



Design and Implementation of Multilevel Security Architectures

PhD Candidate:

Angelo Liguori

First PhD Supervisor:

Prof. Dr. Gaetano Giunta

Second PhD Supervisor:

Prof. Dr. Francesco Benedetto

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy in the

DOCTORAL SCHOOL IN ENGINEERING

"Biomedical, electronics, electromagnetics and telecommunications"

UNIVERSITY OF ROMA TRE

Engineering Department

May 2016

Declaration of Authorship

I, ANGELO LIGUORI, declare that the thesis titled, ‘DESIGN AND IMPLEMENTATION OF MULTILEVEL SECURITY ARCHITECTURES’ and the work presented in it are my own.

I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University.
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
3. Where I have consulted the published work of others, this is always clearly attributed.
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
5. I have acknowledged all main sources of help.
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.
7. Parts of this work have been published before submission as the papers listed below:
 - A. Liguori. A novel Multiple Independent Levels of Security/Safety Cross Domain Solution. In IEEE Military Communications Conference, MILCOM 2015, pages 1578-1583, October 2015
 - F. Benedetto, G. Giunta, A. Liguori, and A. Wacker. A novel method for securing critical infrastructures by detecting hidden flows of data. In IEEE Communications and Network Security (CNS), pages 648-654, September 2015

- A. Liguori, F. Benedetto, G. Giunta, N. Kopal, and A. Wacker. Analysis and monitoring of hidden TCP traffic based on an open-source covert timing channel. In IEEE Communications and Network Security (CNS), pages 667-674, September 2015
- A. Liguori, F. Benedetto, G. Giunta, N. Kopal, and A. Wacker. SoftGap: A Multi Independent Levels of Security Cross-Domain Solution. In IEEE Future Internet of Things and Cloud (FiCloud), pages 754-759, August 2015
- A. Liguori. From Multilevel Security to MILS: the Evolution illustrated through a Novel Cross-Domain Architecture. In International Journal of Mobile Network Design and Innovation, Inderscience Publishers, FORTH-COMING

Date:

Signature:

“Numquam autem invenietur, si contenti fuerimus inventis. Praeterea qui alium sequitur invenit, immo nec quaerit. Quid ergo? Non ibo per priorum vestigia? Ego vero utar via vetere, sed si propiorem planioremque invenero, hanc muniam. Qui ante nos ista moverunt non domini nostri sed duces sunt. Patet omnibus veritas; nondum est occupata; multum ex illa etiam futuris relictum est¹”

Lucio Anneo Seneca, *‘Epistulae morales ad Lucilium’* 33.10-11

¹Nothing will be discovered if we rest contented with discoveries already made. Besides, he who follows another not only discovers nothing but is not even investigating. What then? Shall I not follow in the footsteps of my predecessors? I will indeed use the old path, but if I will find one that makes a shorter cut and is smoother to travel, I will open the new path. Those who have raised these problems before us are not our masters, but our guides. Truth shows itself to everybody; it has not yet been laid claim. And there is plenty of it left even for posterity to discover.

UNIVERSITY OF ROMA TRE

Engineering Department

DOCTORAL SCHOOL IN ENGINEERING

"Biomedical, electronics, electromagnetics and telecommunications"

Design and Implementation of Multilevel Security Architectures

Abstract

The problem of securely storing and processing sensitive data is paramount in many sectors. But ICT Security is not a Defense prerogative. Events like Vatileaks and Panama Papers, after Snowden's disclosures, brought to prominence the information security problem. We live in a world that requires from us to be always-on, always-connected. Technological progress and the need to process and share always bigger amount of data led to the inception of distributed systems, smart sensors/networks, cloud computing etc. and transformed the Internet in a high bandwidth medium. We are surrounded by devices and use applications that track and collect our personal information that should be properly protected. Sensitive data should be accessed only by people with valid authorizations and with a specific need-to-know that could affect only a specific subset of data necessary to perform some operations. The requirement to protect information characterized by a hierarchy of sensitivity levels led to the definition of Multilevel Security. In the last years a new paradigm called Multiple Independent Levels of Security/Safety (MILS) seems to be able to effectively address the problem.

This thesis illustrates the design and the implementation of Multilevel Security Architectures. We pinpointed the drawbacks of the currently proposed solutions and analyzed the problem from different perspectives: high-assurance security requirements, certification according to international schemes, and performance. We proposed an innovative MILS Distributed Architecture and implemented a specific MILS component that enforces the security policy of connecting domains characterized by different classification level information. We also faced the multilevel-related covert channel problem proposing a novel detection algorithm and building an open source covert timing channel in order to evaluate its performance.

Acknowledgements

I would like to thank the **Signal Processing for Telecommunications and Economics (SP4TE)** group of the University Roma Tre and the **Applied Information Security (AIS)** group of University of Kassel for their insightful advice and their unfailing support.

I would also thank the **German Academic Exchange Service (DAAD)** for the six-months grant provided.

Last but not least I would like to sincerely thank **SIRA s.r.l.** without which I could have never attended the doctoral school I was enrolled in without scholarship.

Contents

Declaration of Authorship	i
Abstract	iv
Acknowledgements	v
List of Figures	x
List of Tables	xi
Acronyms & Abbreviations	xii
1 INTRODUCTION	1
1.1 The Multilevel Security Problem	1
1.2 MLS Applications	3
1.3 Security Certification Schemes	4
1.4 Covert Channel Problem	5
1.5 Drawbacks of Current Solutions	6
1.6 Outline of the Thesis	6
2 FROM MULTILEVEL SECURITY TO MILS	8
2.1 Security Models	10
2.1.1 Bell-La Padula	10
2.1.2 Biba	11
2.1.3 RBAC	12
2.1.4 Clark-Wilson	12
2.1.5 Chinese Wall	13
2.2 Related Work	13
2.3 State of the Art of Classic MLS	14
2.3.1 Operating Systems	14
2.3.2 Databases	16
2.3.3 Virtualization	16
2.4 Cross-Domain Solutions	16
2.4.0.1 MLS Guards	17
2.4.0.2 Air-Gap	18

2.4.0.3	Data-Diode	18
2.5	Drawbacks of the classical approach	19
2.6	Multiple Independent Levels of Security/Safety	21
2.6.1	NEAT Paradigm	23
2.7	State of the Art of MILS Solutions	23
2.7.0.1	Separation Kernel Hypervisors	24
2.7.0.2	Separation Kernel Operating Systems	25
2.8	Security Evaluation Criteria	26
2.8.1	ISO/IEC 15408	28
2.8.1.1	Security Evaluation Phases	28
2.8.1.2	Evaluation Assurance Levels	29
2.8.1.3	Compositional Approach	31
2.9	MILS Projects	32
3	NOVEL MILS ARCHITECTURES	33
3.1	Proposed MILS Distributed Architecture	34
3.1.1	MILS Architecture Components	36
3.1.1.1	Trusted Front End	36
3.1.1.2	Policy Server	37
3.1.1.3	Transitional Secure Server	37
3.1.1.4	MILS Yarn Trusted Host	37
3.1.1.5	SoftGap	38
3.1.1.6	Application Servers	38
3.1.2	Security Requirements	38
3.1.3	MILS Architecture Use Cases	41
3.1.3.1	Unidirectional Secure Import from Unreliable Network	42
3.1.3.2	Secure Download from Internal File Server	44
3.1.3.3	Hybrid Security Requirements Correlation	46
3.1.4	Cipher Suites	47
3.2	Evaluation	49
3.2.1	Security	50
3.2.1.1	System Model	50
3.2.1.2	Attack Model	51
3.2.1.3	Security Analysis	51
3.2.2	Performance	53
3.2.2.1	Message Overhead	53
3.2.2.2	IKE and TLS Message Overhead	55
3.2.2.3	Further Overhead Elements	57
3.2.2.4	Message Size Overhead	62
3.3	SOFTGAP: a novel MILS Cross-Domain solution	66
3.3.1	SoftGap Architecture	67
3.3.2	System Model	68
3.3.3	Attack Model	69
3.3.4	Security Enforcing Functions Details	70
3.3.5	Sequence of Operations	70
3.3.6	Design Approach Rationale	73
3.3.7	Security Analysis	73

3.3.8	Security Evaluation Considerations	75
4	COVERT CHANNEL DETECTION	80
4.1	The Prisoners' Problem	80
4.2	Covert Channel Definition and Taxonomy	82
4.2.1	Covert Storage Channels	83
4.2.2	Covert Timing Channels	86
4.3	Detection Algorithms	88
4.3.1	Covert Storage Channel Detection Techniques	89
4.3.2	Covert Timing Channel Detection Techniques	90
4.4	Novel Covert Timing Channel Detection Algorithm	93
4.4.1	System Model	93
4.4.2	The Weibull-ness Test	94
4.4.3	Performance Analysis and Numerical Results	96
4.5	Open Source Covert Timing Channel	99
4.5.1	Manchester Coding	103
4.5.2	Hamming Code (12,8)	103
4.5.3	Development and Target environment	104
4.5.3.1	Sender	105
4.5.3.2	Receivers	106
4.5.3.3	Data test	106
4.5.4	Empirical results	106
4.5.4.1	Network conditions	107
4.5.4.2	Covert bit interval time	109
5	CONCLUSIONS	110
5.1	Summary of Contributions	110
5.2	Lessons Learned	112
5.3	Open Challenges	113
5.4	List of Publications	114
5.4.1	International Journals	114
5.4.2	International Conferences	114
5.4.3	Books	115
A	Covert Timing Channel Source Code	116
A.1	Simple OSCTC	116
A.1.1	Simple OSCTC Client	116
A.1.2	Simple OSCTC Server	120
A.2	OSCTC with Manchester Coding	125
A.2.1	OSCTC with Manchester Client	125
A.2.2	OSCTC with Manchester Server	129
A.3	OSCTC Manchester + Hamming	134
A.3.1	OSCTC Manchester + Hamming Client	134
A.3.2	OSCTC Manchester + Hamming Server	134
A.4	Passive OSCTC	135
A.4.1	Passive OSCTC Client	135

A.4.2	Passive OSCTC Server	135
B	NSA Cipher Suites	140
C	Weibull Probability Density Function	142
C.1	Weibull Distribution	142
	Bibliography	145

List of Figures

1.1	Range of costs for Common Criteria evaluation at EAL2, EAL3, and EAL4 [1]	5
2.1	Security Certification Report for Integrity-178B Operating System	29
3.1	Novel Distributed MILS Architecture	35
3.2	Functional Class Structure	38
3.3	Functional Family Structure	39
3.4	Security Functional Components Structure	39
3.5	Performance in Mbps for an IPsec Tunnel with AES-NI	58
3.6	Minimum bits of security comparison	59
3.7	Connection latency with communication within a domain	60
3.8	Throughput with and without IPsec with a single TCP connection	61
3.9	Performance of IPsec tunnel at different packet size	62
3.10	Performance of IPsec hardware accelerated implementations	62
3.11	TLS Packet Structure	63
3.12	SoftGap Architecture	68
3.13	SoftGap Sequence of Operations	78
3.14	LSS sequence diagram	79
3.15	HSS sequence diagram	79
4.1	Prisoners' Problem: Alice and Bob represent hosts that exchange data through a channel that is hidden inside licit communications. Warden can read, drop and manipulate the licit communication.	81
4.2	System Model: Alice and Bob could be for example a client requesting services and an internal Web Server replying to these requests.	81
4.3	Weibull Probability Density Function	94
4.4	P_D of Weibull-ness and chi-square tests	99
4.5	Web Browser downloading the picture used by our passive OSCTC	105
4.6	Covert message error rate percentage at different TIME_SLOT in Kassel network	108
4.7	Covert message error rate percentage at different TIME_SLOT in path Kassel - Duisburg	108
4.8	Covert message error rate percentage at different TIME_SLOT in path Kassel - Rome	108
4.9	Characters-rate in different scenarios	109
C.1	Exponential Probability Density Function	143
C.2	Rayleigh Probability Density Function	143

List of Tables

2.1	Classic MLS technology overview	19
2.2	Shortcomings of classic MLS approach	20
2.3	MILS features and solutions overcoming the classic MLS drawbacks	23
2.4	MILS technology overview	26
2.5	Correspondence of Certification Levels	27
3.1	Hybrid Security Requirements Correlation Matrix	47
3.2	Comparison between ECDSA and RSA signature	59
3.3	Sequence of operations	71
4.1	P_D of the analyzed methods for a fixed $P_{FA} = 10^{-2}$ and small case	98
4.2	P_D of the analyzed methods for a fixed $P_{FA} = 10^{-2}$ and medium case	98
4.3	P_D of the analyzed methods for a fixed $P_{FA} = 10^{-2}$ and large case	98
4.4	Distance (hops) and RTT value (ms) from the sender	106
B.1	NSA Cipher Suite A	141
B.2	NSA Cipher Suite B	141

