot slower in encryption than other schemes. All of the joint compression and encryption algorithms display drawbacks related to the video coder/decoder (CODEC). As in their implementation they alter the CODEC, hence the stream cannot be played back with a standard video CODEC equipped device – hence they require special equipment for playback. More unfortunate, most of them are susceptible to known- and chosen-plaintext attacks which reduce their security efficiency and they cannot be used for perceptual encryption. [20]

3.1.2.2 Compression Independent Algorithms

The main idea behind compression independent algorithms is that encryption is completely unrelated to compression. More specifically, those algorithms perform encryption prior to compression or encryption after the compression. Encryption before compression minimizes the compression computational cost and not the encryption cost as the encryption step minimizes the redundant input text so finally there is little plain-text that can be compressed. As a result, encryption before compression is rarely used as it minimizes the benefits of compression and increases the cost of encryption.

Encryption after compression has little similarities to naive algorithms, although it may sound so. The encryption can be performed in specific parts of the compressed image data

stream, using a combination of I-frames and P/B-frames and thus reducing the encryption computational overhead, as encryption is not performed on the whole compressed stream. These algorithms are the most prominent algorithms as they are applicable to standard video CODECs, they maintain the compression efficiency as they are used after compression and hence can reap the compression benefits, and they can reduce the encryption overhead by up to 90% comparing to naive algorithms (see for example the puzzle algorithm, Liu and Koenig, 2005 [20]). Unfortunately, these algorithms are still in research and most of them are vulnerable to known-plaintext and perceptual attacks[20].

3.1.3 Comparison and Conclusion on Security schemes on IP Video Surveillance systems

The two categories of video encryption algorithms have some common strengths and weaknesses in comparison to the naive algorithms. Both of them have lower encryption overhead, with joint compression encryption algorithms better minimize encryption overhead. Joint compression and encryption algorithms require an alteration in the video CODEC in order to play back via an existing system, implying that these algorithms cannot be used for systems were coding/decoding occurs in hardware. Compression independent and naive algorithms on the other hand, do not face this problem. Most importantly, both categories of the video specific encryption algorithms offer weaker security in comparison to naive algorithms, hence most of them are not acceptable for security sensitive applications [20]. This is the main reason why none of these video encryption algorithms are included in the international standards for information security. Research has yet to find a design for fast but secure video encryption. For this reason, this thesis project considers only the conventional (naive) algorithms in the implementation of the proposed security design. The author of the thesis together with the industrial supervisor decided that is better to sacrifice the compression and encryption overhead advantages of video encryption algorithms for the higher security and international standard compliance of naive algorithms, as video surveillance systems carry sensitive personal information.

3.2 Cryptographic libraries in C/C++

As cryptography became an essential part of video surveillance systems, as well as of other systems and applications, cryptographic libraries emerged in order to unify the different cryptographic algorithms and implementations, minimizing the time spent writing code and avoiding security pitfalls. There are a lot of cryptographic libraries; some of them are open source and some of them require licenses, even if their source code is open to the public. Crypto++, OpenSSL, Botan, and LibTomCrypt are famous examples of commercially well-known and widely used C/C++ cryptographic libraries.

To ensure unified video stream security in every part of the client's network, it is important to provide data privacy by establishing a secure channel between the devices operating in our network, as well as to provide data encryption for video streams stored by a NAS or any other storage device. For this reason, we have focused on C/C++ cryptographic libraries, as

Veracity's video surveillance network protocol is implemented in C++ and all the research and tests will be based on the company's existing test network.

The author of this thesis decided to implement a security scheme to be integrated with both the network and the COLDSTORE (by integrating security in COLDSTORE's C++ library as implemented by Veracity), enabling data encryption and device authentication. To implement this scheme, the OpenSSL library was used as it is a robust, open source library, and it implements all of the prominent cryptographic algorithms. It also provides a specific software component to enable applications using it to be declared "*FIPS compliant*" (see section 3.2.2 for more information). Because it is an open source project there are a lot of examples online and it has quite rich documentation (although these documents are scattered over different websites).

3.2.1 OpenSSL Basics

The OpenSSL Project is an open source toolkit which provides a robust, market quality, full featured product implementing the Secure Socket Layer (SSL) version 2/3 and Transport Layer Security (TSL) version 1 protocol, together with a general purpose cryptographic library. As the project is open source, it is maintained by volunteers throughout the world who use their collaborative efforts to design and develop this toolkit and its documentation. The library is written in C and is derived from the SSLeay library, developed by Erik A. Young and Tim J. Hudson in the middle of the 1990s. As the OpenSSL toolkit is licensed under an Apache-style licence, it is free for anyone to use it for any commercial or non-commercial purpose subject to some licence conditions[22]. This thesis project makes use of the cryptographic library, part of the OpenSSL project, to create a secure channel between our network's video surveillance system devices.

The OpenSSL cryptographic library includes the most prevalent cryptographic algorithms for secret key and public key cryptography, hash functions, and message digests. It also provides a pseudo-random number generator and various methods for manipulating typical certificate formats and managing key elements[16]. The service provided by this cryptographic library is used for OpenSSL implementations of SSL, TLS, and Secure/Multipurpose Internet Mail Extensions (S/MIME), and is also used to implement SSH, OpenPGP, and various other cryptographic standards[22].

OpenSSL is available for almost all UNIX operating systems (OSs), including Solaris, Linux, MAC OS X, and open source BSD OSs, as well as standard versions of Microsoft Windows and OpenVMS. It is freely available for download in source form from the OpenSSL's website[22]. Detail instructions for installation for all the aforementioned OSs can be found in the source distribution. Installation for UNIX and Windows OSs require Perl and a C compiler. In Windows distributions Borland C++, Visual C++, and GNU C compilers are supported for the OpenSSL distribution. Moreover, in order to include assembly optimizations in Windows, the system needs either MASM or NASM. Details of the system used in this thesis project to install OpenSSL and installation instructions are described in Chapter 4 where we discuss the methodology used to conduct our experiments.

3.2.2 FIPS Object Module

Companies' desire to validate their cryptographic schemes has led the OpenSSL community to release, on December 2012, a special version of OpenSSL validated under the FIPS 140-2 computer security standard. This object module meets the FIPS 140-2, Level 1 requirements. The OpenSSL Software Foundation (OSF) operates as the "vendor" for this validation. As the OpenSSL community notes in [22] "*OpenSSL itself is not validated, and never will be. Instead a special carefully defined software component called the OpenSSL FIPS Object Module has been created*". This FIPS Object Module is unique among all the other FIPS validated products as the module is delivered in source code form and requires specific build and installation instructions in order to be used as a validated cryptographic scheme. If the application that the user is using needs even the smallest modification of the source code or the build process, then the module cannot claim to be FIPS 140-2 compliant.

The most recent validation source of the FIPS Object Module that this master thesis security scheme uses is the OpenSSL FIPS Object Module v2.0, including FIPS 140-2 certificate numbered #1747. This component has been designed to be compatible with the standard OpenSSL library and API, thus any scheme using the standard OpenSSL library can be smoothly converted to use FIPS 140-2 validated cryptography.

Before the 12th of February 2013, OSF assisted users' efforts in requirements for code or build process modifications, at a minimum price. These validations were named "cookie cutter" or "private label" validations. Unfortunately, OSF is no longer accepting new "private label" validations due to the interpretation of the "guidance" in section 9.5 of the Implementation Guidance (I.G. 9.5) document [23] enforced by CMVP. The interpretation imposes some difficult code changes to libraries such as the FIPS Object Module and those deriving from them, so the OSF team decided to stop supporting requests for "private label" validations, at least for a period of time[24].

In this thesis project we decided to implement our secure communication channel and video stream encryption on the NAS side utilizing the OpenSSL FIPS Object Module v2.0 library in order to provide "FIPS compliant" characteristics to the proposed security scheme. This means that our security scheme uses cryptographic modules whose algorithms and operations are validated to be FIPS 140-2 compliant.

3.3 VPN

In this master thesis project we used VPN technology in order for video stream data to be securely transferred across our network. A VPN provides a fast, secure, and reliable point-to-point connection between two or more remote points which are connected with one or more unsecured links. This connection creates a VPN using the underlying physical networks to enable a secure communication between the connected parties. Because a VPN can use the existing networks, such as Internet, it is often used for private networks' connection via public networks, without bearing the high cost of leased lines[25].

VPN software sets up a tunnel between the connected parties where each payload is encrypted and encapsulated inside another packet, and then sent via the layer below over the network. This so called tunnelling technique enables secure data transfer between the endpoints by isolating the traffic from other parties while using public or private networks[26].

3.3.1 VPN security

A VPN enables data security by ensuring authenticated remote access and data encryption. These services provide sender authentication, traffic confidentiality, data (such as video stream) integrity, and non-repudiation regarding the sending device. More specifically, the endpoints of a tunnel should authenticate each other before the tunnel is established to ensure that the data is accessible only by authorised devices. For integrity and authentication purposes a VPN uses hash algorithms, such as MD5 or SHA1, and key exchange (frequently through public/private certificate deployment in the end device). When data is transferred the communication channel is secured by using encryption mechanisms. [25]

There are various VPN security technologies implemented nowadays with the most popular being:

a) **Internet Protocol Security** (IPsec) is a security protocol for Internet Protocol (IP) communications. IPsec provides end-to-end security by implementing a secure tunnel where authentication and data encryption is enabled. A major characteristic of this technology is that it implements data encryption at the network layer, i.e., at the IP level.
b) **Secure Socket Layer/Transport Layer Security** (SSL/TLS) are protocols that provide communication security implemented at the transport layer. SSL/TLS can implement a secure tunnel, where all data is encrypted and authentication by the endpoints is provided.
c) **Point-to-Point Tunneling Protocol** (PPTP) is a method to implement a VPN running on the data link layer. Although the PPTP specifications do not state the need for encryption or authentication, most PPTP implementations provide this security functionality. The most prevalent implementation is that used in Microsoft's Windows OS. PPTP implementations mostly provide tunneling to point-to-point protocol (PPP) packets implementing both authentication and encryption.
d) **Layer 2 Tunneling Protocol** (L2TP) is another tunneling protocol which supports VPNs. It does not provide encryption or authentication, but it can be combined with security protocols, such as IPsec (L2TP/IPsec), for encryption and authentication support through tunneling. [25][26]

3.3.2 OpenVPN

In this master thesis we establish a secure communication channel between the system's devices using the OpenVPN application. OpenVPN is an open source software application that implements secure VPNs utilizing SSL/TLS technology. Specifically, OpenVPN tunnels the traffic of any IP subnetwork or a virtual Ethernet adapter over UDP or TCP connection

using all the security methods (encryption, authentication, certification features) of OpenSSL library[29].

OpenVPN captures the incoming traffic from the operating system, using a virtual network interface (VNI), typically implemented as a TUN/TAP device. The TUN device emulates a network device with a point-to-point interface operating with network layer packets such as IP packets. The TAP device on the other hand, emulates a virtual Ethernet network device operating with data link layer frames, such as Ethernet frames[30]. Consequently, the VNI acts like a real network interface which is visible from all applications and to all users. The VNI sends outgoing captured traffic to the OpenVPN application located in the user-space, where the data are encrypted with the help of OpenSSL, and then this encrypted data is delivered to the real (Ethernet, Wi-Fi, etc.) network interface. The fact that OpenVPN is a user space application provides cross-platform flexibility, as the application can be ported to any operating system without alterations. The VNI also captures incoming traffic which is sent to the OpenVPN application to decrypt the transferred data and then the resulting cleartext packets are routed to the destination application. Figure 3.1 depicts an OpenVPN tunnel between two endpoints, while Figure 3.2 depicts the packet flow from one side of the tunnel to the other. [25][26]

3.3.2.1 SSL VPN

As we described in section 3.2.1, there are many VPN technologies used today. OpenVPN implements an SSL VPN connection and is not compatible with any other VPN technologies, such as IPsec, L2TP, etc.[27] The reason for using a SSL VPN instead of IPsec is portability across operating systems, because it uses a user space implementation rather than a kernel space implementation as IPsec generally does. A SSL VPN is also firewall and NAT-friendly, offers configuration simplicity, dynamic addressing, and supports multiple protocols [29].