Acronyms & Abbreviations

ABI	A pplication B inary I nterface
ACPII	A dvanced C onfiguration and P ower I nterface
AEAD	A uthenticated E ncryption with A ssociated D ata
AES	A dvanced E ncryption S tandard
AES-NI	A dvanced E ncryption S tandard - N ew I nstructions
AKA	A lso K nown A s
CA	C ertification A uthority
CAP	C omposed A ssurance P ackage
CCE	C orrected C onditional E ntropy
CCRA	C ommon C riteria R ecognition A greement
CDS	C ross- D omain S olution
CIA	C onfidentiality I ntegrity A vailability
COTS	C ommercial O ff- T he- S helf
cPP	collaborative P rotection P rofile
CPU	C entral P rocessing U nit
CS	C ipher S uite
CSC	C overt S torage C hannel
CSMA/CD	C arrier S ense M ultiple A ccess/ C ollision D etection
CSRC	C omputer S ecurity R esource C enter
CTC	C overt T iming C hannel
CTPEC	C anadian T rusted P roduct E valuation C riteria
CTS	C lear T o S end
DAC	D iscretionary A ccess C ontrol
DB	D ata B ase
DH	D iffie H ellman

DBMS	D ata B ase M anagement S ystem
DMA	D irect M emory A ccess
DNS	D omain N ame S ystem
DoD	D epartment of D efense
DSA	D igital S ignature A lgorithm
EAL	E valuation A ssurance L evel
ECC	E lliptic C urve C ryptography
ECDHE	E lliptic C urve D iffie H ellman E phemeral
ECDSA	E lliptic C urve D igital S ignature A lgorithm
ESP	E ncapsulation S ecurity P ayload
FLASK	FL ux A dvanced S ecurity K ernel
FO	F ragment O ffset
FPGA	F ield P rogrammable G ate A rray
GCM	G alois C ounter M ode
GOTS	G overnment O ff- T he- S helf
HSM	H ardware S ecurity M odule
HSS	H igh S ide S ubject
HTTP	H yper T ext T ransfer P rotocol
K-S	K olmogorov- S mirnov
IAR	I mpact & A nalysis R eport
ICMP	I nternet C ontrol M essage P rotocol
ICT	I nformation & T elecommunications T echnology
ID	I Dentification
I&A	I nformation & A uthentication
IEC	I nternational E lectrotechnical C ommission
IEEE	I nstitute of E lectrical and E lectronics E ngineers
IEG	I nformation E xchange G ateway
IKE	I nternet K ey E xchange
IP	I nternet P rotocol
IPD	I nter P acket D elay
ISN	I nitial S equences N umber
ISO	I nternational O rganization for S tandardization
ITA	I nternet T echnologies and A pplications

ITSEC	I nformation T echnology S ecurity E valuation C riteria
LAN	L ocal A rea N etwork
LSS	L ow S ide S ubject
MAC	M andatory A ccess C ontrol
MIC	M andatory I ntegrity C ontrol
MILS	M ultiple I ndependent L evels of S ecurity/ S afety
MCDS	M iniaturized C ross D omain S olution
MYTH	M ILS Y arn T rusted H ost
MLS	M ulti L evel S ecurity
MR	M edian R ank
MSL	M ultiple S ingle L evels
MTU	M aximum T ransmission U nit
NATO	N orth A tlantic T reaty of O rganization
NBS	N ational B ureau of S tandards
NEAT	N ot-bypssable E valuatable A lways-invoked T amperproof
NICTA	N ational I nformation C ommunications T echnology A ustralia
NIST	N ational I nstitute of S tandards and T echnology
NOOP	N O O Peration
NSA	N ational S ecurity A gency
NSS	N ational S ecurity S ystem
OS	O perating S ystem
OSCTC	O pen S ource C overt T iming C hannel
PDA	P ersonal D igital A ssistant
PDF	P robability D ensity F unction
PKI	P ublic K ey I nfrastructure
PP	P rotection P rofile
PS	P olicy S erver
RFC	R equest F or C omment
RM	R eference M onitor
RTCP	R eal T ime C ontrol P rotocol
RTOS	R eal T ime O perating S ystem
RTP	R eal-time T ransport P rotocol
RTS	R equest T o S end

SCOMP	Secure COMmunications P rocessor
SA	Security A ssociation
SE	Security E nhanced
SEF	Security E nforcing F unction
SFC	Security F unctional C omponent
SFR	Security F unctional R equirement
SKH	S eparation K ernel H ypervisor
SKPP	S eparation K ernel P rotection P rofile
SLS	S ingle L evel S ecure
ST	S ecurity T arget
SWaP	S ize W eight and P ower
TCB	T rusted C omputing B ase
TCP	T ransmission C ontrol P rotocol
TCSEC	T rusted C omputer S ecurity E valuation C riteria
TFE	T rusted F ront E nd
TLS	T ransport L ayer S ecurity
TOS	T ype O f S ervice
TSS	T ransitional S ecure S erver
TTL	T ime T o L ive
UCDMO	U nified C ross- D omain M anagement O ffice
UDP	U ser D atagram P rotocol
U.S.	U nited S tates
VM	V irtual M achine
VoIP	V oice o ver I P
VPN	V irtual P rivate N etwork
WLAN	W ireless L AN
XML	e Xtensible M arkup L anguage
XSM	X en S ecurity M odule

*Dedicated to those who, despite of the age,
don't give up staying hungry and foolish.*

Chapter 1

INTRODUCTION

HISTORICALLY, organizations categorize information according to its level of sensitivity, its value, and the impact that its disclosure, alteration, and destruction without authorization could cause. Data classification standards can be considered the starting point for any security initiative. After information is created, modified or received, its sensitivity level needs to be defined. Such classification is useful to determine the basic security controls necessary for data protection and the enforcement of the established Security Policies.

Moreover, the evolution of electronic devices and telecommunications infrastructures, together with the widespread availability of the Internet, has deeply changed the way information is created, exchanged and stored. Nowadays data are produced massively and faster than previously and their value often resides in the speed they are shared and made available to those requiring them. Such set of events changed the data protection requirements, that evolved according to the new threats and attacks of the cyber-security era.

1.1 The Multilevel Security Problem

The strength and competitiveness of commercial and governmental organizations, regardless of their specific concern, is directly connected to the information they operate with, and to the reliability they can place in security systems designed for data protection. More and more sectors require high-robustness, high-reliability and high-assurance security systems. Defense is the paramount actor in the Information & Communications Technology (ICT) security, particularly nowadays that its activities are moving from the

classical battlefield towards the cyber-defence concept. The main goal is to protect information that is relevant for the national security of a country.

In the military environment data are labeled according to a classification scheme that changes from a Government to another and they are subject to well-defined security models. Such models shoot for enforcing different security properties: Bell-La Padula is involved in the protection of data confidentiality, whereas Biba's goal is the protection of data integrity. According to the former model, for example, given a software process with clearance Secret¹, it is granted to read files characterized by a classification label equal or below Secret, whereas it cannot write any data on files with a lower classification level to avoid disclosure of Secret information. For the very same reason the process can write on a Top Secret file but cannot read it. Actually the situation is even more complicated by the presence of compartments, the user need-to-know, the difficulty of developing effective security solutions in which Bell-La Padula and Biba models can co-exist and so on. Sometimes the military context shares resources with the civilian world in specific applications. Cosmo-SkyMED² [2] is an example of aerospace program co-financed by military and civilian agencies (it belongs to the so-called dual-use systems). In this case it is important to protect sensitive information on which users belonging to these different contexts operate, and that is characterized by various levels of classification. Often satellite programs are financed by a joint participation of countries (e.g. the MUSIS³ program [3]) thus, it becomes mandatory to enforce the National Eyes Only requirement stating that only personnel belonging to a specific country is granted access to the data of interest for that given country.

Critical infrastructures like chemical, communications, dams, emergency services, energy, public health, transportation, water, waste-water and financial service systems represent other sectors, often interconnected, demanding very high ICT security solutions.

Defense, Aerospace, Critical Infrastructures, Banks are contexts that do not admit any information leakage and cannot even easily recover from such events, compared with other commercial and industrial realities. For this reason the defense community in particular continually demands solutions whose security requirements are not met by commercial standards.

The need to protect information with different level of sensitivity led to the definition of a new design approach called Multilevel Security (MLS). The adjective "multilevel"

¹For the sake of clarity and given the fact that military information classification scheme calls immediately to mind the categorization concept, the military context is used as reference environment in the thesis. The standard hierarchy usually involves the unclassified, confidential, secret and top secret levels even if arbitrary lattices can be considered.

²The **CO**nstellation of small **S**atellites for **M**editerranean basin **O**bservation (COSMO-SkyMed) is a French-Italian satellite constellation for Earth observation for dual use operations (military and civilian users).

³The **MU**ltinational **S**pace-based **I**maging **S**ystem (MUSIS) is a spatial imaging system for Defense and Security financed by France, Italy, Belgium, Germany, Greece, and Spain.

has a double meaning: it refers to the levels information is classified with, and to the level of trust (clearance) given to authorized users by an authority responsible for establishing their trustworthiness. The Multilevel Security problem itself is characterized by different issues that overlap and intersect each other on different layers. An MLS system must not only protect a hierarchy of data from access by users that do not have sufficient clearances and need-to-know. It should deny any illicit data flow trying to subvert the established security policies by direct or indirect information and/or resources observation. It should guarantee that, in time-shared systems, no disclosure of information can derive from data left on the computer memory between different security level computations, and no interference between different security level operations can occur.

1.2 MLS Applications

There are many applications of MLS systems that can be profitably used in the aforementioned contexts. Such applications respond to the evolution of computers and telecommunications which, in turn, changed the security needs of their users. From small networks composed by separate systems dedicated to specific applications and specific types/levels of information, the ICT trend has moved towards the network of networks concept. Such evolution, pushed by technology improvements and the increasing need of information sharing, led to new security problems to be solved and a different approach to those faced theretofore with the use of stand-alone computers and duplication of networks.

The MLS applications can be divided in the following main groups:

- **Secure Consoles**
- **Secure Database Management Systems (DBMS)**
- **Cross-Domain Solutions (CDS)**

A Secure Console is a specific machine that allows authorized users to share its resources in unilevel and multilevel operations [4]. In unilevel applications a computer is time-shared by users to process data belonging to different classification levels. This solution increases the machine use and reduces hardware costs. In multilevel applications the MLS console is used to process several classification levels at the same time. It normally shows data in different windows, each containing only data of a specific classification level.

An MLS DBMS, unlike classical database management systems, is designed to manage a hierarchy of data enforcing a Mandatory Access Control (MAC) based upon the chosen

classification scheme. Moreover, the Security Policy can not be subverted or bypassed in any manner. An MLS DBMS normally uses labels to discriminate data belonging to different classification levels with a granularity than can go down to compartments, sub-compartments and so on.

The Cross-Domain Solution group encompasses all those systems that “provide the ability to *access* or *transfer* data among security domains” [5]. This is an important research field, whose significance increased over the last years together with the complexity of networks and the need to share information across different operational boundaries. A CDS, type-access, usually runs different operating system instances, each connected to a different network, and it normally shows data in different windows, each containing only data of a specific classification level (thick-client system). Currently, a different solution foresees thin-clients connected with a server that provides the secure connection to different classification level networks [6]. Cross-Domain Solutions, type-transfer, are instead used to move data between different classification level domains. They can be designed to enforce unidirectional or bidirectional data flow control security policies. Such devices in particular are a NATO target in the concept of Information Exchange Gateway (IEG). An IEG will enable effective NATO information exchange within and between NATO and National Consultation, Command and Control (C3) systems [7].

1.3 Security Certification Schemes

High-assurance systems used in the aforementioned applications require a formal certification establishing the level of confidence a user can have on the security functionalities claimed by such systems. Security certifications require a recognized trusted third-party that operates in compliance with a security certification methodology and with the assistance of an accredited laboratory to perform the evaluation activities. If the system/product under evaluation fulfills the required evidences, then a security certificate is released by the Certification Authority. Usually a system is evaluated according to specific security components and classes of assurance, that set up the level of certification the system is certified at. According to the specific security evaluation criteria, such levels can differ in nomenclature and/or number and often it is difficult to determine a precise correspondence between levels.

Traditionally the certification sponsor, normally the system vendor, provides the evidences that the system behaves correctly and enforces the desired security properties. To get such proofs the vendor or the system developer performs different activities, e.g. independent functional testing, code inspection, semi-formal and formal modeling, but also analysis and checking of processes and procedures, analysis of vulnerabilities and penetration testing.

The laboratory in charge of the evaluation verifies such evidences performing their own tests, verifying the adherence of the system properties, goals, functionalities to those described in the required documentation, and verifying the compliance of the documentation itself with the scheme. Duration and costs of a security evaluation are straight connected to the certification level and to the complexity of the system/product. Considering for example the ISO/IEC 15408 standard (aka Common Criteria [8]) and three of the 6 evaluation assurance levels foreseen by the scheme⁴, the relating effort in terms of cost and time is depicted in Fig. 1.1.

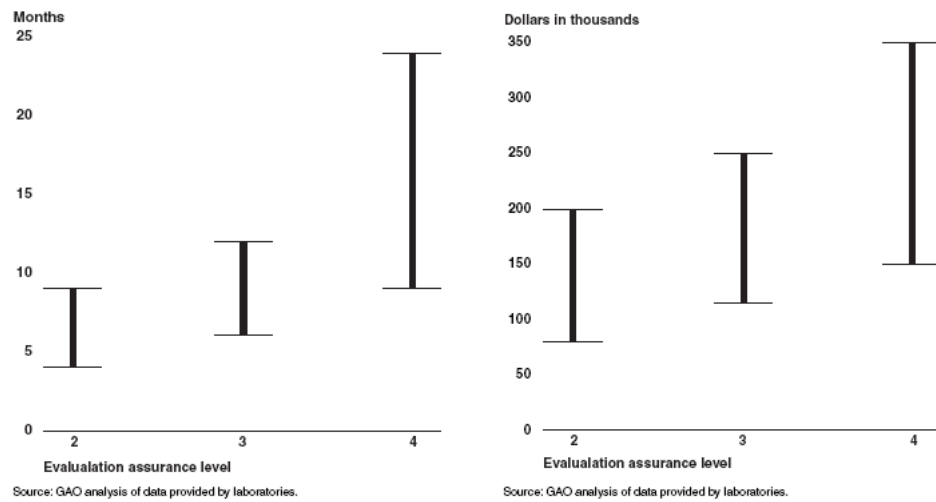


FIGURE 1.1: Range of costs for Common Criteria evaluation at EAL2, EAL3, and EAL4 [1]

A security certification is often a mandatory requirement for many National Programs, normally belonging to military and government sectors, but it is also becoming a valued requirement in the civilian world. For this reason it is important to consider this topic as integral part of the multilevel problem.

1.4 Covert Channel Problem

One of the goals a multilevel security solution should meet is the capability to be certified at very high levels according to security evaluation scheme like, for example, the ISO/IEC 15408 one.

Since an MLS system must perform a Mandatory Access Control on classified data and enforce the information flow control policy, it should also provides evidences that it is able to prevent illicit data flows that could subvert such policy. An information control flow policy is focused on data flows that move between different levels but also among

⁴Common Criteria foresee 6 Evaluation Assurance Levels. Higher levels require more accurate verification on a higher number of security functionalities.

objects that can be located along the communications path.

According to the ISO/IEC 15408 standard a covert channel is an enforced, illicit signaling channel that allows a user to surreptitiously contravene the multilevel separation policy and unobservability requirements of a high-assurance system. To prevent such risk the certification scheme requires a covert channel analysis and capacity estimation to be performed in order to mitigate such vulnerabilities. A covert channel could for example arise from a shared resource or from resource contention issues present in traditional computer architectures. According to us, covert channels represent another topic that a security engineer/architect should face when dealing with MLS systems.

1.5 Drawbacks of Current Solutions

Although an MLS system is a requirement for many high-assurance security contexts, actually the goal has been achieved only partially, because the mechanisms used to enforce the CIA⁵ properties present several aspects of weakness that will be discussed in this thesis. The issues mainly concern the design approach that led to complex systems, the difficulty to demonstrate the effectiveness and the assurance of the system security mechanisms, and the impossibility to evaluate the proposed solutions at the high levels required for multilevel systems.

A new approach has been proposed aiming at going beyond the limits of the classic MLS approach: the Multiple Independent Levels of Security/Safety. The new paradigm, together with the development of formal languages, new architectural/platform modeling frameworks, and the recent advances in commercial microprocessors made MILS a practical and feasible solution for real applications.

1.6 Outline of the Thesis

The remainder of this thesis is structured as follows: the next chapter ([Chapter 2](#)) will present the classic multilevel security approach, the state of the art of the MLS solutions and the drawbacks that restricted them for narrow military applications. The limited success of MLS started a research path that resulted in a new paradigm, called Multiple Independent Levels of Security/Safety (MILS) that currently represents the effort shared by Industry and Academia. The main part of the thesis is focused on a novel MILS architecture that was designed during the doctorate studies and that led to the development of a specific Cross-Domain component ([Chapter 3](#)). In [Chapter 4](#) the covert channel problem is presented together with a taxonomy of such vulnerability class.

⁵Confidentiality, Integrity and Availability.

Because the design of a high-assurance solution cannot disregard the covert channel problem, a novel detection algorithm has been specifically designed for a class of covert channels called Covert Timing Channels (CTC). Additional details as the code of the developed CTC are given in [Appendix A](#), [Appendix B](#) and [Appendix C](#).

From Multilevel Security to MILS. This chapter depicts in detail the Multilevel Security approach, its peculiarities, innovations, drawbacks. After introducing the basic definitions and concepts, the enforced security models and the drawbacks of the classical multilevel approach, the new MILS paradigm is introduced.

Novel MILS Architectures. In this chapter a distributed novel MILS architecture designed to fulfill the high assurance objectives required for multilevel security solutions is presented. The design approach and a specific architectural component implementing a Cross-Domain Solution are also described.

Covert Channel Detection. In this chapter the covert channel problem is presented and a detailed taxonomy of such vulnerabilities is introduced. The state of the art of the detection algorithms is also presented in order to give the reader some basic concepts used in the proposed one. A novel detection algorithm is then presented for the Covert Timing Channels class. To test its effectiveness, different CTCs have been implemented, whose code is detailed in [Appendix A](#).

Conclusions. Here the main contribution of the thesis is reviewed, learned lessons are pinpointed, and some ideas for future research are presented.

Chapter 2

FROM MULTILEVEL SECURITY TO MILS

THE TERM MULTILEVEL SECURITY arises from the security standard known as Orange Book [9], that defines a Multilevel Secure System as “a class of systems containing information with different sensitivities that simultaneously permits access by users with different security clearances and needs-to-know, but prevents users from obtaining access to information for which they lack authorization”. The idea behind MLS is that a system must be able to enforce the mandatory security policies over classified data, accessed simultaneously by users with different clearances and need-to-know, and to preserve the fulfillment of the applied security models.

An MLS system is usually based on a hierarchical access control model that is founded on a regular tiling called “lattice”¹. In this model the system is required to assign security labels to objects and processes, aiming at enforcing the policies established by the Bell-La Padula model. The mechanism is known as Mandatory Access Control (MAC): the user can not disable or bypass it. On the opposite there is the Discretionary Access Control (DAC) that commits to file owners the responsibility to enforce the access control policies. In the last case the owner of the file can accidentally or intentionally violate the policies by simply modifying the access permissions. By default Bell-La Padula protects information from its leakage toward lower sensitivity levels and maintains the confidentiality of the objects that cannot be accessed by lower clearance subjects. It ensures data confidentiality with the MAC policy “no read-up” and “no write-down”. Its dual model, called Biba, enforces data integrity according to the MAC policy “no write-up” and “no read-down”. When these two models coexist in one single environment, it means that both the policies must be enforced simultaneously and no flows are

¹An information flow model consists of objects, state transitions, and lattice (flow policy) states.

allowed downwards and upwards. This is the case when communications among different sensitivity levels is not allowed at all and each element of the MLS architecture becomes a *Single-Level Secure* (SLS) component.

Enclaves can also use *Multiple Single-Levels* (MSL) components. In this case dedicated machines are used for each specific level and data can be exchanged in compliance with the Bell-La Padula model from a level to a higher one using portable storage media.

Usually, when a system must enforce a mandatory access control, it is referred according to the security mode of operation it uses. Each security mode of operations is determined by the users' clearance, their need-to-know, the levels of classification of data and a formal access approval to access data. There are different modes of operations:

- *Dedicated Security Mode*
- *System-High Security Mode*
- *Compartmented Security Mode*
- *Multilevel Security Mode*

In the *dedicated* security mode of operations all users have a security clearance or authorization and the necessary need-to-know for all information. In a domain operating at *system-high* security mode all users have a security clearance or authorization but not the necessary need-to-know for the processed information. In *compartmented* mode users have the appropriate clearance but not the formal approval and the need-to-know to access all the processed information. Finally, the *multilevel* mode of operations further restricts the users access on processed data, because they do not have a valid clearance, authorization and need-to-know for all data treated in the security domain.

Often military enclaves use the system-high security mode of operations to treat information that otherwise would be characterized by different classification levels. All non-MLS modes let information blow up quickly in the sensitivity hierarchy and requires a swift and robust declassification procedure to enable the flow in the opposite direction (from up to low).

The MLS problem is not limited to the sectors introduced in the previous chapter in fact, more generically, ICT systems need to exchange data and their software needs to be patched/updated. Hence, they need a connection to the Internet. Attacks can be launched by insiders or by remote attackers that use this link to reach their targets all over the world. Through a compromised host, attackers can steal and exfiltrate information or can launch further combined attacks. As underlined by an important security professional [10], if two domains need to be interconnected, even if each of them works at system-high security mode of operations, they need a multilevel device in their connection path.

Designing and developing an MLS solution is anything but a trivial task, because it requires the analysis of many aspects that belong, but are not limited, to the technological, regulatory, and mutual international certification agreement sphere. Such aspects are reflected in many requirements to be fulfilled, and in the intrinsic bigger complexity of multilevel security systems.

The combination of system-high security mode of operations, Single-Level Secure and Multiple Single-Levels components is quite far from an effective MLS system that must store and forward simultaneously information at different classification levels and enforce the MAC policies in combination with the Bell-La Padula security model. In the light of this, the MLS problem has been faced for long by Academia and Industry.

2.1 Security Models

As already stated, Multilevel Security represents the capability of an ICT system to store and process concurrently information with different characteristics of sensitivity on which users operate with different clearances, permissions, and roles. An MLS system must enforce the established security policies and specific security models, that is, Bell-La Padula and Biba.

A security model describes a security policy in terms of rules that must be enforced to achieve the desired security objectives. It is usually represented through mathematical notation so that it results unambiguous or cannot lead to misinterpretations. Bell-La Padula, Biba, Clark-Wilson represent formal security models and they are designed to provide high assurance, unlike informal model that are not mathematically validated (e.g. the Chinese Wall model).

If a system is modeled by a finite-state machine with a set of operations that modify the system's state, then a security rule defines the transitions that drive the system from one authorized state to another authorized state. When the transitions lead always the system to authorized states, then the system is considered secure.

2.1.1 Bell-La Padula

Bell-La Padula focuses on data confidentiality and access to classified information, unlike the Biba model that concentrates on data integrity. A system is in a secure state if all ways used by subjects to access objects are compliant to the established security policy. The access scheme is expressed as a lattice. An operation is valid and allowed if and only if the resulting state of the system is secure, that is, all the model properties are satisfied. This model defines two mandatory access control rules and a discretionary access control one, with three security properties:

1. **simple security property**, stating that a subject can access to an object only if her clearance dominates or is equal to the object classification level (no read-up),
2. **star (*) security property**, stating that a subject can access to an object for append-operations only if her clearance is dominated by the object classification level, s/he² can access it for write-operations only if her clearance and the object classification level are equal, and s/he can access to an object for read-operations only if her clearance dominates the object classification level (no write-down),
3. **discretionary security policy**, that uses an access matrix to specify a discretionary access control. It states that a subject can employ only accesses on which s/he has the required authorization.

The transfer of data from a higher level to a lower one can be performed only through trusted subjects that are not limited by the previous properties.