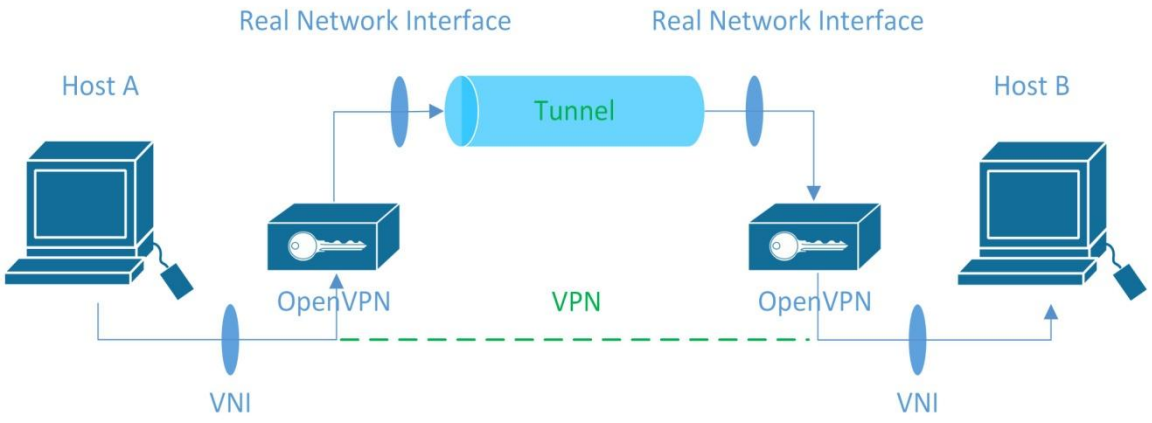

Figure 3.1: OpenVPN Tunnel between two endpoints (Adapted from Figure 1 on page 102 of [28])

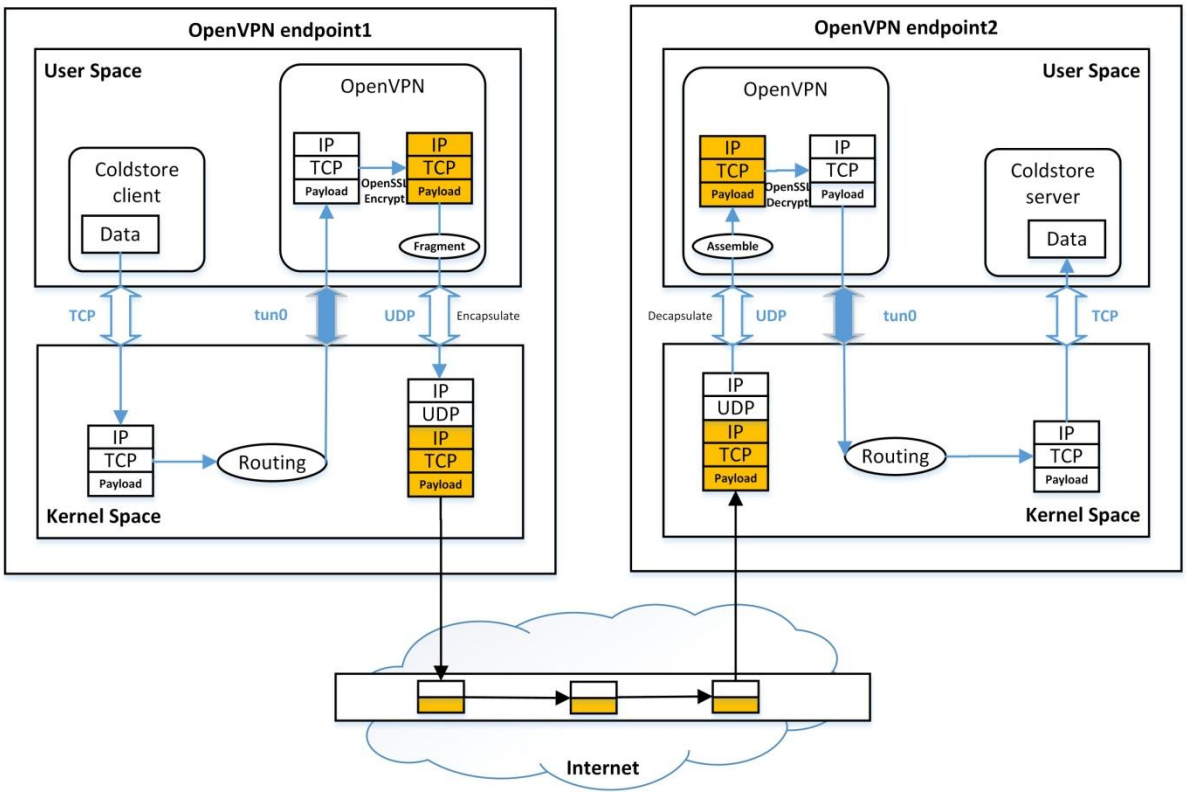

Figure 3.2: OpenVPN packet flow between two endpoints via tunnel (Adapted from Figure 2 on page 7 of [25])

The main objectives of OpenVPN SSL/TLS implementation in client/server mode, is client and server authentication using a public key infrastructure (PKI), making use of certificates and private keys, and secure connection provisioning by exchanging encrypted messages.

3.3.2.2 Security model

OpenVPN provides a security model to protect applications from both passive and active attacks. To enable this protection it uses either pre-shared static keys or SSL/TLS and certificates for key exchange and authentication. When using pre-shared static keys, the keys have to be created and shared between the OpenVPN endpoints over a secure channel, before the tunnel is used. In contrast, SSL/TLS mode uses certificates to authenticate a bidirectional SSL session and different encryption/decryption keys are used in each end-point, which solves the problem of key deployment. Pre-shared keys use symmetric encryption algorithms so it is easier and faster to implement security and consumes less CPU power as compared to SSL/TLS mode. However, SSL/TLS provides a better authentication scheme by using certificates and key renewal, as it uses asymmetric and symmetric encryption algorithms to their best advantage and does not require a prior key exchange. [29]

3.3.2.3 Routing versus Bridging

OpenVPN provides two distinct ways for interconnecting networks: routing and bridging. Routing interconnects separate networks residing in different IP subnets. When a packet is received, a network router examines the destination IP address and forwards the packet to the

appropriate outgoing interface which is connected to the appropriate network by the router. In addition, bridging interconnects networks directly to layer 2, which means that Ethernet frames instead of IP packets are passed to the tunnel. As bridging works in layer 2 it can send data packets using any protocol, such as TCP/IP or even the older NetBEUI and IPX/SPX, and can assist any device that can send Ethernet frames even the ones that use proprietary protocols such as home electronics and voice over IP telephones.

In OpenVPN both options have their advantages and disadvantages. We have chosen to connect our devices using the routing option as we want to serve a large number of devices in different IP networks (for example, different IP cameras available in different subnets communicating with COLDSTORE device which may reside also in another subnet). Moreover, routing add less traffic overhead as it transfers only IP packets destined to the VPN client, and provides better control of the access rights on the client side. Disadvantages of our implementation are that we can only transfer IP packets and TUN interface cannot be used in bridges. As a result, we may face limitations to the devices that can be used in our network as some devices, such as DVRs, use proprietary protocols. [29]

Chapter 4

Method

For creating our security scheme and analysing its impact on the video surveillance systems, different test scenarios were created. These scenarios measures the system's performance decrease due to additional overhead, introduced jitter, and one-way delay; and collect statistical data based on a variety of metrics (throughput, latency, packet loss, etc.). Therefore, section 4.1 describes the methodology that was used to collect and analyse our data. Section 4.2 describes the network topology set up, while section 4.3 presents the security design by describing the cipher algorithms and cryptographic libraries that were used, and gives useful information about the OpenVPN setup. Lastly, section 4.4 presents the different test case scenarios and chapter 4.5 describes the different metrics that were used and their characteristics. All the instructions for installation are given for Windows OS as the project is based on this OS, but there are instructions for other OSs in the appropriate references.

4.1 Methodology

Quantitative inductive method is used to collect and analyze the data based on the different metrics (i.e. overhead, jitter etc.). It is quantitative method because the data collected, packet header, number of packets send, one-way delay etc. are numerical data that form measurable quantities. The measurement units are known and can be compared, and this comparison can be expressed in numerical values.

It is inductive because the methodology used does not rely on any previous theory. The data were collected and their results were analysed to derive a hypothesis which will ultimately form a final theory. In contrast to the deductive method which starts with a hypothesis based on known facts and laws, then collects data in order to confirm or reject the hypothesis.

The investigation utilizes data collected with the help of software and does not embrace any human involvement other than the author of the thesis that collected the data. Since data were collected by computer software, we ensure their reliability. The data also can be replicated, but only if we use the same hardware and software. The reason lies in the processing and memory capabilities of the different hardware devices which can affect delay; as well as different network packet analysers might be more or less precise with respect to the capture time (hence affecting the estimation of the time that it takes a packet to be sent or received from the other end-point). Finally, data were collected at least three times for each experimental configuration and the results are compared in order to test their validity.

In the following sections we thoroughly describe how we built our network based on the aforementioned methodology, what data we collected, and which software we used for the data collection.

4.2 Network topology

A simple example of the deployment of Veracity's client network consists of a few cameras connected with a DVR/NVR device which is also connected with a COLDSTORE device. Figure 4.1 shows this simple network topology. The author of this thesis proposed to introduce a security scheme between the DVR/NVR device and the COLDSTORE device. The reason for this proposal is that only the DVR/NVR and COLDSTORE devices have an OS which can easily support a VPN connection. If the analogue cameras of the Veracity's clients' networks are replaced with IP cameras, then the secure channel can also include these IP cameras. In that case end points of the secure channel are considered the IP cameras (clients) and the COLDSTORE device (server), thus the NVR devices can be omitted.

Given the limitations stated above, we have created a network topology for testing purposes to emulate the communication between a DVR/NVR device and a COLDSTORE device. To implement this test network we used three laptop computers interconnected with a switch through two Ethernet cables. A network time protocol (NTP) server, acting as a stratum 1 time source, is also connected to the switch using an Ethernet cable in order to synchronize the devices' clocks. One laptop acts as a DVR/NVR device generating client commands for video stream storage to another laptop, which acts as a COLDSTORE device. Both laptops use the COLDSTORE library, implemented using the client-server model with the COLDSTORE device acting as the server. The third laptop is used as a monitor to capture packets sent. These packets are collected by using port mirroring configuration on the LAN switch. This test bed network topology, depicted in Figure 4.2, allow us to build an accurate, transparent and replicable testing environment to analyse the proposed security scheme.

The 2 Lenovo laptops are identical with 4GB of memory and run a 64-bit Microsoft Windows 7 operating system (OS). The processor is an Intel® Core™ i5-2450M clocked at 2.5 GHz. The switch is a NetGear GS108T V2 model and the NTP server is a TIMENET PoE powered device produced by Veracity. The Ethernet cables are category 5e cables allowing data rates up to the maximum data rate of the laptops' Ethernet interface, in this case a JMicron PCI express gigabit Ethernet adapter version 6.0.17.1 configured to run at 1000 Mbps in full duplex model.

The switch is a store-and-forward device which powers up 16Gbps in full duplex mode through 8 ports. It is ideal for service message block (SMB) networks transferring Voice over IP (VoIP) or streaming media and many other bandwidth-intensive services. The network latency introduced by the switch in store-and-forward mode for 1000Mbps transmission is 15μs for 64-byte frames. [31]

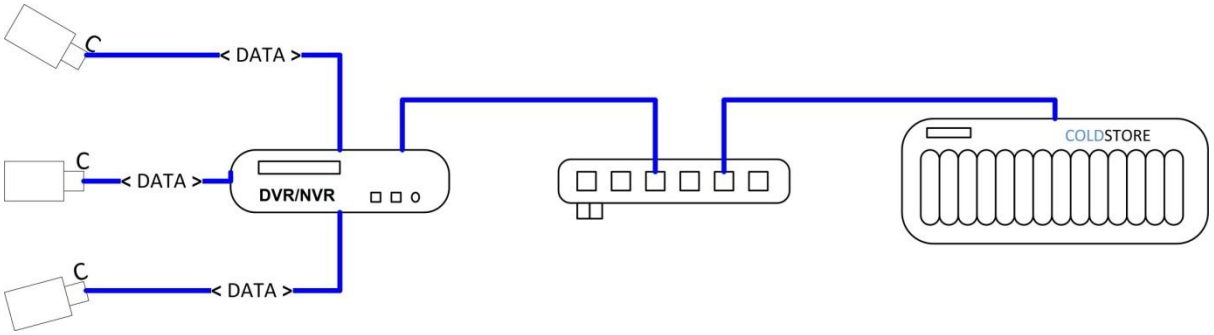

Figure 4.1: Simple example of Veracity clients' network

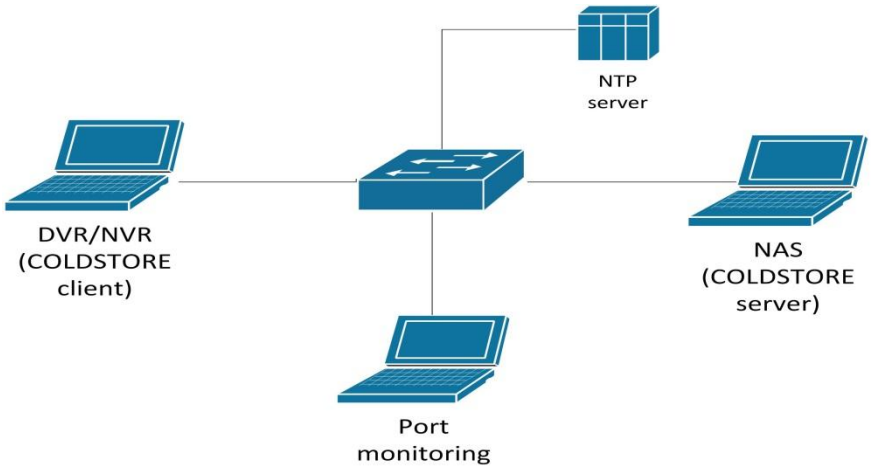

Figure 4.2: Network topology used for testing purposes

As video streams captured by actual video surveillance cameras carry sensitive personal information and their access should be restricted to authorised parties, we simulate video traffic by transferring the Big Buck Bunny open movie, created by Blender Institute (part of Blender Foundation[32]). The movie is encoded using an MPEG-4 CODEC, having approximately 24 frames/s frame rate, 448 kbps audio bit rate, 12007 kbps video data rate, and 1920x1080 resolution. The file size is 885MB encoded using variable bitrate (VBR). The video's specifications are similar to the actual video stream requirements captured by Veracity clients' video surveillance systems. A networked attached storage (NAS) device is used to store large video files and to deliver this data at variable data rates as we change the simulated video capture environment.