2.1.2 Biba

Biba model is focused on the definition of a rule set for subjects operating on objects that are characterized by different levels of integrity. In general preserving the integrity of an object means:

- to avoid data modifications by non-authorized subjects,
- to avoid non-authorized data modifications by authorized subjects,
- to maintain the internal and external consistency of data.

Biba model implements this kind of protection through the definition of an ordered serie of integrity levels for subjects and objects in respect of the following properties:

1. **simple integrity property**, stating that a subject is allowed to write to an object only if her clearance dominates or is equal to the object classification level,
2. **star (*) property**, stating that a subject can access to an object for read-operations only if her clearance is dominated or equal to the object classification level.

It is not simple to make Bell-La Padula and Biba model coexist on the same machine. As already underlined, when they are applied “as is” in a context, there can be no

²The term s/he could be read as “she or he” throughout the Thesis.

information flow between different classification levels and the system reduces itself to a SLS domain. To use both models in an MLS domain it is necessary that the policies applicable to confidentiality and integrity have independent classifications.

2.1.3 RBAC

The Role-Based Access Control is an approach based on the concept of role, used in restricted access systems for authorized users. Specific roles are assigned to members that gain in this way permission to perform specific functions. Since rights are not directly assigned but only acquired through the role (or roles) assigned to users, the management of user individual rights becomes a simple assignment of appropriate user roles. Three basic rules are defined for RBAC model:

1. **Role assignment**, a subject can execute a transaction only if s/he has been assigned to a role,
2. **Authorization roles**, an active role for a subject must have been authorized,
3. **Transaction authorization**, a subject can execute a transaction only if the transaction is authorized for the active role of the subject.

Additional constraints can be foreseen, and also roles can be combined in a hierarchy where higher levels are made by the permissions of the lower level roles.

2.1.4 Clark-Wilson

Clark-Wilson model was developed in 1987 with the Biba's goal of protecting the integrity of data, in particular against frauds and errors in the commercial sector. Such context, unlike the defense one, is interested more in the integrity of processed data (particularly in commercial transactions) than confidentiality and in the consistency of the system state. The model identifies four main concerns related to the commercial integrity goal:

- **Authentication**, each subject must be authenticated,
- **Audit**, any modification must be logged,
- **Well-formed transactions**, user can manipulate data only in constraint ways,
- **Separation of duty**, a critical task cannot be carried out by one entity and so must be divided into different parts on which different subjects operate.

In contrast to Biba that protects information from being modified by unauthorized users, the Clark-Wilson model prevents also unauthorized modifications by authorized users and maintains the consistency of data.

2.1.5 Chinese Wall

Another security model focused on data integrity for the commercial world is the Chinese Wall one developed by Brewer and Nash. The example in [11] explains in a simple way the idea behind this model, defining the code of practice of a market analyst working for a financial institution providing corporate business services: *“such an analyst must uphold the confidentiality of information provided to him by his firm’s clients; this means he cannot advise corporations where he has insider knowledge of the plans, status or standing of a competitor. However, the analyst is free to advise corporations which are not in competition with each other, and also to draw on general market information”*.

The main goal of this model is to deny data access to users when a conflict of interest can arise from this access. Unlike Bell-La Padula and Biba, where data access is constrained by attributes of data, in the Chinese Wall model the access is conditional to the information already accessible by the subject.

2.2 Related Work

Since years the Multilevel Security is an important requirement for military and intelligence systems, essential to meet their automation needs in the perspective of the information dominance. The origin of MLS dates back to the '60s, when the defense community identified the requirements for multilevel security [12, 13]. Many products were deployed to solve the problem. Multics, an operating system derived from the MIT’s Compatible Time Sharing System (CTSS) [14], was a mainframe operating system used from 1970 until 2000. Started as a research project, it provided multiple users with secure time-sharing computing resources. It was designed from the scratch in order to meet security, despite of other OSes that added security as an afterthought. [15] describes the design principles underlying Multics security, and the weaknesses of the protection mechanisms.

The Secure Communications Processor (SCOMP) [16] was a processor derived from Multics with formally verified software and hardware components [17]. It was based on a kernel with 4 different rings of protection. Two rings were considered trusted and comprised the SCOMP Trusted Operating System, whereas the other two, called untrusted, provided an environment for the Kernel Interface Packages. SCOMP was applied to control data flows between areas characterized by different classification levels and was

also used as a military mail guard. An important contribution of such system was its usage as a model for the development of the Orange Book [9]. Its descendants, the XTS-200 and XTS-300 systems, were later used to move military plans from Command & Control Systems towards troops lower classification levels.

Later on, many other MLS solutions appeared on the market, described for example in [18, 19, 20, 21]. In 1993, Kand and Moskowitz developed the U.S. Naval Research Laboratory (NRL) Network Pump to enforce data flow control from a low security level network to a higher one [22].

A general but detailed overview of the development of MLS systems can be found in [23].

2.3 State of the Art of Classic MLS

In the '70s, the U.S. Air Force commissioned a study pointed to analyze and develop strategies for building and verifying the assurance and the effectiveness of MLS systems. The results of this study was summarized in the Anderson Panel Report [24]. In order to make a design verification feasible, this report suggested that an MLS system should enforce the established security policies through a reference validation mechanism, also called Reference Monitor (RM). An RM is a software element that enforces the MAC through mediation of the access control decisions. It should be small in terms of lines of code in order to make verifiable the completeness and the effectiveness of the operations it claims to perform. Modern Operating Systems (OS) include a portion of the software architecture that incorporates an RM and has unrestricted access to the computer resources. It represents the Security Kernel. On the top of it some programs and processes, that support users, run with privileged access to system resources without bypassing the Security Kernel. The set of computer hardware, Security Kernel, and its privileged components constitutes the Trusted Computing Base (TCB), that is the system component responsible for enforcing the MLS security policies. The assurance and verification that the TCB performs the operations properly ensures that the MLS restrictions are effectively enforced.

2.3.1 Operating Systems

By the early '70s, it was already clear that the commercial OSes did not offer enough data protection in terms of confidentiality, integrity and availability. Starting from this awareness, designers developed systems like Multics and SCOMP. Later on, many OSes were designed in order to enforce the MAC security policy and limit the effects of a security breach. Such OSes are commonly referred as Trusted Operating Systems.

Often software houses maintained a separate developing line for such OSes in order to offer commercial versions without MAC enforcement options, but this trend was later abandoned in the perspective of security certifications required by customers. The current marketplace offers many Common Criteria certified OSes.

Oracle Solaris 11, for example, is EAL4+³ certified and has some features that extend its security capabilities by enforcing a label-based MAC policy.

The TrustedBSD project was originally targeted to meet the security functionalities required by the Common Criteria. Nowadays the TrustedBSD effort is focused on the research and the development of security extensions to meet specific security goals. The TrustedBSD MAC Framework augments the system security mechanisms in order to implement the mandatory access control. Its features are now integrated in the FreeBSD operating system [25].

Solaris and FreeBSD use a strong mandatory access control architecture called FLux Advanced Security Kernel (FLASK) [26]. FLASK provides a flexible support for security policies that was initially integrated into the NSA's Security Enhanced Linux (SE Linux). It progressively became the core framework of security-focused operating systems like, for example, Solaris, FreeBSD, Red Hat Enterprise Linux⁴, CentOS, Fedora, Darwin Kernel etc.

Other OSes like Suse Enterprise Linux, OpenSUSE, Debian, Gentoo etc. use a different kernel security model called AppArmor [27] that supplements the classical DAC model with the mandatory access control one. In contrast to SE Linux that utilizes labels for files, it works with file paths and uses a combination of static-analysis and learning-based tools to proactively protect the OS and the applications.

IBM z/OS V2R1 and BAE Systems STOP OS v.7⁵ are other security-oriented operating systems evaluated at EAL4+ according to the Common Criteria scheme.

Microsoft introduced in Windows Vista a security feature called Mandatory Integrity Control (MIC) for the enforcement of a security model that resembles the Biba one. It restricts the access permissions of applications in a way that a lower-level subject cannot modify a higher-level object. Unlike the strict integrity policy of Biba, Windows does not inhibit higher-integrity subjects from reading or executing lower-integrity objects [28].

³The concept of Evaluation Assurance Levels (EAL) in the Common Criteria scheme is described later in the Thesis.

⁴Currently the version 7 is under certification at EAL4+ according to the Common Criteria scheme.

⁵STOP OS has a strong history of evaluated products, including SCOMP and the XTS systems.

2.3.2 Databases

Together with the implementation of OSeS, the Database Management System (DBMS) technology has been provided with multilevel security features. As well as SLS components, the maintenance of separate databases for different levels of classified data is costly in terms of money and administrative effort. Oracle, for example, introduced a “label security” tool to access hierarchical data by comparing their labels with the user’s clearance.

There are many Commercial Off-The-Shelf (COTS) certified products for database: Oracle Database 11g Release 2 Enterprise Edition (EAL4+), Microsoft SQL Server 2012 Database Engine Enterprise Edition x64 (EAL4+), IBM DB2 Version 9.1 for z/OS Version 1 Release 10 (EAL4+) to name but a few.

2.3.3 Virtualization

In the panorama of MLS products, virtualization represents an interesting solution particularly in terms of Size, Weight and Power (SWaP) savings. Virtualization introduces an abstraction class that decouples adjacent layers to deliver flexibility and better resource utilization. It is a mature technology whose effectiveness is increased over the years thanks to the improvements of commercial microprocessors. Virtualization allows different operating systems and standard applications from multiple security domains to run on a single machine. Xen Security Modules (XSM) and Xen Flask Security Module are for example software elements that allow the implementation of a MAC through the FLASK architecture introduced before. XSM enables an administrator or developer to manage fine-grained control over a Xen domain.

Also VMware released different products to encapsulate OSeS, applications and data into isolated layers. VMware View is a component built on the EAL4 certified VMware vSphere for Desktop platform that allows an organization to manage different levels from a single administrative console. Different instances of OSeS reside in the cloud and authorized users can access their sessions from different networks.

Many other solutions are actually provided by IBM, Citrix, Microsoft, Nimboxx etc., highlighting the successful adoption of such technology in high-critical security environments.

2.4 Cross-Domain Solutions

The need to rapidly collect, elaborate, and share information is the paramount requirement for the armed forces. Often data are moved among domains that are not under the

same security policies, for example in federated or coalition networks. Through the integration of different domains it is possible to increase the overall capability of the joint system. This is called *Cross-Domain Synergy*. The need to share information across different operational and coalition boundaries is not an exclusive military prerogative, in fact all the sectors introduced in paragraph 1.1 require this kind of solution.

A cross-domain system is “a form of controlled interface that provides the ability to manually and/or automatically access and/or transfer information between different security domains” [5]. The U.S. Unified Cross-Domain Management Office (UCDMO) indexed the cross-domain mechanisms as transfer, access, and multilevel. “A transfer device permits the movement of data from one domain to another. An access device allows a user to sit on one workstation and access multiple domains but not move data between them. A multilevel device stores and processes information of different security levels in a common repository, but only allows a user to view appropriate information based on his/her credentials” [29].

A dual-use system requires a cross-domain solution. The same solution is necessary to connect two or more system-high military networks. Extending the concept, all the systems belonging to the aforementioned contexts need to be regularly updated and patched, in other words need a connection to the Internet (at least by a cross-domain type-transfer device). In multilevel systems it is important to control the information flows and prevent those that could illicitly disclose data as in the presence of covert channels [30].

In particular there are applications that require unidirectional data flows, for example when a classified domain needs a connection to an unclassified one or to the Internet. Actually, different cross-domain solutions are used in order to enforce the Bell-La Padula model and the unidirectional data flow control, as introduced in the next paragraph.

2.4.0.1 MLS Guards

A Cross-Domain Solution (CDS) is often referred as “guard” or “secure gateway”, because it positions itself as the unique interface among different security classification level domains.

A guard is an important requirement for NATO in the concept of Information Exchange Gateway (IEG): a security architecture with standardized interfaces conceived to enable secure communications among NATO and non-NATO nations, organizations and agencies.

Currently many companies offer their products based on the multilevel technologies mentioned in the previous paragraphs, enriched with encryption mechanisms and supporting different security protocols. BAE Systems, for example, proposes its XTS Guard 5 to

enable secure sharing across different classification, operational, and coalition boundaries. It allows users to share data like chat, XML, imagery from a single computer system in conjunction with the secure STOP operating system.

General Dynamics [31], Forcepoint [32], Lockheed Martin [33], Thales [34], and Advatech Pacific [35] are other examples of companies that offer a plethora of MLS solutions based on COTS processors, SE Linux, AppArmor, Virtualization platforms and so on. The need of secure guards to connect mobile devices such as laptops and Personal Digital Assistants (PDA) in tactical environments, drove Owl Computing Technologies to design a Miniaturized Cross Domain Solution (MCDS) suitable for Android GOTS platforms [36].

2.4.0.2 Air-Gap

The Network Working Group defines the Air-Gap as “*an interface between two systems at which (a) they are not connected physically and (b) any logical connection is not automated*” [37]. Air-Gap represents a security system that physically detaches a protected network from an unreliable one and it is often used in highly secure environments to protect data from illicit disclosure. It is sometimes referred as “manual” guard.

The air-gap is composed by two elements, the first one (called also Bastion) is connected to a network on which specific security policies are enforced, e.g. a Local Area Network (LAN). The second one is physically isolated from the protected LAN and is connected to a network that is not subordinated to the domain security policies (thus, non-reliable from the LAN perspective). An authorized user can make use of the Bastion, for example, to save data on a portable storage media and export them to the unreliable network (after proper declassification) connecting the same storage device to the second host. On the opposite, the second host could be used to download data from the Internet and import them to the internal LAN through the Bastion.

2.4.0.3 Data-Diode

The Data-Diode is used as well as the Air-Gap to enforce the Bell-La Padula security model, establishing one-way communications when data are required to flow from a network with specific classification level to a higher classification level one. It usually consists of two operative interfaces. At the low security level host, the electrical signal is converted into light and sent to the high security level host via an optical fiber. The high security level host does not have input connections and also lacks the ability of converting electrical signal into light. Since the unidirectional flow is realized in hardware and supported by a light firmware, these solutions can get very high levels of assurance

in security certifications like the ISO/IEC 15408.

The marketplace provides a wide range of high-assurance data-diode solutions for civilian and military applications.

Table 2.1 summarizes the state of the art of classic MLS solutions.

TABLE 2.1: Classic MLS technology overview

Category	Name	Producer
Operating System	Solaris 11	Oracle
	FreeBSD	The TrustedBSD Project
	Suse Enterprise Linux	Novell
	OpenSUSE	The openSUSE Project
	Debian	The Debian Project
	Gentoo	Gentoo Foundation
	z/OS	IBM
	STOP v.7	BAE System
	XTS-400	BAE System
	HP-UX	HP
	RHEL 7	Red Hat
DBMS	Oracle DB 11g EE	Oracle
	SQL Server	Microsoft
	DB2 v.9.1 for z/OS rel.10	IBM
Virtualization	Xen	The Xen Project
	VMware vSphere	VMware
	KVM	Open Virtualization Alliance
	<i>others</i>	IBM, Citrix, Microsoft, Nimboxx, etc.
Cross-Domain Solution	Air-Gap	<i>general purpose computers</i>
	Data-Diode	Owl computing Technologies, BAE Systems, Advenica, Nexor, Fox-IT, etc.

2.5 Drawbacks of the classical approach

Although a MLS system is a requirement for many high-security systems, actually the goal has been achieved only partially, because the involved mechanisms present several aspects of weakness. Table 2.2 summarizes the shortcomings of the classic approach described in this paragraph.

In the past, secure systems have been designed with the concept of Security Kernel and TCB. It means that the security decisions and the security mechanisms were integral part of the TCB. Following this approach, designers and developers started adding more and more of their system functionality into the TCB. This approach led to the following drawbacks:

- the security functions are difficult to separate from other system features (monolithic applications),
- the size of the system is inclined to be very large,
- it becomes impossible to formally verify the correctness of a system with thousands of lines of code.

TABLE 2.2: Shortcomings of classic MLS approach

ID	PITFALL
MONO	monolithic applications
TCB	complexity of TCB
SIZE	size of TCB
EVA1	unfeasible formal/semi-formal evaluation
EVA2	difficult and costly high-level evaluation
CC	difficult covert channel analysis
AG1	non automatic data transfer (Air-Gap)
AG2	management of additional sw/hw (Air- Gap)
DD1	high costs (Data-Diode)
DD2	non suitable for C-O protocols (Data-Diode)
GOV1	weak government policies for MLS
GOV2	government advocacy for weak MLS solutions

MLS systems are usually associated to multiple levels of security with Mandatory Access Control enforcing the Bell-La Padula rules. But many multilevel systems that are currently used in the defense environment implement simply the system-high security mode of operations with some form of communications between them, through for example Air-Gaps and Data-Diodes.

Another problem is represented by covert channels, a class of vulnerability that can lead to disclosure of sensitive information and that benefits from the speed improvements of the current communications networks [38]. Covert channel analysis becomes very difficult with large and monolithic TCB.

Air-Gap effectively ensures total isolation and separation of networks and is not very

expensive compared to other classic multilevel solutions, but it presents some disadvantages. For example, it requires manual intervention of an operator to import (low to high) or downgrade-and-export (high to low) data. It also requires the management of the Air-Gap hosts (software and hardware) and the CD/DVD/usb-sticks used to move data, in all their life-cycle.

The advantage of Data-Diodes is the high level of assurance got in security certification schemes like Common Criteria. The drawbacks are different, for example the high costs and the capability to allow only connectionless protocol communications. Modern Data-Diodes support also connection-oriented (C-O) protocols like TCP by adding software proxies that automatically simulate the connection establishment and termination handshakes but with no possibility of packet re-transmission if some errors occur [39]. Bell underlined in his paper [23] the fact that the marketplace has never produced high security products spontaneously and government versions of MLS products have never been viable over the long term. Because of the shortage of MLS products, most organizations chose to deploy multiple computer networks, each dedicated to the specific security level they needed. When specific critical applications required an MLS system, often such requirement was fulfilled by low-assurance commodity technology.

According to Bell, even if the U.S.A. context defined a road-map to address the high-assurance security needs of the cyber-era, it enforced weakly the policy to use validated products. Also, when the use of certified products has been fostered in the last years, the National Security Agency (NSA) advocated weak⁶ NSA products, often preferring them to stronger commercial ones.

Furthermore, the high costs of an MLS solution, owing to the certification process and combined with the limited number of customers, made the success of classic MLS only partially feasible.

2.6 Multiple Independent Levels of Security/Safety

In the last years a new multilevel security paradigm has been pursued by security engineers: the Multiple Independent Levels of Security/Safety (MILS).

The MILS paradigm has been developed to solve the difficulty of certification through a “*divide et impera*”⁷ approach, that is the creation of small high-secure and high-reliable software elements. The separation of security mechanisms and issues into independent and manageable components simplifies the process of the specification, design and analysis of high-assurance computer systems. Together with the concept of “*compositional*

⁶According to Bell, here the term “weak” stands for a certification level less or equal to EAL4.

⁷Latin expression from which the english “divide and conquer” comes from.

approach” of Common Criteria certification, MILS can enable a real, fast, and cost-effective approach to security certifications required by the existent market [40].

The concept of MILS arose from the need of *“robust partitioning”* required for Integrated Modular Avionics (IMA) by standard like ARINC 653 [41] and DO-178B [42]. Such partitioning is required to avoid that a fault within a computer system could propagate and affect the operations of other safety critical functions in avionic architectures. The origin from the airborne systems is clearly quoted in the word *Safety* in the MILS acronym. Traditionally the aircraft control functions were implemented separately on different fault-tolerant machines that needed to exchange simple sensor and control data. Later the avionic architecture started collecting these functions on a single fault-tolerant computer where fault containment was more difficult to achieve. Robust partitioning comprises the protection of each partition’s memory addressing space, the restriction of the processing time assigned to each partition and the restriction to the execution of privileged instructions.

Rushby examines the principles and the mechanisms to provide assured partitioning in [43]. In particular he states: *“because partitioning shares some concerns with computer security, security models are reviewed and compared with the concerns of partitioning”*. The concept of robust partitioning is very close to the Secure Isolation Formal Model, that later was used into a security verification technique called *“Proof of Separability”*, presented in [44]. In his approach Rushby proposed to decouple the verification of components implementing trusted functions from the verification of the Security Kernel. This result can be achieved by dividing memory into partitions and allowing only strictly controlled communications across them. In this way a single partition is required to offer services to another one with minimal action from the Security Kernel.

These concepts led to the introduction of the Separation Kernel (SK) whose features were collected in the Separation Kernel Protection Profile (SKPP) [45], an effort by the U.S. Government that was later abandoned in 2011 [46]. In spite of the sunset of the SKPP, the Separation Kernel approach represents the cornerstone of current MILS solutions. The Separation Kernel and its network-homologous, the Partitioning Communication System (PCS), constitute the core of MILS architectures. They implement and enforce the separation and isolation policies on a single machine and over an entire computer network, respectively.

The following table presents the subset of MILS features that overcome the drawbacks of the classic MLS.

TABLE 2.3: MILS features and solutions overcoming the classic MLS drawbacks

MILS feature	MILS drawback	MILS solution
Robust Partitioning	MONO, EVA1, CC	SK Hypervisor, SK OS
Divide & Conquer Approach	TCB,SIZE,EVA1,CC,EVA2	SK Hypervisor and SK OS
Compositional Approach	AG1, EVA1	SoftGap, MILS Guards
Virtualization	AG2, DD1, DD2, EVA2	SK Hypervisor
CCRA	GOV1, GOV2, EVA2	Common Criteria standard, certified cryptographic and hashing algorithms

The specific MILS solutions are detailed in the next paragraphs.

2.6.1 NEAT Paradigm

A MILS system must be designed according to essential requirements that were specified initially for the Reference Validation System in the Anderson Panel Report [24]. These requirements are today referred as NEAT properties, where the acronym stands for:

- **Not-Bypassable**, the safety functions must not be subverted;
- **Evaluatable**, the safety functions must be small enough to be mathematically verified and evaluated;
- **Always-Invoked**;
- **Tamperproof**, the security functions must not be altered by malicious code.

The enforcing of formally-demonstrated security policies guarantees information flow control, data isolation, predictive control over their processing operations, limitation of damages in case of a security breach, availability of resources and the possibility of certification at high levels of assurance.

2.7 State of the Art of MILS Solutions

In the last few years the need of multilevel systems that went beyond the limits of the classic approach encouraged the appearance of new MILS solutions. In the next paragraphs the Separation Kernel Hypervisors and MILS Operating Systems are presented. The MILS Guards are instead detailed in the next section.

2.7.0.1 Separation Kernel Hypervisors

A Separation Kernel Hypervisor is a virtualization platform that allows to simultaneously run multiple OSes on a single hardware platform, guaranteeing that the OS guests are separated in a secure way and cannot affect each others' functionalities. The physical device access and the information flows among the Virtual Machines (VM) are strictly controlled. The goal of an SKH is to host high-assurance systems on general purpose devices in order to support affordable security evaluations. A Separation Kernel Hypervisor is usually very small in terms of lines of code because it is stripped of functions on which it could not provide adequate protection, whereas it concerns only about those that enforce the MILS policy. Such features make an SKH suitable for high performance/real time applications and make practical a high assurance security evaluation that requires semi-formal and formal analysis of the claimed security mechanisms.

Lynx Software Technology offers its LynxSecure Separation Kernel Hypervisor [47], a type-0 hypervisor designed in the principle of least privilege. It means that an entity is not granted more privileges than necessary for it to perform its functions, limiting the damages that the entity could do in case of failure [48].

Wind River provides a type-1 hypervisor called VxWorks MILS [49] that, as well as LynxSecure, meets the requirements in the "U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness", version 1.03.

Other companies offer their own SKHs like Green Hills Software with its Integrity Multivisor or Sysgo with PikeOS. The former is a real-time platform for safely and securely combining OSes such as Linux, QNX, Android or Windows with safety and security-critical software on 64-bit multicore processors. It is based on Green Hills Software old Integrity 178-B OS. PikeOS was chosen in the EURO-MILS project [50] in the effort of establishing it as the European security platform for embedded systems certified up to level EAL5+.

An open source solution specifically developed for embedded devices by General Dynamics Mission Systems is the OKL4 Microvisor. It can be used as a hypervisor as well as a simple real-time operating system with memory protection.

SeL4 Microkernel is an open-source variant of OKL4 owned by General Dynamics Mission Systems and it has been formally verified by the Australian Information and Communications Technology Research Center (NICTA).

All the aforementioned solutions support a wide variety of specifications and standards and have been certified according to specific schemes, but none of them has been yet certified according to the Common Criteria one. This represents the major problem of their usage in European projects that require ISO/IEC 15408 certified GOTS/COTS.