4.3 Security design

To ensure video stream privacy we designed a security scheme which creates secure communication channels between the network's devices and provides data encryption of the stored stream. The author of this thesis decided to establish a secure tunnel between the IP cameras/DVRs/NVRs and the COLDSTORE device to secure their communication. In our test bed network topology, the secure channel was established between the NAS device (a laptop computer acting as a COLDSTORE server) and the DVR/NVR device (acting as a COLDSTORE client), as depicted in Figure 4.2. To implement this secure tunnel we used

OpenVPN. OpenVPN was selected specifically because of its cross-platform portability as the network devices in an implementation of this solution could utilize different OSs.

Along with the implementation of the secure tunnel, the author of the thesis decided to implement a cryptographic C class. This class, was integrated into the COLDSTORE device software enabling it to encrypt the video stream data at the NAS, thus we can provide fully secure communication within our network and securely store the collected video stream.

Both the OpenVPN and the cryptographic C class use the same cipher algorithms to create the secure tunnel and to provide encryption and authentication in the NAS device. These algorithms described in section 4.3.2, depend upon the OpenSSL library to call the appropriate cipher functions to provide the desired privacy. Specifically, as mentioned in section 3.2.2, we use the OpenSSL FIPS Object Module to make our security scheme FIPS compliant. In the following subsections, we describe how we installed and utilized the FIPS Object Module with OpenSSL and the OpenVPN application to accomplish our security scheme.

4.3.1 Initial design

Firstly, the author of this thesis together with the industrial supervisor decided to implement a security scheme to be integrated both with the network and the COLDSTORE (by integrating security into COLDSTORE's C++ library as implemented by Veracity), enabling data encryption and device authentication. Our first attempt to implement this integration of the OpenSSL library into COLDSTORE's C++ library by introducing methods for encryption, decryption, signing, and verification of data using the appropriate OpenSSL methods. The aim was to compare the performance metrics of this security method, where only the video stream data is encrypted and signed, with the performance of the VPN, where the packet header is also included in the encryption and signature. The advantage of this approach is that this security scheme could be used to perform encryption on the NAS side *and* to protect stored data.

Unfortunately, we were unable to use the OpenSSL library with the COLDSTORE's C++ library due to installation issues with the OpenSSL FIPS Object module. By default the FIPS object module is installed based on the system's specifications, in our case two laptops with 64 bit processors. When attempting to install a 32bit version of OpenSSL FIPS Object module, as COLDSTORE's C++ library is built for 32bit processors, we were unable to verify the module's fingerprints; hence this implementation would not meet FIPS requirements. Due to having a limited time we decided to skip this implementation and to continue with measurements and evaluation of the performance metrics for the proposed VPN security scheme. Given these measurements we can estimate what the performance would be were we not to include the packet headers in the data that was being encrypted and decrypted.

4.3.2 Cipher Algorithms

As stated in chapter 3 our security scheme for IP video surveillance systems is based on naive algorithms. Specifically, as mentioned in section 3.1, public key (symmetric)

encryption is more secure than secret key (asymmetric) encryption, thus the RSA algorithm with a 2048 bit key length is used in order to establish session keys between the COLDSTORE server and the COLDSTORE client. The RSA algorithm was chosen as it can be used for both encryption and authentication. As the underlying premise behind the security of the RSA algorithm lies in the difficulty of factoring large numbers, the selected key size should ensure that the proposed solution will be secure for many years [33].

On the other hand, secret key cryptography exploits rapid key creation, thus secret key encryption is used for encrypting the actual video data utilizing AES 128 bits in counter (CTR) mode of operation for session key creation. The reason for choosing AES as the secret key algorithm is that AES is the prevalent secret key cryptographic algorithm used by cryptographers and cryptographic systems. The key size that has been chosen is the minimum key size for AES, requiring only 10 cycles of iteration, slightly improving the algorithm's speed compared to the 12 and 14 iterations, required for 192-bit and 256-bit key lengths respectively. The length of the key results in a complexity of 2^{128} brute force attacks. This key size is considered unbreakable at the time this thesis was written. Counter mode is suitable for random accessed files [16], such as COLDSTORE's data chunks, as the key for any given offset can be generated.

To assure authentication and non-repudiation of data we compute a message digest with the SHA-256 hash function and then we sign the message digest with the device's private key, in our test scenario with the COLDSTORE device's private key. We have chosen SHA-256 as it is a high speed hash function; it has no major vulnerabilities, such as the collision attacks against the SHA1 family, and this hash function is preferred by Veracity's clients as it has been used as the default option in various applications [34].

4.3.3 FIPS Object Module build and installation with OpenSSL

The OpenSSL FIPS Object module, which is only available in source code, was downloaded from the OpenSSL's website[22]. The version used in this thesis project is 2.0.2 tested on Windows x86-64 with SSE2 32 and 64 bits, similar to our laptops' specifications.

The CMVP has introduced new restrictions for the verification of the OpenSSL's version 2.0 source code distribution. Specifically, the source code should be obtained via a "trusted path", such as from OSF on a CD. Because the purpose of this thesis project is to implement a test scenario to test our security scheme rather than install it on the end system, we have skipped this step. Instead, we downloaded the tarball straight from the OpenSSL's website [22] and the integrity and authentication of the Module was verified by checking its signature using the GPG application [35].

The next step in this installation procedure is to build the module. Using a command line window, we navigate to the OpenSSL FIPS Object Module source code folder and we run the command `ms\do-fips no-asm`. The `no-asm` option is used to avoid the use of assembly language optimizations in order to make our test scenario scalable to multiple processors with comparable results.

The final step of OpenSSL installation is to compile a standard OpenSSL distribution and link it with the already built and installed FIPS Object Module. For this reason, we downloaded OpenSSL standard library version 1.0.1e and we ran the following commands:

- `perl Configure VC-WIN64A fips –with-fipslibdir=c:\fips\path` (where "`c:\fips\path`" is where the `fipsscanister.lib`, `fipscanister.lib.sha1`, and `fips_premain.c` are located from the previous step)
- `ms\do_win64a`
- `nmake -f ms\ntdll.mak`
- `cd out32dll`
- `..\ms\test`

where the last command initiates cipher tests to check that all the algorithms have been installed correctly. The instructions followed to install FIPS Object module and OpenSSL standard API are specific and documented in OpenSSL's UserGuide [36]. If everything is well installed then the tests should be successfully passed and we can use the OpenSSL standard API which is now FIPS compliant.

4.3.4 Cryptographic C class

The proposed cryptographic C class provides encryption, authentication, integrity, and non-repudiation to a video stream by implementing four distinct functions which enable encryption, decryption, signing, and verification of data (respectively). This cryptographic C class can be utilized by the COLDSTORE C++ library in order to encrypt and sign a stored video stream in the COLDSTORE device to ensure data privacy. This recorded video stream can also be viewed by using a DISKPLAY device, a device produced by Veracity to permit replay and review of a video stream stored on the COLDSTORE disks running on a PC, in order to decrypt the video stream and verify its integrity, authenticity, and non-repudiation.

To implement the cryptographic C class we have used the high level cryptographic interface of OpenSSL's library, called EVP. More specifically, we used EVP_PKEY… functions for key management, EVP_Seal… and EVP_Open… for public key encryption and decryption, and EVP_DigestSign… and EVP_DigestVerify… to implement digital signatures [22].

4.3.5 OpenVPN installation and configuration

The secure tunnel between the communicating laptops participating in our network topology was established by installing OpenVPN in both laptops following the instructions provided at the OpenVPN website [37]. In our test scenario we installed OpenVPN version 2.3.1 for x86-64 bit architectures using the self-installing exe file for Windows.

OpenVPN includes and uses a version of the standard OpenSSL library to provide encryption, integrity and non-repudiation of transferred data, and endpoint authentication. As in this thesis project FIPS compliance is a key goal, we have replaced the standard OpenSSL distribution with the FIPS capable OpenSSL that we previously installed. Specifically, we replaced the `openssl.exe`, `libea32.dll`, and `ssleay32.dll` files residing in the

OpenVPN's bin folder with the ones residing in the out32dll folder of the FIPS capable OpenSSL.

4.3.5.1 PKI and certificate establishment

OpenVPN supports bidirectional authentication of the tunnel's endpoints using certificates. This means that the server must authenticate the client's certificate and the client must authenticate the server's certificate. For this reason, an essential stage in creating a secure VPN connection is the establishment of a PKI. The PKI includes a certificate (a public key bound to a *distinguished* name) for the server and each client of the network, and a master certificate authority (CA) certificate and a private key used to sign the server and clients' certificates. The master CA helps both the server and the clients to authenticate each other by verifying that the presented certificate is signed by the trusted CA. Then they verify the information included in the certificate's header, such as the certificate's common-name or the type (client-server).

This trusted chain has various advantages for VPN architectures:

a) the server's certificate management is minimized as it needs to know only its own pair of certificate and private key and not the certificates of each potential client;
b) the server needs only to verify that the client's certificate was signed by the master CA, as the server can verify the signature using the CA's public key, thus eliminating the need for the CA to reside in the same machine as the server or even to be connected to a permanent network;
c) a private key's compromise can be easily handled by adding its certificate to a certificate revocation list (CRL) (however, dealing with CRLs will require that the CRLs be distributed by the network to all of the relevant devices); and last
d) the server can provide different access rights to clients based on certificate fields, such as the common-name.

There are two types of CAs: private and public. A private CA has the responsibility to sign certificates and be trusted only by the members of its own organization, whereas a public CA, such as Verisign, issues certificates to any member of the public. For our thesis purposes a private CA is suitable as we only need to authenticate members of our closed network.

Consequently, to assist our test scenario we generated three pairs of certificates and private keys: a master CA pair, a server pair and a client pair. All these key generations were stored in the server for simplicity purposes. For the PKI management we made use of *easy-rsa*, a set of scripts included in OpenVPN package.

To generate the aforementioned certificate/private key pairs, firstly we open a command prompt and navigated to the easy-rsa folder, which is located inside the folder where OpenVPN was installed. Then we run the following batch file which initializes the configuration file (or overwrites the existing one):

```
init-config.
```

Then we edit the generated `var.bat` file by setting the KEY_CONFIG to point to the cnf file in the apps folder of the FIPS capable OpenSSL, the KEY_SIZE to 2048, and the KEY_HASH to sha256. Also, we have set appropriately the KEY_COUNTRY, KEY_PROVINCE, KEY_CITY, KEY_ORG, and KEY_EMAIL parameters.

Next, we initialized the PKI by running the following commands in the command window:

- `vars`
- `clean-all`
- `build-ca`

The `build-ca` command creates the master CA certificate and private key utilizing the FIPS capable OpenSSL library. While running the `build-ca` command we will be prompted to fill in the certificate parameters or select the default ones. We select the defaults in every field, but the `Common Name`, which we explicitly set to "*CA"*.

Next we had to generate the server and client pair of certificates and private keys. As depicted in Figure 4.2 we only have one server and one client, thus we need one pair of private and public keys for each. To do this we run the following commands in the command window to create the server and client pairs accordingly:

- `build-key-server server`
- `build-key IPCamera`

Note that for each of the commands we are prompted to set the certificate parameters, where we used the default values and set the `Common Name` to "server" and "IPCamera" respectively.

We must also generate the DH parameters for the server in order to periodically create session keys in order to maintain security of our tunnel. RSA is not preferred as it is too slow for key generation in comparison with DH. To generate the DH parameters we run the following commands:

```
build-dh
```

After this step it is time to copy the pairs of certificates and private keys to the relevant devices. In our project, for simplicity purposes, the master CA will be located in the server. Thus, ca.crt, ca.key, dh2048.pem, server.crt, and server.key will be located in the server device. The IPCamera.crt, IPCamera.key, and the ca.crt will be located on the client side.

To securely transfer the appropriate files from the server to the client we used the isolated Ethernet connectivity of our closed network that connects the two devices (see Figure 4.2). This was done for simplicity purposes, but in an actual deployment it is also possible to establish a PKI without using an existing secure channel. In this scenario the client device is responsible of generating its own private key locally, and then it sends a Certificate Signing Request (CSR) to a key-signing machine. The machine in return processes the CSR and sends

a signed certificate back to the client. This approach does not require the secret .key file to leave the hard drive of the machine where it was generated.

4.3.5.2 Configuration files creation

After the successful establishment of our CA and the generation and deployment of certificates and keys to the server and client, what is left is to create the OpenVPN configuration files for both the server and the client. In this project, we used the sample configuration files as our starting point. These files are located under OpenVPN\sample-config folder. Then we edited specific parameters in these configurations to customize our VPN tunnel.

The configuration file of the server is:

```
dev tun
port 1057
proto tcp
ca ca.crt
cert server.crt
key server.key
crl-verify crl.pem
dh dh2048.pem
server 10.44.66.0 255.255.255.0
ifconfig-pool-persist ipp.txt
keepalive 10 120
persist-key
persist-tun
status openvpn-status.log
verb 4
```

The configuration file of the client is:

```
client
dev tun
proto tcp
remote Eng-Laptop-PC 1057
resolv-retry infinite
nobind
persist-key
persist-tun
ca ca.crt
cert IPCamera.crt
key IPCamera.key
ns-cert-type server
verb 4
```

The above configurations enable us to create a basic VPN tunnel using the TUN network interface (for routing), listening to port 1057 for establishing new TCP or UDP connections,

and assigns addresses to clients from the subnet 10.44.66.0/24[1]. By default, OpenVPN configuration supports the Blowfish (BF) cipher in CBC mode and SHA1 for hash function. The ca, cert, key, and dh parameters points to the files that we generated above and help us to encrypt the video stream and authenticate the devices. Also, the crl-verify parameter, that is present only in the server configuration, points to a CRL file where all the revoked certificates are stored. Changes in this file take immediate effect on new connections and existing ones when they negotiate their SSL/TLS channel (by default this occurs once per hour).

4.4 Test scenarios

In order to analyze and evaluate the performance of our proposed security scheme and subsequently of our network, we have created three different test scenarios. The test scenarios compare the current network implementation, which uses TCP to send the packets over the Ethernet cable having no security scheme implemented, with the introduction of VPN technology to the current network, as well as the utilization of different cryptographic algorithms and authentication mechanisms to the VPN network. The scenarios implementing VPN connectivity can be run on TCP and UDP to compare protocol stack differences. Explicitly the test scenarios include:

a) a simple transfer without VPN utilization;

b) a default VPN configuration using BF in CBC mode and SHA1 function, as described in section 4.3.5;

c) default VPN connection, introducing extra integrity verification by adding an HMAC-SHA1 signature to the SSL/TLS handshake packets;

d) VPN using AES with 128 bits key in CBC mode using SHA256 function; and

e) VPN using AES with 128 bits key in CBC mode using SHA256 function, introducing extra integrity verification by adding an HMAC-SHA256 signature to the SSL/TLS handshake packets.

In order to enable the use of AES in CBC mode by the VPN we include the following entries in both the server and client VPN configuration file:

```
cipher AES-128-CBC
auth sha256
```

To include the additional HMAC signature in all SSL/TLS handshake packets we need to include an extra command in both the server and client VPN configuration files.

In the server configuration, we add:

```
tls-auth "C:\\Program Files\\OpenVPN\\easy-rsa\\keys\\server\\ta.key" 0
```

[1] This is a private network address range.

In the client configuration, we add:

```
tls-auth "C:\\Program Files\\OpenVPN\\easy-rsa\\keys\\server\\ta.key" 1
```

4.4.1 Video transfer

In order to test the proposed secure communication channel using the different test scenarios we need to simulate video traffic between the client and the server. As we do not have access to the real video streams captured by a Veracity network, we assumed that our traffic is generated by an IP camera with data rate 12007kbps (i.e., video data rate used to encode the Big Buck Bunny movie). In a real case scenario this video stream is stored in a file on the DVR/NVR device and COLDSTORE client application cuts the file into 1MB chunks and then sends it through the network to the COLDSTORE server application, which in turn saves the file with a timestamp in the COLDSTORE NAS device.

To simplify our scenario the industrial supervisor and the author of the thesis decided to skip the communication between the camera and the DVR/NVR device, as the delay that is added from this communication channel is always the same. That delay includes the time to deliver the video from the camera to the DVR/NVR device as well as the time for the device to analyse, compress the video stream, and send it to the COLDSTORE client application. We also assumed that the COLDSTORE client application always has data to read from the file, thus eliminating the delay of the COLDSTORE client application while waiting for 1MB to be available to be sent.

The COLDSTORE client application on our test scenarios reads 1MB from the Big Buck Bunny movie and creates TCP segments of 1500bytes. It also sets the TCP flag to “don’t fragment” so our network’s maximum transmission unit (MTU) should be set to a minimum 1500bytes.

4.5 Data collection and analysis

In our test scenario implementation we capture the packets transferred between both sides using a monitoring device and we also capture the packets sent and received on both ends. Useful information can derive from this packet monitoring, such as:

- Total delivery time of the video data from COLDSTORE client application to COLDSTORE server application – helping us identify the total delay introduced to our communication channel from the implementation of security;
- Encryption and authentication time per byte of data sent from client to server;
- Network overhead calculated by adding the different protocol headers introduced by the protocol stack to the video stream; and

- One-way delay variation measured by the time a packet is sent from the COLDSTORE client application to the time the packet is received on the COLDSTORE server application.

For each test scenario, the transferred data packets were captured using Wireshark [38], a network protocol analyzer. We installed Wireshark on our monitoring device to capture the packets travelling from both the server and the client device. In order to sniff packets in our network we selected our device's Ethernet interface, from the interface list, as shown in Figure 4.3. To collect information about our network's one-way delay, described in section 5.1.4, we capture a packet when entering the tunnel on the client device, until it explicitly exists the tunnel, thus monitoring the window's tap interface.

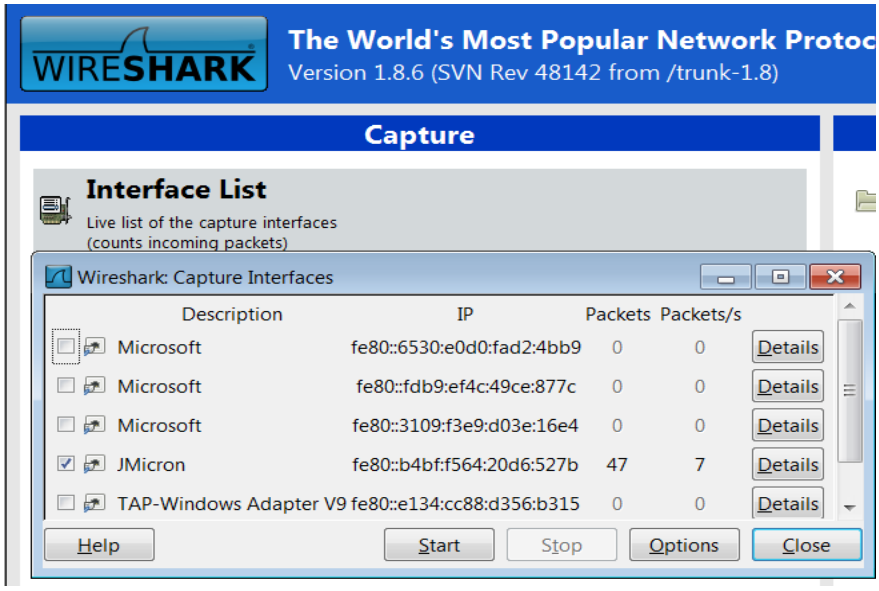

Figure 4.3: Sniff packets on Ethernet interface

We also used Wireshark display filters to filter out irrelevant traffic transferred on our network. Based on our test scenarios, we are only interested in TCP or UDP traffic regarding the VPN configuration. To display only the relevant traffic, we select the TCP or UDP option from the conversation list available under the statistics menu, as depicted in Figure 4.4. Wireshark will analyze the traffic and display the different conversations available in the current capture. It is useful for our throughput and one-way delay measurements to select only a specific client to server connection. Thus we select a specific conversation and we apply a filter to extract the traffic flowing from server to client, as shown in Figure 4.5. This results in Wireshark displaying only those packets generated by the server having as a destination the client.

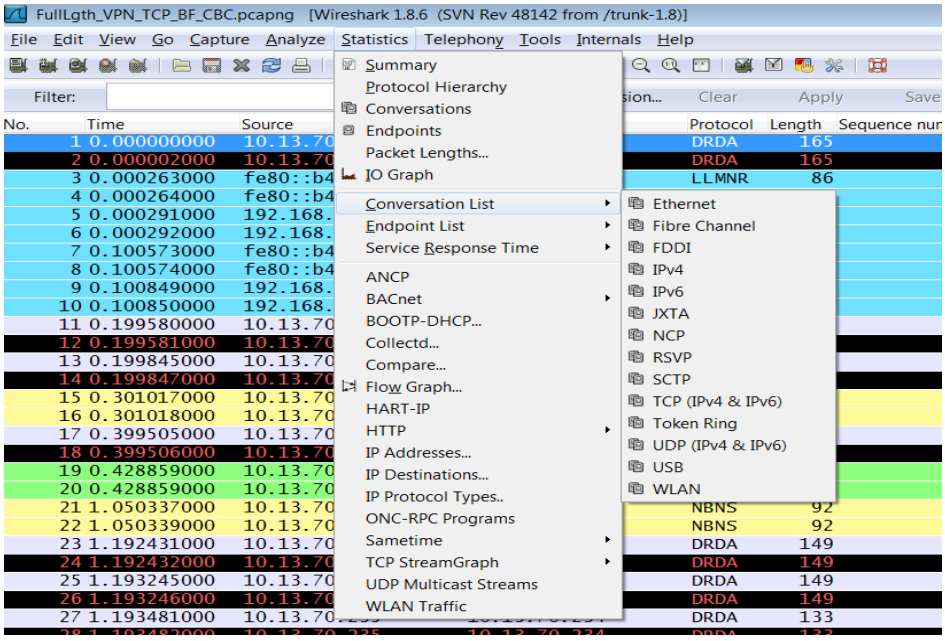

Figure 4.4: Filter out conversations in Wireshark

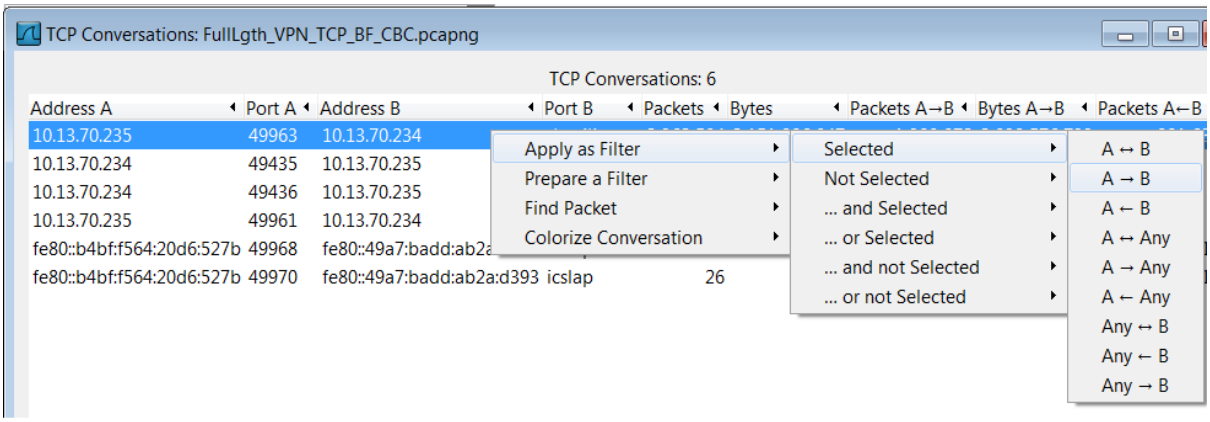

Figure 4.5: Select server to client connection in Wireshark

Finally, in order to see the traffic for the OpenVPN protocol and to be able to analyse the data under SSL/TLS protocol we edit the OpenVPN protocol preferences. Specifically we select *Preferences* under the *Edit* menu tab, and from the *Protocols* menu on the right panel we select *OpenVPN* and we set the OpenVPN TCP port and OpenVPN UDP port respectively, as shown in Figure 4.6.