2.7.0.2 Separation Kernel Operating Systems

In the field of Operating Systems, Green Hills Software Integrity 178-B RTOS is the only one that has been certified at a very high assurance level. It gained the Common Criteria EAL 6+ certification by the National Information Assurance Partnership (NIAP) in 2008 [51] on a Power PC platform. The relatively simple hardware of the platform does not meet the features of the current workstations [52].

Wind River VxWorks 365 is an ARINC 653-compliant operating system that uses multicore separation and partitioning to allow the reuse of legacy software. It is used in many different aerospace and medical programs, for example Selex ES company chose it for its MILS gateway solution [53].

The Micrium C/TimeSpaceOS is a real-time kernel that allows several independent applications to be executed simultaneously on the same hardware platform, guaranteeing that the applications do not interfere with each other. Each application is independent from other partitions.

Quest is an operating system designed to be dependable and predictable. It is an open source software available under the terms of the GNU General Public License v3. It can be used as guest OS on the Quest-V separation kernel that isolates guest domains in separate sandboxes. Quest-V supports also OSES such as Linux.

The Muen⁸ Separation Kernel is an open source microkernel developed by the Institute for Internet Technologies and Applications (ITA) at the University of Applied Sciences Rapperswil (HSR). It is formally proven to contain no runtime errors at the source code level [54].

Qubes OS is another operating system that utilizes virtualization technology to isolate applications from each other and also to sandbox components like networking and storage subsystems. It is based on Xen hypervisor. The OS designers decided to choose this solution because Xen isolation is based on virtualization enforced by a thin hypervisor. On the contrary, solutions like KVM rely on the Linux kernel to provide isolation that, according to the designers, is not as secure as Xen. Qubes OS provides integration of the applications hosted in different VMs through a common user desktop, whose GUI is hosted in an administrative VM, called *Dom0*, that is the only one that has direct access to graphics and input devices.

PolyXene is a MILS solution that ensures the secure cohabitation of data and applications of different classification levels on a single workstation [55]. It is the result of the collaboration between Bertin Technologies and the French Department of Defense, certified in 2009 according to the Common Criteria standard at EAL5 assurance level. PolyXene version 2 is currently being assessed for Common Criteria EAL5+ certification

⁸Muen comes from Japanese and can be translated into “without relation” and refers to the components that are isolated by the separation kernel.

[56]. The solution foresees a trusted zone and one or more standard zones, that can be considered just like separation kernel partitions. Like Qubes OS, it provides virtualization through an external component, here the VirtualLogix VLX virtualization software, owned today by Harman International Industries and used specifically for mobile and automotive solutions [57].

TABLE 2.4: MILS technology overview

Category	Name	Producer
Separation Kernel Hypervisor	LynxSecure	Lynx Technologies
	VxWorks MILS	Wind River
	Integrity Multivisor	Green Hills Software
	PikeOS	Sysgo
	OKL4 Microvisor	General Dynamics Mission Systems
	SeL4 Microvisor	General Dynamics Mission Systems
SK Op- erating System	Integrity 178-B RTOS	Green Hills Software
	VxWorks 356	Wind River
	μ C/OS	Micrium
	Quest	Boston University
	Muen	Rapperswil University
	Qubes	The Qubes Project
Cross-Domain Solution	Polyxene	Bertin Technologies
	SoftGap	Sira
	SecureOne KOV-7	Rockwell Collins
	MicroTurnstile	Rockwell Collins
	Themis	Thales

Table 2.4 summarizes the state of the art of MILS solutions.

2.8 Security Evaluation Criteria

One of the goal a multilevel security solution should meet is the capability to be certified at very high levels according to security evaluation standard like the ISO/IEC 15408. Though a security evaluation it can provide certified evidences that it addresses appropriately critical properties like security and safety.

The avionic sector first developed standards and guidelines to design, analyze and evaluate safety systems, nevertheless such effort was not able to meet the security issues of high assurance systems.

In the '70s, under the auspices of the Department of Defense (DoD), different works

were published to address computer security problems and to establish uniform DoD policy, requirements, administrative and technical countermeasures [9].

Concurrently with the DoD's efforts, the National Bureau of Standards (NBS) focused its attention on the definition of problems and solutions for building, evaluating, and auditing secure computer systems. Its work provided criteria for the evaluation of computer security functionalities effectiveness. Such criteria were used by the MITRE Corporation [58] to establish a set of security evaluation criteria and a metric useful to measure the security of a system. The effort was later employed by the DoD Computer Security Center that in 1985 introduced a systematic set of standards, and published a set of criteria for developing and evaluating systems. It was called the Trusted Computer System Evaluation Criteria (TCSEC) or "Orange Book". In Europe a different scheme was used for the evaluation of a security product, the Information Technology Security Evaluation Criteria (ITSEC) standard [59].

In order to combine together these two approaches, in 1996 the Common Criteria standard was published. Since 2005 Common Criteria or ISO/IEC 15408 constitutes the international standard to design, analyze, and evaluate security critical systems claiming very high levels of assurance.

The Orange Book contemplated evaluations at a number of classes with A1 being the highest, and moving downwards through B3, B2, B1, C2 and C1. The early MLS products, like Multics, SCOMP, XTS-300/400 etc. were evaluated according to the Orange Book standard at high certification classes (between A1 and B2 levels), and currently the same assurance levels are the target for multilevel systems aiming at the Common Criteria certification. Translating to the Common Criteria, B1 is approximately EAL4, B2 is roughly EAL5, and B3 and higher are EAL6/EAL7⁹.

The equivalence of the security evaluation levels is reported in the following table:

TABLE 2.5: Correspondence of Certification Levels

Common Criteria	ITSEC	TCSEC
EAL1	E0	D
EAL2	E1	C1
EAL3	E2	C2
EAL4	E3	B1
EAL5	E4	B2
EAL6	E5	B3
EAL7	E6	A1

⁹Often a system/products is evaluated according to an EAL "augmented". This expression is stated by a "+" after the assurance level (for example EAL4+). It denotes an augmentation to the EAL, that is the realization of security objectives not required by the corresponding EAL.

2.8.1 ISO/IEC 15408

Since more than 10 years the ISO/IEC 15408 (Common Criteria) became the reference certification methodology for the military and the civilian community. This standard was born as the attempt to unify and harmonize the pre-existing security evaluation criteria like TCSEC, ITSEC, and the Canadian Trusted Product Evaluation Criteria (CTPEC). The first version of Common Criteria was published in 1994 by the Common Criteria Management Board. In 1999 Common Criteria, in the version 2.1, became an ISO/IEC standard: the ISO/IEC 15408. The main added value of such standard, that is the mutual recognition of international evaluations, led the participants to sign an Arrangement on the Recognition of Common Criteria Certificates in the field of IT Security. The other objectives were [60]:

- to ensure that evaluations of Information Technology (IT) products and Protection Profiles¹⁰ are performed to high and consistent standards and are seen to contribute significantly to confidence in the security of those products and profiles;
- to improve the availability of evaluated, security-enhanced IT products and protection profiles;
- to eliminate the burden of duplicating evaluations of IT products and protection profiles;
- to continuously improve the efficiency and cost-effectiveness of the evaluation and certification/validation process for IT products and protection profiles.

Later on, other versions of the scheme followed and the current one is the 3.1 Release 4 that reflects the progress of Common Criteria and the interpretations deriving from past certifications experience.

2.8.1.1 Security Evaluation Phases

An evaluation process involves different subjects: a Certification Authority (CA), an accredited laboratory that involves accredited professionals and performs the evaluation activities, the customer that is s/he who requires the evaluation, and the system provider. The process is characterized by the following steps:

- Preparation,

¹⁰A Protection Profile (PP) is a document that describes an implementation-independent set of security requirements of a class of ICT products.

- Realization,
- Conclusion.

The preparation phase involves the customer, providing the documentation required by the scheme, and a laboratory that evaluates the Security Target (ST) document. A Security Target is the most important document provided by the customer that contains all the assumptions, the requirements and the security objectives the system must fulfill. Through the ST the designated laboratory can issue a detailed plan for the evaluation. Once defined the agreement between the laboratory and the customer, the latter presents the formal request to the Certification Authority delivering also the ST and the evaluation plan produced by the chosen laboratory.

The realization phase starts when the CA, evaluated the received documentation, ratifies the plan and formally accepts the certification in the scheme.

In the conclusion phase the laboratory delivers the Evaluation Final Report that summarizes the results and that is used as reference point by the CA that issues the Certification Report like the example depicted in Fig. 2.1.



FIGURE 2.1: Security Certification Report for Integrity-178B Operating System

2.8.1.2 Evaluation Assurance Levels

“Assurance is grounds for confidence that an IT product meets its security objectives” [61].

The Common Criteria standard provides assurance by an evaluation that aims at determining the security properties of an IT product referred in the scheme as Target

of Evaluation (TOE). In the scheme it is also possible to obtain other combinations of assurance thorough the notion of “augmentation”. Augmentation means that it is possible to add and substitute specific assurance components to those present in the required EAL. Augmentation is represented by a “+” following the obtained certification level.

EAL1 (Functionally Tested) is applicable for systems that must provide evidences of correct operations in environments that encompass low level threats. EAL1 requires a limited Security Target that lists the intended Security Functional Requirements¹¹ (SFR) rather than deriving them from threats, assumptions, and security goals. EAL1 provides the evaluation of the TOE, some independent tests, and the analysis of the provided guidance documents. Main objective of this EAL is the provision of evidences that the system works in a way that is consistent with its documentation.

EAL2 (Structurally Tested) is applicable when the system must face low and mid-level threats. It requires the cooperation of the customer that must provide information about design and test results. EAL2 provides an increased level of assurance by requiring developer testing, a vulnerability analysis, and independent testing based upon more detailed TOE specifications.

EAL3 (Methodically Tested and Checked) is applicable when the customer and/or users require a moderate level of security. This EAL represents a meaningful increase in assurance from EAL2, given that it requires more complete tests of the Security Enforcing Functions (SEF) and procedures providing evidences that the TOE is not tampered during the development phase.

EAL4 (Methodically Designed, Tested, and Reviewed) is the highest level at which retrofitting to an existing product line is cost-effective. EAL4 is applicable when customer or users require a moderate to high-level assured security. This EAL requires deeper design description and improved mechanisms/procedures that provide confidence that the TOE is not be tampered during the development phase.

EAL5 (Semi-Formally Designed and Tested) is the first level that requires to the customer rigorous commercial development practices supported by moderate application of specialist security engineering techniques. It means that the TOE is designed with the intent of achieving EAL5 assurance. EAL5 is therefore applicable when a high level of independently-assured security is required together with a rigorous development approach and security engineering techniques. The significant difference in comparison with the previous levels is that it requires semi-formal design descriptions. In addition, it requires a structured architecture and improved mechanisms/procedures that provide confidence that the TOE is not tampered during the development phase.

¹¹The concepts relating to Security Functional Requirements (SFR) is explained in the next Chapter.

EAL6 (Semi-Formally Verified Design and Tested) is used for high-assurance systems where high value assets are threatened. It requires a deeper description of SFRs, objectives, and security mechanisms. It also requires architectural structure and vulnerability analyses together with a semi-formal design description. Another set of controls is required on the configuration management and development environment.

EAL7 (Formally Verified Design and Tested) is suitable for developing systems to be used in extremely high risk situations and where the high value of the assets justifies higher effort in terms of cost and time. This level requires extensive formal analysis, though the use of formal representations, formal correspondence, and comprehensive testing.

Even if seven EALs are contemplated, another classification can be considered: the first meta-group, comprising the EAL5, EAL6 and EAL7, for which a design semi-formal or formal analysis design is required, and the second one that has not this certification requirement. Such difference plays an important role in the customer choice of the desired certification level. The requirement of system design semi-formal or formal analysis represents a major complication that until now, with the only exception of the Green Hill Integrity 178-B OS, led to the successful certification at high levels of assurance only few solutions. Such solutions are mostly characterized by light firmware or small software components, typical for example in devices like smart-card or Hardware Security Modules (HSM). It becomes very difficult to perform such kind of analysis for complex solutions like Operating Systems. The need of formal verification for high-assurance systems arises from the need, in the evaluation process, to anticipate all possible events that pose risks of adverse consequences. Methods of such analysis can be considered as approximate means to search for a broader class of circumstances violating the established security policy [62].