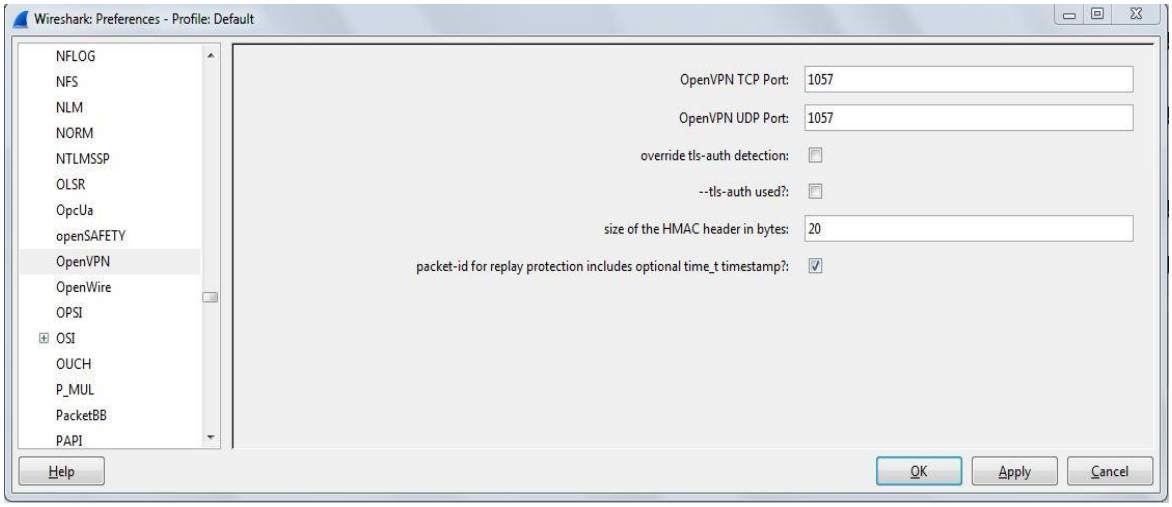

Figure 4.6: Edit OpenVPN preferences in Wireshark

Chapter 5

Measurements

In this section we will discuss and analyze the results collected from the test scenarios as described in section 4.4. In section 5.1 we analyse our network's performance in the different test scenarios in terms of transmission time, overhead, and one-way delay. In section 5.2 we describe certificate deployment obstacles in Veracity's video surveillance network and provide solutions to resolve them.

5.1 Performance measurements of the proposed security scheme

To assess and evaluate the proposed security scheme we analyse the network's performance utilizing different metrics. In the next section we will consider the total delivery time to transfer a video file from client to server, the network overhead introduced by OpenVPN, and the one-way delay.

5.1.1 Total Delivery time to transfer a video file

An important aspect of this project is how much time it takes for the video to be successfully delivered from the client to the server with or without the introduction of the proposed security scheme. It is of important that the video stream is transmitted as quickly as possible following its capture since the video surveillance system's effectiveness is positively affected by rapid identification of security incidences. For this reason, Figure 5.1 displays a comparison between the delivery times of a video stream from the COLDSTORE application client to the COLDSTORE application server for each of the different test scenarios (see Appendix A).

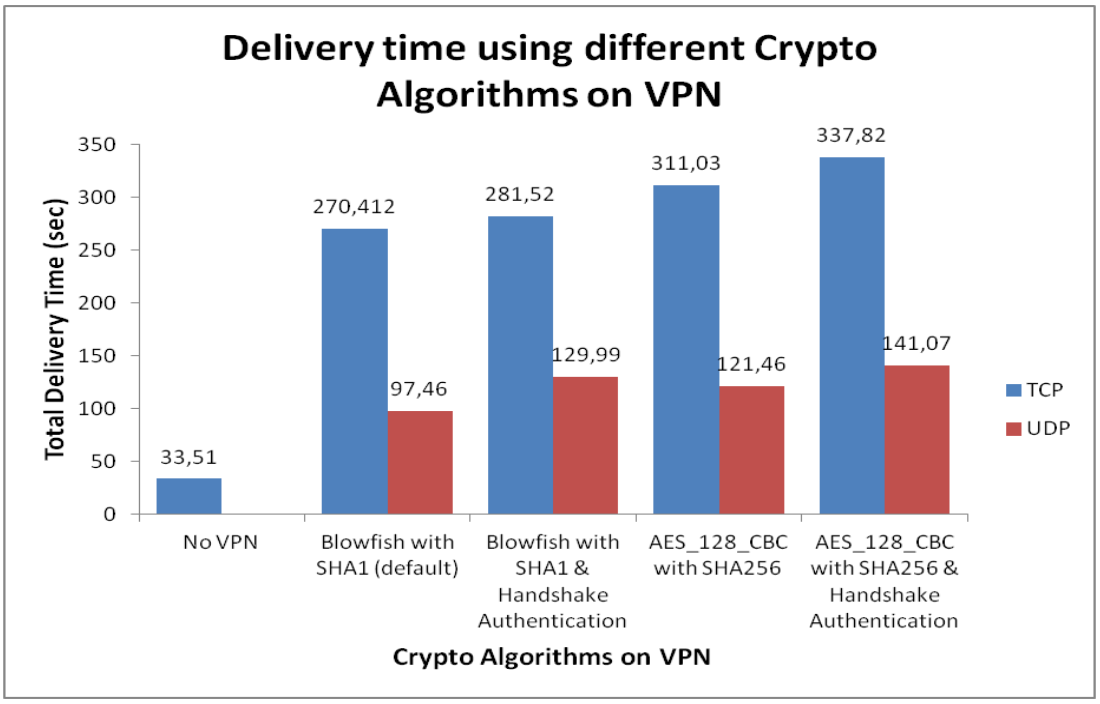

Figure 5.1: Total delivery time to transfer a video file from the COLDSTORE client application to the COLDSTORE server application with and without VPN utilizing different cryptographic algorithms.

The amount of time needed to deliver the video file nearly triples when using the default VPN connectivity with UDP packets and is 8 times greater when using TCP. Moreover, if we use the extra authentication check for every SSL/TLS packet the time increases slightly (of the order of 1% for both UDP and TCP VPN connections). The delivery time also increases by 1% when compared with the default VPN configuration when the VPN utilizes the proposed security scheme (VPN using AES using 128 bits key in CBC mode with SHA 256 hash function), for both UDP and TCP connections.

The big difference in the transmission time between VPN over UDP and VPN over TCP is an effect commonly known as the *TCP meltdown problem* [39]. This problem occurs when applications use TCP tunnelling. While we transmit a TCP packet through a TCP tunnel, two congestion controls operate on our communication, one end-to-end and one for the TCP tunnel. It is known that under certain circumstances the communication may experience a severe reduction in its throughput, thus degrading the end-to-end network performance. As shown in Figure 5.1, UDP tunneling provides much better performance in terms of a shorter delivery time than TCP tunneling. For this reason UDP tunnelling is preferred.

From the number of bytes sent from the client to the server, and the delivery times for each test scenario we can calculate the encryption and authentication time per byte of video data. The results are displayed in Table 5.1 below. The results presented in the table shows that the actual encryption and authentication time in milliseconds per byte of video data sent from the client to the server is similar for both algorithms. Thus, our security scheme choice can be supported as we can utilize the more secure AES algorithm rather than BF while experiencing the same encryption and authentication time per byte. On the other hand, running the VPN

over TCP increased the transferred time by 200% as compared to running the VPN over UDP, which supports our previous statement that we should use UDP for tunnelling.

Table 5.1: Encryption and Authentication time in milliseconds per byte of video data for the different test scenarios

	TCP BF CBC	**TCP AES128 CBC**	**UDP BF CBC**	**UDP AES128 CBC**
Encryption & Authentication Time per byte (ms)	0,003901546	0,00208116	0,006962201	0,001305314

5.1.2 Network overhead

The introduction of VPN tunnelling and the use of SSL/TLS for authentication and key negotiation negatively affect the video transfer performance by increasing the packet overhead. In the following section we will analyse the network overhead introduced in Veracity's network and we will compare it with the overhead introduced by OpenVPN over TCP and over UDP. We will also calculate the overhead introduced by the SSL/TLS handshake protocol and compare it with the overhead of the SSL/TLS session resumption protocol.

5.1.1.1 SSL/TLS overhead

In our network topology, as described in Chapter 4, the Ethernet's MTU was set to 1500 bytes. A simple video transfer in a Veracity's network use TCP/IPv4 protocols for the transfer, where normally each TCP segment has 20 bytes of TCP header (as TCP timestamps are not used), leading to 40 (2x20) bytes overhead in total (due to 20 bytes of IP header). So for video transfer over TCP/IPv4 the overhead in bytes is:

```
(40 * 100%) / (1500 - 40) = 2,7%.
```

In a tunnelled network the transport header doubles as two protocol stacks are used (a TCP/IP packet created by the COLDSTORE client application is encapsulated inside another TCP/IP packet introduced by the tunnel). Also a TLS record header of 5 bytes is added to the data transferred, which results to 85 bytes of header if VPN over TCP is used. Thus, for video transfer utilizing VPN over TCP the overhead in bytes is:

```
(85 * 100%) / (1500 - 85) = 6%,
```

Finally, if VPN over UDP is used, then the packet header is 73 bytes (the TCP/IP packet created by the COLDSTORE client application is encapsulated inside a UDP/IP packet introduced by the tunnel, where the UDP header is 8 bytes). Thus, for a video transfer utilizing a VPN over UDP the overhead in bytes is:

```
(73 * 100%) / (1500 - 73) = 5,1%,
```

The overhead when using a UDP is less than the TCP overhead on the order of 15%. This partially explains why the total delivery time of a transferred file is greater in our TCP VPN

implementation than in the UDP implementation, as shown in Figure 5.1. This supports our statement that a VPN implementation over UDP is more preferable than TCP.

Comparing the network overhead introduced in Veracity's network with a VPN over TCP implementation we calculate that the overhead doubles when using VPN. This means that with an MTU size of 1500 bytes and without VPN we can send 1460 bytes per IP packet, while with VPN over TCP only 1415 bytes (an increase in overhead of 120%) per packet, and with VPN over UDP 1427 bytes per IP packet (an increase in overhead of 88%). The difference between the default network implementation and the VPN implementation is due to our TCP segment being large enough that the VPN overhead is not significant, but the increase in fragmentation is large. This is the main reason behind the significant increase of total delivery time between the non-VPN and VPN implementation, as depicted in Figure 5.1.

In the case of using a VPN over UDP the number of packets sent from client to server increased by 7% when utilizing BF in CBC mode and 9% when utilizing AES 128bits in CBC mode (excluding initial handshake authentication) when compared with video transmission on Veracity's unsecured network. The increase in the total packets has only a slightly increase of overhead in our network, but the benefits of a secure communication outweigh this increased overhead.

A choice of different TCP segment size would have resulted in a better performance in our network, since our TCP segments with headers would fit into our UDP packets - while still avoiding fragmentation.

5.1.1.2 SSL/TLS handshake overhead

For server/client authentication and key establishment, SSL/TLS introduces a handshake protocol which delays the establishment of the connection between the client and the server and delays the start of a video transfer. The messages involved in the SSL/TLSv2 handshake are as follows:

Client		**Server**
Client Hello (empty SessionTicket extension)[2]	---------->	
		ServerHello (empty SessionTicket extension)[2] Certificate* ServerKeyExchange* CertificateRequest*
	<----------	ServerHelloDone
Certificate* ClientKeyExchange CertificateVerify* [ChangeCipherSpec] Finished	---------->	
		NewSessionTicket[2] [ChangeCipherSpec] Finished
Encrypted video stream transfer	<---------->	Encrypted acknowledgements

* Indicates optional or situation-dependent messages that are not always sent as defined in [40].

Figure 5.2: Message flow for our network's SSL/TLS handshake (Adapted from Figure 1 on page 4 of [40])

The analysis below derives from our observation of the packets exchanged between the COLDSTORE client and the COLDSTORE server while establishing a secure communication channel, when filtering out only OpenVPN traffic. All of the messages described below are encapsulated in a TCP segment (20 bytes of header), which is encapsulated in an IP datagram (20 bytes of header), and then encapsulated in an Ethernet frame (14 bytes of header).

The first message sent is the *ClientHello* message which consists of a 5 bytes TLS record header, 4 bytes of handshake header (SSL record header), and 47 bytes of content. The content of this message includes:

- version (2 bytes);
- random (32 bytes, 4 bytes for timestamp + 28 random);
- session id length (contains 0 value) (1 byte);
- one cipher suite supported (cipher cuites length, cipher suites) (2 + 1x2 bytes, as we support only one cipher suite per test scenario); and
- compression algorithm (compression length, compression algorithm) (1 + 1 byte, in our scenario these bytes are always null)

[2] A TLS mechanism that enables the server to resume clients' sessions without requiring maintaining session states per-client.

- extensions supported (extensions length, extension list) (2 bytes + 4 bytes). In our test scenario our extensions list includes only the *SessionTicket TLS* message which initiates a new SSL/TLS session. The message is empty as the client does not currently have any ticket from the server.

After the *ClientHello* message the server replies with several fragmented TCP segments that include *ServerHello*, server *Certificate*, *ServerKeyExchange*, *CertificateRequest* and *HelloDone* messages. Using the data from our Wireshark capture we could reconstruct the initial single message from the reassembled messages. In the next paragraphs we will describe each message in detail.

The *ServerHello* message consists of 5 bytes TLS record header, 4 bytes of handshake header, and 44 bytes of content. This message content includes the following:

- version (2 bytes);
- random (32 bytes, 4 bytes for timestamp + 28 bytes random);
- session id length (contains 0 value) (1 byte);
- selected cipher suite (2 bytes for selected cipher suite); and
- compression algorithm (1 bytes which contains null value in our scenarios)
- extensions supported (extensions length, extension list) (2 bytes + 4 bytes). The server also sends an empty *SessionTicket TLS* message to inform the client that it will send a new session ticket when the session details have been negotiated.

The *ServerHello* message is followed by the server certificate which must comply with the selected cipher suite. The *Certificate* message consists of the TLS record header and the handshake header, and 2,334 bytes of certificate. The certificate includes the signed certificate, an algorithm identifier (shaWithRSAEncryption), padding length which equals to 0, and the encrypted certificate.

In addition, the *ServerKeyExchange* message is sent, which is used by the clients to authenticate the server, as OpenVPN supports bidirectional authentication (as described in section 4.3.5.1). The message consists of 786 bytes including the TLS record header, the handshake protocol, and 777 bytes of DH parameters including p value, g value, and public key value and signature.

Finally, the server finishes its transaction with one TLS record message which encapsulates two handshake protocol messages. The first one is a *CertificateRequest* message where the server requests the client's certificate, which is followed by the *ServerHelloDone* message informing the client that the *ServerHello* phase has finished.

The client replies by sending *Certficate*, *ClientKeyExchange*, *CertificateVerify*, *ChangeCipherSpec*, and *Encrypted Handshake Message* in one large protocol data unit which is delivered fragmented into several TCP segments to the server.

The *Certificate* message, which is similar to the server's certificate message, includes the TLS record header and the handshake header, the certificate length, and 2,318 bytes of certificate. The *ClientKeyExchange* message includes the DH client parameters. The message's size is 267 bytes including 5 bytes of TLS record header, 4 bytes of handshake

header, 258 bytes of DH client parameters (a public key length and content). The client also sends a *CertificateVerify* message which includes a signed concatenation of all the exchanged handshake messages starting from *ClientHello* up to this point, without including the message. This message provides verification of the client's certificate and consists of 258 bytes, and a TLS record and handshake protocol header.

To continue, the client sends a *ChangeCipherSpec* message of fixed size (1 byte), together with 5 bytes of TLS record header, indicating to the server that the client uses the selected cipher suite as negotiated. The *ChangeCipherSpec* message is not part of the handshake protocol, but its existence in the handshake protocol helps in the efficient configuration of SSL/TLS. Finally, a *Finished* handshake message of 48 bytes is sent by the client to the server, which contains an encrypted hash of all the handshake messages. This is the first message that uses the negotiated cryptographic structures (i.e. algorithms, keys, secrets).

The server encapsulates the session details into a *New Session ticket* message which informs the client about the session ID of this established connection. This ID can be used to resume a previously established session, thus minimizing the SSL/TLS handshake overhead. The message consists of a session ticket lifetime, the session ticket length, and the session ticket content (a total of 1,366 bytes). A *ChangeCipherSpec* (1 byte) message and a *Finished* (48 bytes) message are also transmitted to the client. At this point both sides are ready to start transmitting the encrypted and authenticated video frames.

Now that we have a good picture of the messages exchanged during the SSL/TLS handshake protocol we can calculate the protocol's overhead. The total bytes exchanged during the SSL/TLS handshake are 7,739 bytes and the total overhead time is 0.165077 sec. Thus, we can conclude that SSL/TLS handshake protocol does not introduce significant delay in the transmission of the video files.

5.1.1.3 SSL/TLS session resumption

Client and server SSL/TLS handshake authentication occurs only once for the same client-server pair. For performance efficiency, in order to avoid the additional latency and computational cost of the SSL/TLS handshake protocol, SSL/TLS provides the ability to resume a session, once the session is established. During session resumption the negation of security parameters are omitted, thus saving significant time.

As a result, the messages involved in SSL/TLS resumption are as follows:

Client		**Server**
Client Hello (SessionTicket extension)[3]	---------->	
		ServerHello (empty SessionTicket extension)[3] [ChangeCipherSpec]
	<----------	Finished
[ChangeCipherSpec]		
Finished	---------->	
Encrypted video stream transfer	<---------->	Encrypted acknowledgements

Figure 5.3: Message flow for our network's SSL/TLS session resumption (Adapted from Figure 2 on page 5 of [40])

The *ClientHello* message in the SSL/TLS session resumption differs from the message sent on the SSL/TLS handshake protocol only in the session id field, where it contains 32 bytes for the session ID, and in the SessionTLS Ticket under the extensions list, where it contains 192 bytes for the resumed session ID. The *ServerHello* message also contains a 32 bytes session ID and an empty SessionTLS Ticket field. *ChangeCipherSpec* and *Finished* messages include the same fields as described in the SSL/TLS handshake protocol above. As a result, the total bytes exchanged during the SSL/TLS session resumption are 480 bytes, and the overhead time is 0.009611 sec. Thus, it took as only 5.8% of the time it took for the full handshake for a savings of 93.82%.

5.1.4 One-way delay

The measurement of one-way delay is an interesting characteristic for network performance analysis. One-way delay provides much more information than the round-trip time (RTT) as it gives additional information about the state of the network(s) and the application's performance. For network characterization, we extract transmission and propagation delays by looking at the delay and compare it to the minimum end-to-end delay. The variations in one-way delay are related to queuing effects. We can also use this information to infer network topology and congestion state or route changes. From an application performance point of view, if OpenVPN is implemented over TCP, large delays will hider sustaining a high bandwidth due to TCP's dependency on RTT [41].

The end-to-end delay can be divided into several components:

Transmission delay	Time needed to transmit the bits of a packet via a data link.
Propagation delay	Time needed to propagate a bit through the data link.
Processing delay	Time needed to process a packet in each router.
Queuing delay	Time needed to wait in a router queue before transmission.

[3] A TLS mechanism that enables the server to resume clients' sessions without requiring keeping session states per-client.

Based on our network topology there are negligible processing or queuing delays in our system as the two laptops are connected using a switch and the switch only receiving traffic from the two laptops and the NTP server. The NTP server sends a NTP packet per 2 minutes, which is considered negligible considering the network load. The one-way delay was measured by capturing packets at the two end-points, the server and the client. As the two laptops are synchronised to the same NTP server, we can accurately measure the time a packet leaves the source and when the packet reaches the destination.

Figures 5.4 to 5.8 depict the end-to-end delay measurements of the various test scenarios using histograms. The histogram shows the distribution of packets with regard to a one-way delay metric. The x-axis represents the one-way delay in milliseconds (ms) and the y-axis the percentage of packets that fell into a specific delay bucket. The histogram helps us to identify the maximum and minimum one-way network delay and the number of packets subject to different delay intervals.

Observation of these figures shows that most of the distributions display a typical Gamma-like distribution curve with a Gaussian tail (the green curve displayed in the figures) [42]. The gamma distribution belongs to the probability distributions family with two parameters. For the gamma curve the probability distribution function is:

$$f(x; \alpha, \beta) = \frac{1}{\beta^{\alpha}\,\Gamma(\alpha)}\; x^{\alpha-1}\; e^{\frac{x}{\beta}}$$

where the gamma function (Γ(α)) interpolates the following factorial:

$$\Gamma(\alpha) = (a-1)!$$

From the figures we understand that our packets mostly experience low one-way delays (the first high spike depicted in all figures), but we also have many packets that experience higher one-way delay times (the second short spike depicted in all figures). This might be due to the processing time effects when the packet code can run.

The scale parameter (β) for all the distribution curves in our different scenarios is between 0,5 and 1. The larger the scale parameter the wider the distribution, which in our case means the more variation we have in one-way delay times. The consistency of the β parameter in our scenarios depicts persistent congestion which in our network means no congestion as our traffic is transferred through a closed network, with no additional traffic being transferred, hence there should be no queuing effects from routers and the switching time of the switch used is 15μs per 64-byte frame.

The shape parameter (α) of our gamma distributions, on the other hand, displays significant fluctuations for the different security scheme. The shape parameter depicts the height of the peak in our measurements. The smaller the parameter, the higher the peak of the distribution. Between our VPN implementations the highest peak occurs for the test case of a VPN over TCP utilizing AES 128 bits in CBC mode algorithm, and the lowest in VPN over UDP when utilizing BF in CBC mode algorithm.

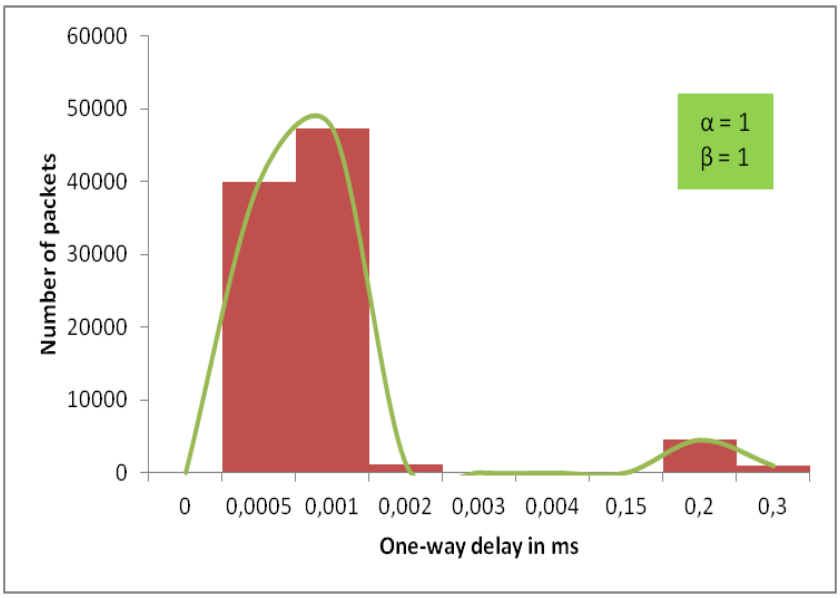

Figure 5.4: One-way delay histogram for simple network connection without a VPN

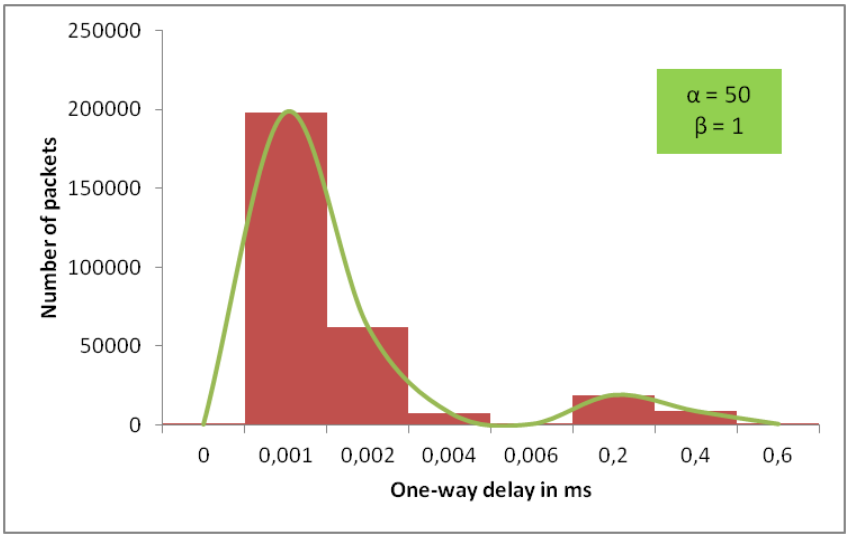

Figure 5.5: One way delay histogram for sending TCP packets using BF in CBC mode algorithm.