2.8.1.3 Compositional Approach

A relevant innovation has been introduced by the version 3 of Common Criteria documentation. The latest version of the standard describes the possibility to consider systems previously certified in the scheme to build a certifiable *Composed TOE*.

The concept of using the results of previous certifications belongs to Common Criteria Assurance Continuity paradigm for the maintenance of certification, necessary when the TOE is subjected to changes after a completed evaluation. Following an Impact Analysis Report (IAR), a decision is made whether the TOE must be re-evaluated or not, depending on the level of criticality of the implemented changes.

Starting from this approach, the scheme foresees two different modes:

1. **Composite Evaluation,**
2. **Composed Evaluation.**

A Composite Evaluation can be performed when an independently evaluated product is part of a final composite product to be evaluated. In such composite system, composed by at least two products, the underlying platform must be part of the previously certified component. The EAL of the certified component limits the EAL of the composite system.

A Composed Evaluation can be performed when two or more previously certified products are assembled to build a new system to be certified. For this category, the scheme introduced an Assurance Class called Composition (ACO) characterized by three Composition Assurance Packages (CAP), that is CAP-A, CAP-B, and CAP-C. Each CAP consists of a combination of assurance components and CAP-C represents the maximum achievable assurance level, approximately comparable with EAL4¹². The Composed Evaluation was introduced for those cases when the evaluator possesses only high-level information about the TOE and its evaluated components.

The introduction of ACO class and the potential usage of composite evaluation represents an important evolution of the methodology that fits perfectly the assumptions underlying the new MILS paradigm.

2.9 MILS Projects

Different European Projects have been set up in the last few years. The main goals are the development of solutions for virtualization that could provide strong separation of resources, and the establishment of a common framework for critical system development and certification by means of Common Criteria scheme and formal methods.

The EURO-MILS project (FP7-ICT-2011-8) has been financed by the Seventh Framework Programme and ended at the end of March 2016 [50]. White papers and interesting deliverables are public available on the EURO-MILS website.

The D-MILS is another project of the Seventh Framework Programme (project number 318772) for research and technological development (FP7) that ended in 2013 and whose results are freely available at [63].

¹²CAPs only consider resistance against attackers with an attack potential up to Enhanced-Basic. This is due to the level of design information that can be provided and that affects the rigor of vulnerability analysis performed by the evaluator [61].

Chapter 3

NOVEL MILS ARCHITECTURES

DIFFERENT ELEMENTS contributed to the adoption of the MILS paradigm by Academia and Industry. The triggering event can be attributed to diverse elements: the evolution of processors architectures and separation kernel hypervisors, a deeper comprehension of security certification methodologies, new interpretations of the Common Criteria scheme and the multilevel security problem.

MILS is a mature technology that today more than ever can meet a scheme, the Common Criteria one, that has equally matured thanks to the successes and failures of past certification experiences.

Notwithstanding, currently the marketplace presents only few MILS solutions for military applications and still lacks offering such systems for customer applications.

Many requirements characterized our design approach. Our solution should have been innovative, designed from the scratch in the defense-in-depth concept, security certification-oriented, that made use of the world wide recognized security standards and algorithms, that were characterized by an agile design modeling, and that could have been applied in a transparent way to many legacy architectures.

In order to meet these ambitious goals, we decided to face the problem from different perspectives. For example, concerning the security certification-oriented objective, we decided to use the ISO/IEC 15408 standard as a reference since the starting phase of the project, in order to define punctually the security requirements for a high-assurance system.

The use of the Common Criteria standard, together with an agile design modeling, gave us the opportunity to benefit from the Common Criteria rigorous approach and the rapidly progression of incremental system improvements derived by the agile modeling.

In order to provide functionalities evaluable according to international standards, we also decided to use security algorithms (e.g. encryption, hashing etc.) validated by the National Institute of Standard and Technology (NIST¹) [64].

A general presentation of our design approach and our novel MILS Distributed Architecture is introduced in this chapter. We will illustrate the essential concepts through use cases and the punctual specification of some representative security requirements fulfilled by our solution. Such requirements do not exhaust the totality of the security requirements pinpointed during the design phase and should be considered as examples used to clarify our method. The architecture of a specific component of our MILS distributed solution, that has been fully developed during the PhD studies, is also deeply described in this chapter.

3.1 Proposed MILS Distributed Architecture

In the design of a new architectural solution is important to consider the possibility to use legacy systems and software. The objective is to provide seamlessly at least the core functionalities of the application environment. Nowadays the interconnection of networks implies also that a single operating system together with a set of software applications is no more sufficient to meet the needs of modern agencies and organizations. Modern systems become more and more complex and distributed, and such tendency broadens the possibility of fallibility of computer software. In order to face these issues we designed a new MILS architecture aiming at allowing a soft transition from a preexisting network model to an effective multilevel one.

Our solution places itself as a guard within the security domain to protect the communications among hosts, whose users are characterized by different clearance and need-to-know, and the application servers involved in the provision of the required services. In the engineering phase a generic network architecture has been considered, upon which our MILS solution components have been integrated.

We assume the following system environment for the preexisting scenario:

- existence of a Public Key Infrastructure (PKI) that uses Elliptic Curve Digital Signature Algorithm (ECDSA) certificates and the relating servers,
- presence of a directory service server,
- presence of a computer network protocol that provides centralized user Authentication, Authorization, and Accounting (AAA) service,

¹NIST is the federal technology agency that works with industry to promote and maintain measurement standards. Through the Computer Security Resource Center (CSRC) it provides resources for information security standards, guidelines, and security-related publications.

- PKI and AAA servers take responsibility for the establishment of application level session-key between the involved actors.

Within such architecture we inserted five distinct components, designed ad-hoc to enforce the established security policy. Information is characterized by a sensitivity hierarchy. The architectural overview of the proposed solution is depicted in Fig. 3.1.

The five novel components are:

1. Trusted Front End (TFE),
2. Policy Server (PS),
3. Transitional Secure Server (TSS),
4. MILS Yarn Trusted Host (MYTH),
5. SoftGap.

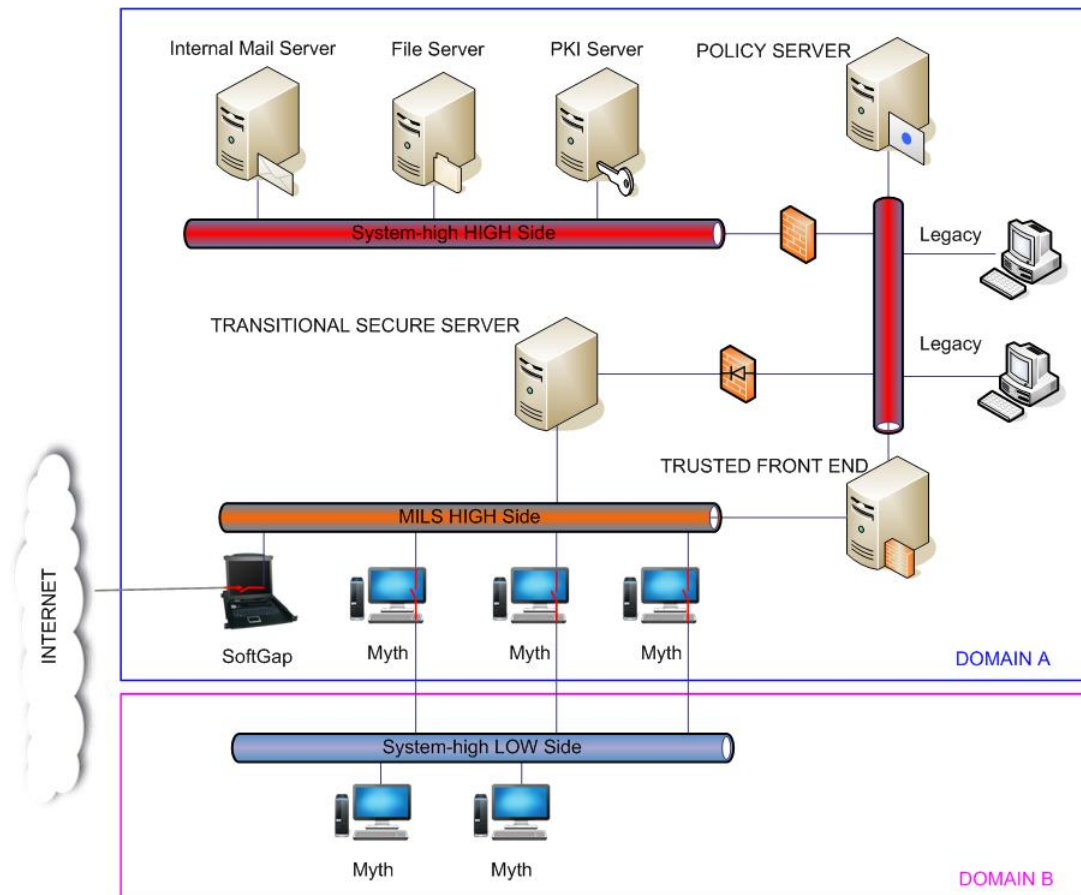


FIGURE 3.1: Novel Distributed MILS Architecture

In Fig. 3.1 two different domains originally working at system-high mode of operations are considered: a high level Domain A and a low level Domain B. Domain A has a

network architecture that foresees a system-high network branch (System-high HIGH Side) on which legacy hosts and servers are connected, and a MILS network branch (MILS HIGH Side) on which our new MYTH and SoftGap components are connected. These two network branches are detached by our main component: the TFE. The PS and the TSS are other new components sited on the System-high HIGH Side network. The SoftGap is the only MILS host that has a connection to the Internet. It is important to underline that the legacy hosts can communicate only with systems connected to the System-high HIGH Side. On the opposite the MYTH hosts are characterized by two operating ways that are mutually exclusive: high-level and low-level duty. In the former mode they can communicate to the System-high HIGH Side through the TFE's mediation, whereas in the latter mode they can exchange data with the low-level domain (Domain B) hosts.

Domain B has a single system-high network (System-high LOW Side) on which legacy hosts are usually connected, even if nonetheless MILS hosts like MYTH and SoftGap can be used in the domain.

3.1.1 MILS Architecture Components

The MILS architecture is composed by distributed components that cooperate together to perform a secure communication among users with different clearances that operate on a hierarchy of information.

The operational role of each component is described in the following paragraphs.

3.1.1.1 Trusted Front End

The Trusted Front End is a high-performance network device that sections the high level domain network acting as a guard of the information flow directed to the application servers. Such flow is composed by the requests that hosts attached to the MILS HIGH Side send to the TFE. This component:

- receives a host request,
- verifies the legitimacy of the request querying the Policy Server,
- forwards the request to the specific server, and
- returns the required data to the host.

TFE is characterized by a software layer that implements the Separation Kernel and Virtualization functionality and enforces a strict control over the operations permitted

to the higher level Operating Systems. Each software layer adds security mechanisms that jointly cooperate to enforce the established security policy.

3.1.1.2 Policy Server

The Policy Server manages the MAC security policy that is set up by the Domain Security Responsible. It provides the TFE all the information concerning the legitimacy of an operation required by a MILS host user. The PS is equipped with a Security Enhanced Linux kernel in order to implement a rigid control over the configuration and the security policy files. In order to speed up the operations, a caching mechanism can be considered relatively to already served requests.

3.1.1.3 Transitional Secure Server

The Transitional Secure Server (TSS) is a specific MILS host that temporally stores the data provided by the application servers following up a host request. It is characterized by different secure partitions, each of them characterized by the classification level defined for the Domain. Such secure partitions are built upon a type-0 Separation Kernel on which ad-hoc security mechanisms operate. Once data are sent in an encrypted form by the application servers to the TSS, it stores them in the appropriate partition for that level. TSS is equipped with two network cards. One of them is connected to the System-high HIGH Side through a hardware firewall that enables only communications towards the TSS. The other network interface is directly connected to the MILS HIGH Side network in order to allow a direct communication path among TSS and the MILS hosts.

3.1.1.4 MILS Yarn Trusted Host

The MILS Yarn Trusted Host (MYTH) are specific machines that provide users with a secure environment to exchange data between two different classification Domains. A MYTH has a Separation Kernel Hypervisor that provides two different classification level partitions and has two different network cards. A user can send/receive data to/from the lower Domain only through the low level Operating System and a low level network card. The usage of the Operating Systems is scheduled in a way that operations are allowed only on the component with the control focus. A third partition is used as temporary repository for the exchanged data. Ad-hoc security mechanisms verify on temporal basis and/or events the integrity of the secure partitions and automatically activate the recovery of the system secure configuration.