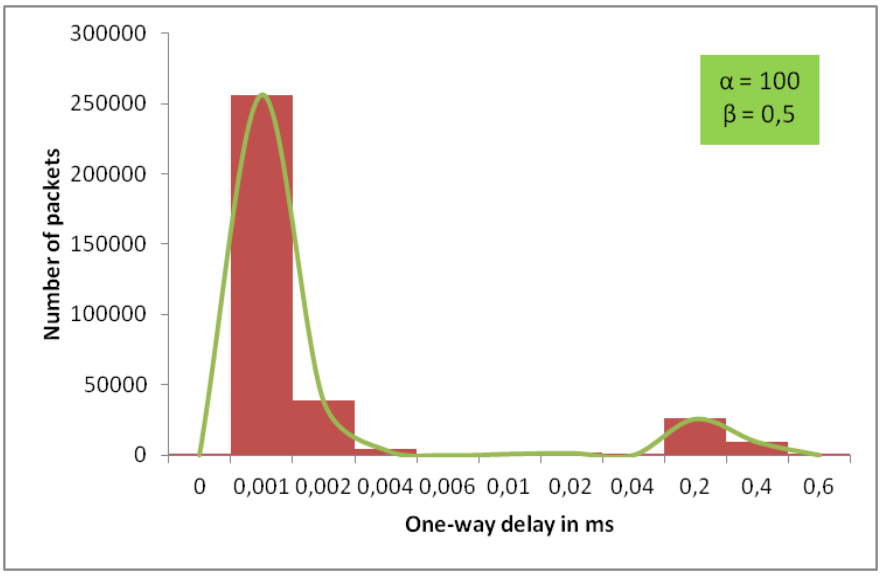

Figure 5.6: One-way delay histogram for sending TCP packets using AES 128 in CBC mode algorithm.

In this simple test network 42% of the packets are subject to low one-way delays of 0,0005 ms, while 50% experienced 0,001 ms delays. Only 5% of the packets experience delays of 0,2 and 0,3 ms. Figure 5.4 shows that 66% of the transferred data packets sent using TCP with BF in CBC mode experience 0,001 one-way delay while only 9% of the other packets are subject to delays of 0,2 between 0,4 ms. Lastly, sending the video data via UDP with AES 128 bit key algorithm on CBC mode with SHA256 hash functions results in 87% of packets experiencing a minimum delay of 0,0008 and 0,001 ms, while only 12% experienced delays of 0,2, 0,4 and 0,6 ms.

The aforementioned results display no major differences between the different tests. This is a useful outcome as it means that no matter what the computation differences between the different algorithms that were tested, our network adds little delay to the tunnelled packets.

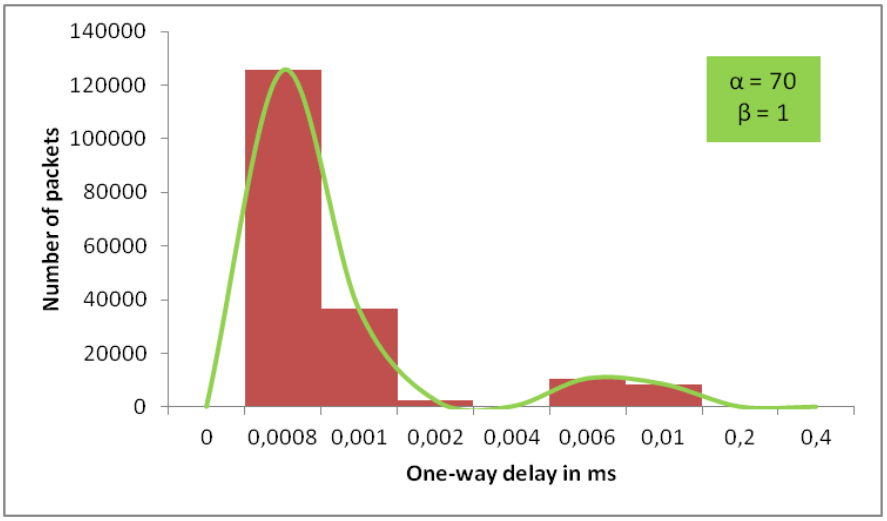

Figure 5.7: One-way delay histogram for sending UDP packets using default BF in CBC mode algorithm.

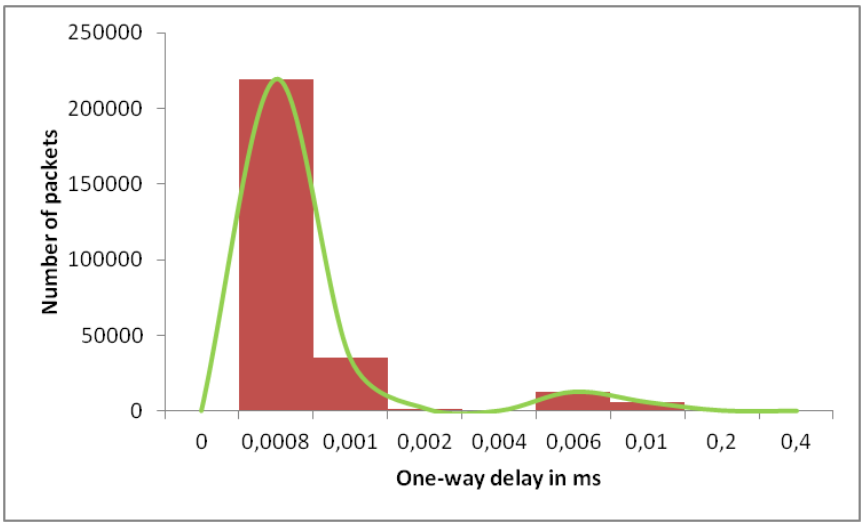

Figure 5.8: One-way delay histogram for sending UDP packets using AES 128 bits in CBC mode algorithm.

5.2 Certificate deployment

OpenVPN's server/client model makes the establishment and maintenance of a VPN network a fairly easy process. If we need to add a new camera to our network, we simply create a certificate for each new camera and add this certificate together with the CA's certificate to the camera's file system. Another useful functionality of the server/client model is that we do not have to worry about assigning an appropriate IP address in the private network for the VPN to the introduced camera as the server will automatically assign one from its assigned subnet. However, the IP camera still needs to get an address for the network to which it is physically connected. This can be achieved via DHCP or via a static address assignment.

If a replacement of a camera is required we can simply transfer the old camera's certificate to the new camera or generate a new certificate for the new device. The new camera's configuration file will be identical to the old ones so we can even use the previous configuration files with some slight changes regarding the stored certificate. Both the aforementioned scenarios can be implemented without interrupting the already running VPN tunnels. This means that the server can still smoothly collect video data from the rest of the network devices and that the time needed for certificate replacement is in the order of seconds. Specifically, the only new certificate being introduced is the new camera's certificate; hence only the new camera has to read its new certificate. In both cases, certificate reusability or creation of new certificate, a new SSL/TLS session must be established which will incur the overhead described in section 5.1.1.2. Taking into consideration the above, we consider that certificate replacement in our system has a linear. Expiration of certificates are expected after a certain amount of time, most commonly yearly, thus new certificates and new configurations of all the devices will be required when the certificates' validities expire.

Unfortunately, a VPN is not so flexible concerning server replacement. Although server changes are seldom there is always the need for device replacement. For this reason it is essential that we configure our IPcameras with multiple server entries. We can also utilize special reverse proxy devices, called the load balancers, which are responsible for decreasing the load on the servers and distribute the traffic over the servers. These is useful not only for load balancing, but also provides fault tolerance as the clients have a chance to select another server from the list if the active server becomes unreachable. However, use of load balancers is not currently supported by the company's clients or software.

Another important aspect of certificate deployment arises from the nature of video surveillance systems. Stored cipher streams must always be available to be provided to public and regulatory authorities in order for the stream to be played back on a different device than where it is stored. Fortunately, OpenVPN's end-to-end security adds no restrictions as the video stream is decrypted before being stored on the server side. To enhance our system's security it might be necessary to encrypt the video stream before storing it into the COLDSTORE storage device. We can do so using the COLDSTORE device's private key. As a result, if we need to decrypt the video stream on another device and playback the video stream the only thing we need to have is the COLDSTORE server's public key which we can obtain using a new SSL/TLS connection between the server and the playback device and sending a certificate request to the server.

Chapter 6

Conclusions and Future Work

In this section we describe our conclusions based upon the results that were discussed in the previous chapters. Some possible future work is also suggested as parts of planned work were not done in this project due to limited time and resources.

6.1 Conclusions

This thesis project investigated the legal framework of collecting, processing, and storing personal data gathered from video surveillance systems, with aim to propose a security scheme to address this legal framework for Veracity.

This project began by examining the rules and regulations that apply to video surveillance systems concerning the personal data protection that those systems should provide. EU and US federal regulations were investigated. Special consideration has been given to FIPS 140-2, as it includes a specification for security compliance of software and hardware products purchased by the US federal government. This FIPS publication has been accepted as standard for security schemes. Thus, we proposed a security scheme that could be FIPS compliant.

The result was a proposal to use the AES 128 bit algorithm in CBC mode for secret key encryption, while using RSA 2048 bit key certificates for session key establishment between network devices. SHA256 was selected as the hash function for signing video data packets for authentication, integrity, and non-repudiation purposes. The proposed scheme was implemented using FIPS compliant object module generated from the OpenSSL library, thus our security algorithms and operations utilized a validated FIPS implementation, and thus the resulting softwarecan be declared FIPS compliant.

OpenVPN was used to implement the proposed security design using tunnelling between the IP cameras/DVR/NVR (acting as clients) and the COLDSTORE device (acting as a server). OpenVPN is an open source user space application which provides cross-platform flexibility and together with the FIPS compliant module (described above) it is a FIPS compliant implementation. To test the performance of the proposed security we transferred a large video file from client to server using different configurations of the system in a series of test scenarios. These tests included transmission without OpenVPN, with OpenVPN using the default settings and algorithms, and finally using the proposed algorithms and additional SSL/TLS handshake authentication.

These tests were used to analyse the network's performance and efficiency. From the results of these tests, it is apparent that the data delivery of the video is fast enough to satisfy real time requirements, has VPN is not a bottleneck for video surveillance systems. Moreover, as our VPN implementation is a user space application, context switching and

memory copying might decrease the maximum throughput, thus our throughput is lower than might be the case for a kernel-based implementation. This means that OpenVPN is suitable for applications that have time critical security requirements, such as video surveillance systems. The encryption and authentication time per byte of video data transferred is similar for both BF and AES algorithm implementations of the VPN which motivates us to use the more secure AES as we need not be concerned about an increase in the encryption and authentication delay.

More careful consideration should have been taken in the choice of the TCP segment size of the COLDSTORE client application. Our network suffers from high TCP segment fragmentation, thus the total delivery time of the video file is higher than expected. This is also obvious from the number of files sent in the network with and without VPN utilization. Ideally, we should choose the TCP segment size considering the block size of the encryption algorithm and the size of the signed hash.

The VPN tunnel stacks introduced approximately 5% to 6% more network overhead in bytes than non-tunneled delivery. Moreover, the SSL/TLS packet overhead is negligible when using a large MTU size. The initial SSL/TLS handshake overhead takes only 100 ms to establish the session key and this establishment occurs only once when a new client or a new certificate is introduced to the network (as once the devices are attached to the network we assume that they continue to run continuously). So, we can conclude that the VPN protocol stack introduces negligible network delay, while providing quite high throughput (*even for an user space implementation*).

One way delay in each of the different test case scenarios reveal via the histograms, that there is a very low delay introduced by the encryption/decryption and authentication processing. Additionally, the packets experience low delay fluctuations, which makes our network implementation stable enough to transfer for the desired video (and perhaps audio) streams.

During this project we investigated the most relevant rules and regulations covering personal data and analysed the network performance of a video surveillance system with different OpenVPN encryption and signing settings. Although, we were unable to explore the details of our system's performance, we gathered important results concerning the use of OpenVPN as a user space application and the performance of different encryption and hashing algorithms. The next section suggests some future work based upon what was learned during the course of this thesis project.

6.2 Future work

Due to the obstacles that we faced when we tried to install the OpenSSL FIPS Object module for 32bit applications we were unable to integrate the OpenSSL library with the COLDSTORE library. Future work should complete this integration and then perform

measurements of its performance. This integrated implementation's performance should be compared with the performance observed in the test scenarios report in this thesis.

This thesis project only considered an OpenVPN implementation for securing our network communication. Additional, tests and measurements could be conducted to compare OpenVPN with its widely used competitor, the IPSec protocol.

Another useful area of research is the use of hardware accelerators. Hardware accelerators are supported by OpenSSL and can be used to run the computationally intensive hardware cryptographic functions. Most commonly available graphics processing units (GPUs) could be used to off-load this processing from the central processing unit [43].

Regarding the Ethernet technology due to restrictions regarding the Veracity network we only utilized 1500 bytes Ethernet frames. A Gigabit Ethernet network should be set up to send small frames and Jumbo frames in order to compare their performance and to explore the tunnelling limitations for other frame sizes [44].

6.3 Required reflections

This thesis project proposed a security scheme to be utilized to protect sensitive personal data obtained by video surveillance systems. The work conducted is expected to have an impact on the further development of security schemes integrated to video surveillance systems. Although the information that was stored and analysed in this thesis project was made publicly available by its authors and no personal data were stored or processed, we should be aware that the nature of video surveillance systems have a great impact on the socioeconomic and ethical aspects of our communities. For this reason, we tried to provide a security scheme that complies with the latest security standards not only in Europe, but also internationally; and to ensure that the data that are collected from Veracity's clients are fully protected from tampering, thus increasing the integrity of this data.

Additionally, by showing that it is both technically and economically feasible to use the proposed solution to protect the data end-to-end, we would expect that in the future all video surveillance systems would employ security at least as strong as that described in this thesis, thus increasing the protection of this sensitive personal data.

The conditions under which a company or organisation may collect and process video stream data obtained using Veracity's products must always follow the rules and laws set by the relevant governmental body responsible for protecting private information and human rights in the location(s) where this data collection and processing is taking place

.

References

[1] Frank Webster, *Theories of the Information Society*, 3rd ed. United Kingdom: Taylor & Francis Ltd - M.U.A, 2006, 312 pages, ISBN 0415406331, 978-0415406338.

[2] Fredrik Nilsson and Axis Communications, *Intelligent Network Video; Understanding Modern Video Surveillance Systems:* CRC Press, 2008.

[3] Zhaoyu Liu, Dichao Peng, Yuliang Zheng, and Jeffrey Liu, "Communication protection in IP-based video surveillance systems" presented at the Seventh IEEE International Symposium on Multimedia, 2005, pp. 69–78, DOI: 10.1109/ISM.2005.42.

[4] Jayraj Ugarkar, *The essentials of Telecommunications Management; A simple guide to understand a complex industry*, 1st ed., Bloomington, Indiana, USA: AuthorHouse, 2010.

[5] EU Parliament, "Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data," l. 281, pp. 0031 – 0050, Nov. 1995.

[6] Emily Harwood, *Digital CCTV : A Security Professional's Guide*. Burlington Elsevier, 2008, [Accessed: 20-Mar-2013].

[7] Sennewald, Charles A. and Christman, John H, *Retail crime, security, and loss prevention an encyclopedic reference*. Burlington, MA: Butterworth-Heinemann, 2008, [Accessed: 20-Mar-2013].

[8] "Protection of personal data - Justice," *European Commision*, 12-Mar-2013. [Online]. Available: http://ec.europa.eu/justice/data-protection/index_en.htm. [Accessed: 20-Mar-2013].

[9] "Data Protection Act 1998," legislation.gov.uk, 16-Jul-1998. [Online]. Available: http://www.legislation.gov.uk/ukpga/1998/29/contents. [Accessed: 23-Jan-2013].

[10] "Data Protection and Freedom of Information advice - ICO," Information Commisioner's Office. [Online]. Available: http://www.ico.gov.uk/. [Accessed: 23-Jan-2013].

[11] Parliament Office of Science and Technology, "CCTV." Parliament Office of Science and Technology, Apr-2002.

[12] Parliament Office of Science and Technology, "House of Lords - Science and Technology - Fifth Report," Parliament Office of Science and Technology,

Committee Publication Fifth report, Feb. 1998. Available: http://www.parliament.the-stationery-office.co.uk/pa/ld199798/ldselect/ldsctech/064v/st0504.htm

[13] British Standard Institution, "Standards, Training, Testing, Assessment and Certification | BSI Group." [Online]. Available: http://www.bsigroup.co.uk/. [Accessed: 20-Mar-2013].

[14] British Standard Institute, "BS 10012 Data protection. Specification for a personal information management system," bsi. shop, May-2009. [Online]. Available: http://shop.bsigroup.com/en/ProductDetail/?pid=000000000030175849. [Accessed: 23-Jan-2013].

[15] "NIST.gov - Computer Security Division - Computer Security Resource Center," *FIPS PUB 140-2*, 15-Nov-2001. [Online]. Available: http://csrc.nist.gov/groups/STM/cmvp/standards.html. [Accessed: 04-Feb-2013].

[16] Charlie Kaufman, Radia Perlman, and Mike Speciner, *Network Security: Private Communication in a Public World*, 2nd ed. Prentice Hall, 2002, ISBN13: 978-1-4503-1004-8 ; DOI: 10.1145/2046660.2046668

[17] Pravir Chandra, Matt Messier, and John Viega, *Network Security with OpenSSL*. O'Reilly, 2002, ISBN: 059600270X , 978-0596002701

[18] K. John Singh and R. Manimegalai, "A Survey on Joint Compression and Encryption Techniques for Video Data," *Journal of Computer Science*, vol. 8, no. 5, pp. 731–736, 2012.

[19] Xie Dahua and Kuo C-cjay, "Multimedia Encryption with Joint Randomized Entropy Coding and Rotation in Partitioned Bitstream," in *EURASIP Journal on Information Security*, vol. 2007(1), p. 35262, 2007, DOI: 10.1155/2007/35262.

[20] Fuwen Liu and Hartmut Koenig, "A survey of video encryption algorithms" in *Computers & Security*, vol. 29, no. Elsevier, pp. 3–15, Jun. 2009, DOI: 10.1016/j.cose.2009.06.004.

[21] Liu Fuwen and Koenig Hartmut, "A novel encryption algorithm for high resolution video," in *Proceedings of the international workshop, (NOSSDAV '05)*, New York, NY, USA, 2005, pp. 69–74.

[22] Internet volunteer community, "OpenSSL: The Open Source toolkit for SSL/TLS," *OpenSSL; Cryptography and SSL/TLS Toolkit*. [Online]. Available: http://www.openssl.org/. [Accessed: 24-Mar-2013].

[23] "Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program" National Institute of Standards and Technology (NIST) and Communications Security Establishment Canada (CSEC), 21-Dec-2012.

[24] OSF, “FAQ - The I.G. 9.5 Issue,” *OpenSSL Software Foundation, Inc.*, 14-Mar-2013. [Online]. Available: http://www.opensslfoundation.com/fips/ig95.html. [Accessed: 18-May-2013].

[25] Berry Hoekstra and Damir Musulin, “Comparing TCP performance of tunneled and non-tunneled traffic using OpenVPN,” Universiteit Van Amsterdam, System & Network Engineering, Amsterdam, Aug. 2011 Available: http://staff.science.uva.nl/~delaat/rp/2010-2011/p09/report.pdf.

[26] Muhammad Siraj Rathore, Adil Razzaq, Peter Sjödin, Markus Hidell, and Björn Pehrson, “Site-to-Site VPN Technologies : A Survey,” KTH, Telecommunication Systems Laboratory, TSLab, Stockholm, Sweden, 2009 Available: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-72509.

[27] ‘OpenVPN - OpenVPN Community Software - Does OpenVPN support IPSec or PPTP?’ [Online]. Available: http://openvpn.net/index.php/open-source/339-why-ssl-vpn.html. [Accessed: 03-Apr-2013]

[28] Praveen Likhar, Ravi Shankar Yadav, and Keshava Rao M, “Securing IEEE 802.11g WLAN Using Open VPN and its Impact Analysis,” vol. 3, no. 6, pp. 97–113, 2011.

[29] “OpenVPN - Open Source VPN,” 2002. [Online]. Available: http://openvpn.net/index.php/. [Accessed: 19-May-2013].

[30] Maxim Krasnyansky and Bishop Clark, “Universal TUN/TAP driver,” *Vtun - tun/tap devices*, 1999. [Online]. Available: http://vtun.sourceforge.net/tun/. [Accessed: 19-May-2013].

[31] ‘ProSafe 8-port Gigabit Smart Switch GS108Tv2 Data Sheet’, 2009. [Online]. Available: zotero://report/items/0_RBHG62B4/html/report.html. [Accessed: 09-Mar-2014]

[32] “Big Buck Bunny,” *Big Buck Bunny*, Apr-2008. [Online]. Available: http://www.bigbuckbunny.org/index.php/about/index.php. [Accessed: 20-May-2013].

[33] Lenstra, Arjen K. and Verheul, Eric R., ‘Selecting Cryptographic Key Sizes’, *Journal of Cryptology*, vol. 14 (4), pp. 255–293, August 2001, DOI:10.1007/s00145-001-0009-4. [Accessed: 20-March-2013].

[34] Dong Hoon Lee and Xiaoyun Wang, *Advances in Cryptology – ASIACRYPT 2011*. International Association for Cryptologic Research, 2011, ISBN: ISBN3-642-25384-9. [Accessed: 22-March-2013].

[35] “The GNU Privacy Guard - GnuPG.org,” 20-Dec-2012. [Online]. Available: http://www.gnupg.org/. [Accessed: 24-Feb-2013].

[36] OpenSSL Software Foundation, ‘User Guide for the OpenSSL FIPS Object Module v2.0 (including v2.0.1, v2.0.2)’. OpenSSL, 25-January-2013, Available: http://www.openssl.org/docs/fips/UserGuide-2.0.pdf. [Accessed: 19-Feb-2013].

[37] "OpenVPN - Open Source VPN - HOWTO." [Online]. Available: http://openvpn.net/index.php/open-source/documentation/howto.html#quick. [Accessed: 03-Apr-2013].

[38] 'Wireshark: A network protocol analyzer'. [Online]. Available: http://www.wireshark.org/. [Accessed: 16-Sep-2013].

[39] Osamu Honda, Hiroyuki Ohsaki, Makoto Imase, Mika Ishizuka, and Junichi Murayama, "Understanding TCP over TCP: effects of TCP tunneling on end-to-end throughput and latency", presented at the Performance, Quality of Service, and Control of Next-Generation Communication and Sensor Networks III, Boston, MA, 2005, vol. 1, pp. 60110H1–60110H-9 [Online]. Available: http://proceedings.spiedigitallibrary.org.focus.lib.kth.se/proceeding.aspx?articleid=772018

[40] J. Salowey, H. Zhou, P. Eronen, and H. Tschofenig, 'RFC 5077 - Transport Layer Security (TLS) Session Resumption without Server-Side State', *Internet Request for Comments*, Jan-2008. [Online]. Available: http://www.ietf.org/rfc/rfc5077.txt. [Accessed: 06-Oct-2013]

[41] Ren Wang, Giovanni Pau, Kenshin Yamada, M.Y. Sanadidi, and Mario Gerla, 'TCP startup performance in large bandwidth networks', presented at the INFOCOM 2004: Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies, Hong Kong, 2004, vol. 2, pp. 796–805. DOI:10.1109/INFCOM.2004.1356968

[42] A. Hernandez and E. Magana, 'One-way delay measurement and characterization', in *Networking and Service*, Athens, 2007, pp. 114–120 [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4438363. [Accessed: 16-Mar-2014]

[43] Mohamed Khalil-Hani, Vishnu P. Nambiar, and M. N. Marsono, 'Hardware Acceleration of OpenSSL Cryptographic Functions for High-Performance Internet Security', presented at the International Conference on Intelligent Systems, Modelling and Simulation, Liverpool, United Kingdom, 2010, pp. 374–379 [Online]. Available: http://ieeexplore.ieee.org.focus.lib.kth.se/stamp/stamp.jsp?tp=&arnumber=5416065

[44] Mauricio Tsugawa and José A. B. Fortes, 'Characterizing user-level network virtualization: performance, overheads and limits', presented at the International Journal of Network Management, 2010, vol. 20, pp. 149–166 [Online]. Available: http://onlinelibrary.wiley.com.focus.lib.kth.se/doi/10.1002/nem.733/abstract

Appendix A

Total Delivery time to transfer a video file

Total Delivery time to transfer a video file in seconds (secs) for the different test scenarios

	Transport protocol	
	TCP	UDP
No VPN	33,51	0
VPN utilizing Blowfish with SHA1 (default)	270,412	97,46
VPN utilizing Blowfish with SHA1 & Handshake Authentication	281,52	129,99
VPN utilizing AES_128_CBC with SHA256	311,03	121,46
VPN utilizing AES_128_CBC with SHA256 & Handshake Authentication	337,82	141,07

TRITA-ICT-EX-2014:6

www.kth.se