3.1.1.5 SoftGap

SoftGap is a MYTH device specifically designed to enforce the unidirectional data flow control security policy. Such component is suitable for every security environment where a connection to an unreliable network is necessary and when data from a low classification level network must be imported in a higher level one enforcing the Bell-La Padula model. As well as TSS and MYTH hosts, the SoftGap machine is equipped with two different network cards. SoftGap can be used as stand-alone component in any legacy network and its architecture is deeply illustrated in paragraph 3.12.

3.1.1.6 Application Servers

An application server in a multilevel environment has to deal with data characterized by different classification levels. In our architecture all servers are sited in the system-high side of the network and so they have the appropriate clearance to deal with all the classified information. Even if the application servers (mail server, file server etc.) are not new elements in the architecture, they need to incorporate software modules that realize the functionalities required to interface the TFE and the TSS. The servers also handle the dispatch of acknowledgments about served requests and the directory path necessary to retrieve the requested data.

3.1.2 Security Requirements

In order to define in a correct and exhaustive way the requirements for our MILS architecture and its components, we used a hybrid approach deriving from the usage of Common Criteria as reference, and from the use of an agile design modeling.

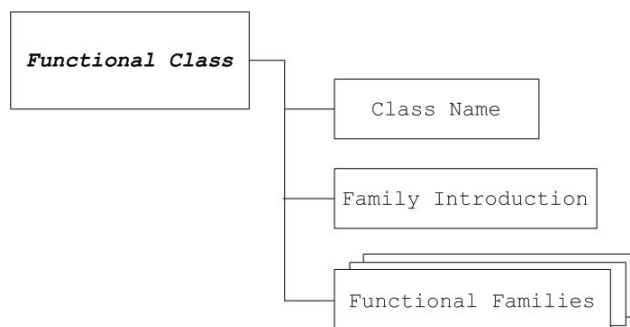


FIGURE 3.2: Functional Class Structure

The Common Criteria standard defines in [65] a complete list of Security Functional Components (SFC) that represent the basis for security requirements and describe the desired behavior of the system that must meet all the claimed security objectives. The

standard use the concept of Functional Classes characterized by a Class Name, a Class Introduction, and Functional Families as depicted in Fig. 3.2.

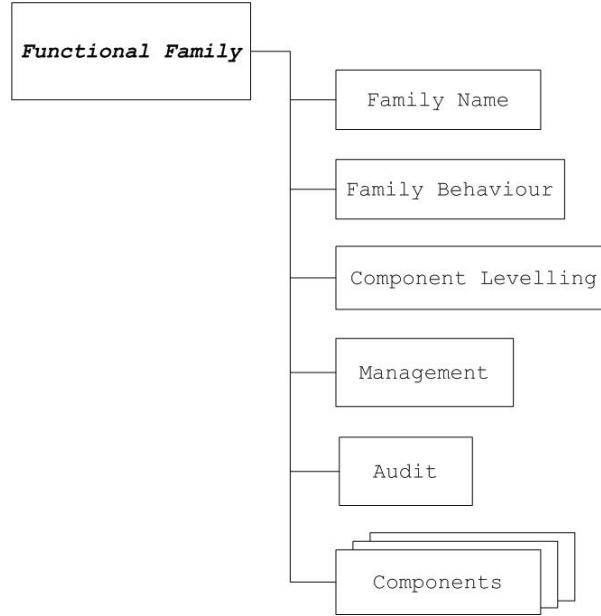


FIGURE 3.3: Functional Family Structure

A Functional Family subclass has the structure detailed in Fig. 3.3 and describes, *inter alia*, the security objectives the Family is addressed to solve and the description of its functional requirements.

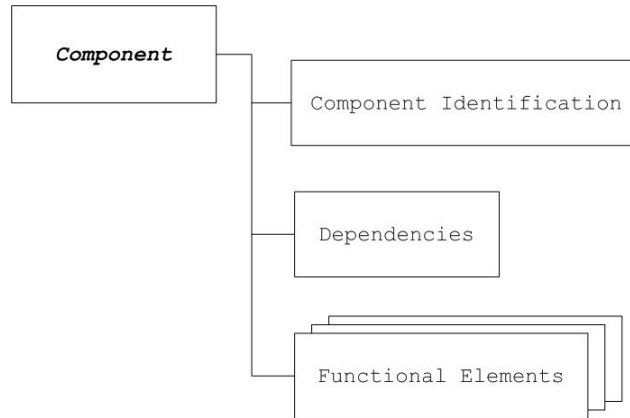


FIGURE 3.4: Security Functional Components Structure

Functional Families also contain one or more Security Functional Components (SFC) which describe security functional requirements that “*if further divided would not yield a meaningful evaluation result*” [65]. The smallest security functional requirement is also referred as Functional Element. The structure of Security Functional Components is depicted in Fig. 3.4.

An example of Functional Class used for our architecture is the FCS (Cryptographic Support). One of its families is Cryptographic operation (FCS_COP) that requires a

cryptographic procedure to be performed according to specific algorithms and keys of specified length.

The usage of the Security Functional Component FCS_COP.1 in the formal documentation required by a Common Criteria certification would be for example:

The TSF² shall perform [assignment: *list of cryptographic operations*] in accordance with a specified cryptographic algorithm [assignment: *cryptographic algorithm*] and cryptographic key sizes [assignment: *cryptographic key sizes*] that meet the following: [assignment: *list of standards*].

The same requirement refined for our MYTH and SoftGap component becomes:

MYTH-FCS_COP.1: The TSF shall perform [*data encryption and decryption*] in accordance with a specified cryptographic algorithm [*AES Galois/Counter Mode*] and cryptographic key sizes [*256 bit*] that meet the following: [*RFC4106 and FIPS PUB 197*].

Instead, the cryptographic requirement for a security function developed on the File Server in our architecture has the following notation:

FS-FCS_COP.1: The TSF shall perform [*data encryption and decryption*] in accordance with a specified cryptographic algorithm [*ECDSA*] with [*Curve P-384*] and cryptographic key sizes [*384 bit*] that meet the following: [*FIPS PUB 186-3*].

We used this kind of notation for all the identified security requirements that our architecture should meet and we used the complete list of Common Criteria SFCs also as reference to identify further requirements.

Some SFCs refined according to our MILS architecture are given as example in the following.

Class FDP (User Data Protection):

- **SG-FDP_ITC.1.1:** The TSF shall enforce the [*unidirectional information flow control Security Function Policy (SFP)*] when importing user data, controlled under the SFP, from outside of the TOE.
- **SG-FDP_ITC.1.2:** The TSF shall ignore any security attribute associated with the user data when imported from outside the TOE.
- **SG-FDP_ITC.1.3:** The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TOE: [*antivirus control rule, imported data integrity rule, security import encryption key rule*].

²TOE Security Functionality, where TOE means Target of Evaluation and represents the system/product under certification.

- **TSS-FDP_RIP.2.1:** The TSF shall ensure that any previous information content of a resource is made unavailable upon the [*deallocation of the resource from*] all objects.

Class FTP (Trusted Path/Channels):

- **TFE-FTP_TRP.1.1:** The TSF shall use [*TLS protocol*] to provide a trusted communication path between itself and the MYTH host that is logically distinct from other communication paths and provides assured identification of its end points and protection of the communicated data from disclosure and detection of modification of the communicated data.
- **TFE-FTP_TRP.1.2:** The TSF shall permit MYTH hosts to initiate communication via the trusted path.
- **TFE-FTP_TRP.1.3:** The TSF shall require the use of the trusted path for initial MYTH host's user authentication.

In conjunction, we joined the Common Criteria requirement definition methodology together with an agile design modeling that foresees the specification of use cases in order to identify and formulate security requirements. The used model, called Unified Process, has been used also for the developing phase of the SoftGap component.

3.1.3 MILS Architecture Use Cases

We defined two main use cases that illustrate the main functionalities of the novel Distributed MILS Architecture. The first one is related to the security objectives fulfilled by the SoftGap component, whereas the second one illustrates the joint cooperation of the other MILS components in order to allow an authenticated user to download a resource from an internal file server. During the design process we defined other use cases that illustrate for example the uploading of a classified file to the network and the exchange of files between users. Since the first two use case are sufficient to explain the functional aspects of our MILS Distributed solution, we will describe them in the following of the thesis.

Use cases represent design artifacts that dynamically change and acquire details according to the evolution of the design phases in the agile developing process. To show the evolution of the architecture that is reflected in different refinements of the use cases, we show in this paragraph a first development of the SoftGap component's sequence of operations. The refined version of the same use case will be detailed in the session of the thesis that deepens the SoftGap architectural solution.

3.1.3.1 Unidirectional Secure Import from Unreliable Network

Name: Unidirectional secure import

Identifier: UC1

Description: this use case specifies the actions performed by the system in order to allow an authorized user to import files from a low level classification network to a high level classification network.

Components: SoftGap

Actors: Authenticated user.

Pre-conditions:

- the subject (i.e. an operating system or a software environment) connected to the low level network foresees an internal user authentication mechanism,
- the high level domain foresees a centralized user authentication mechanism,
- the authenticated user has least privileges on controlled resources,
- the antivirus software components are constantly updated.

Post-conditions: The user imports files from the low level network to the high level network. The information flow is controlled in a way that the security policy of the domain is always enforced. Data are controlled and processed for further dissemination avoiding any flow from the high level network to the low level one.

Basic flow of events:

1. The user logs on to the workstation low level operating system and authenticates himself in order to import a set of files.
2. The system enables the network card connected to the low classified domain.
3. The system provides the minimum functionality required for the retrieval of files on the low level network and implements actions to carry out thorough cleaning (“sanitization”) of the secure local partition (also called local support partition).
4. The user imports a file using a strictly controlled environment (e.g. using an Internet browser).
5. The system saves the file in the local support partition.
6. The “low level” antivirus software checks the imported files.

(The user repeats steps 4-6 until s/he signals the end of operation to the system.)

7. The system performs security activities in order to encrypt and sign the imported files. (MYTH-FCS_COP.1)
8. The system generates an encryption key K and sends it securely to both the Operating Systems.
9. The system encrypts the whole set of files with a symmetric encryption algorithm using the generated key K.
10. The user logs off from the operating system.
11. The system disconnects the network interface card attested to the low classified domain. (SG-FDP_ITC.1.1, SG-FDP_ITC.1.2, SG-FDP_ITC.1.3)
12. The system sends the encrypted and signed set of files to the high level operating system.
13. The user activates the high level operating system and authenticates himself in order to import the downloaded files.
14. The system enables the network card connected to the high classified domain.
15. The system sanitizes the memory used to store files and encryption key.
16. The system performs security activities in order to decrypt and verify the signature of the imported files.
17. The “high level” antivirus software checks the imported files.
18. The user accesses the internal services for the dissemination of the imported files.

Extensions (alternate streams):

- * **a.** At any time the system fails. To ensure the security of data the user administrator performs *Managing Security Subsystems*³ in order to check the integrity of the operating system and its components.

1a, 13a The user fails to authenticate.

1. The user requires the assistance of the user administrator.

6a, 17a The antivirus detects virus/malware.

1. The host deletes the infected file.

³Managing Security Subsystems is another use case not detailed in the thesis.

4a, 5a The user decides to import no files or only some file.

1. The user deletes/deselects files from the application interface.
2. The host logs the event.

5b The user decides to add more files.

1. The user adds the file to the folder.
2. The host runs the antivirus.
3. The host recalculates the digest of the new set of files.
4. The host logs the event.

3.1.3.2 Secure Download from Internal File Server

Name: Secure download

Identifier: UC2

Description: this use case specifies the actions performed by the system components in order to allow an authorized user to download a classified file from the Domain internal file server.

Components: MYTH, TFE, PS, TSS, File Server, and PKI Server.

Actors: Authenticated user.

Pre-conditions:

- the domain foresees a centralized user authentication mechanism,
- a Public Key Infrastructure is already set up in the domain.

Post-conditions: The user downloads the requested file. The information flow is controlled in a way that the security policy of the domain is always enforced. Data are controlled and processed in order to avoid any flow that could subvert the applied security models and protecting data confidentiality and integrity.

Basic flow of events:

1. The user logs on to the specific MYTH host (referred as host in the following) and authenticates himself.
2. The user navigates the local File Server directory list and selects the file to download.
3. The host and the TFE exchange a session key (K_{H-TFE}). (TFE-FTP_TRP.1.1, TFE-FTP_TRP.1.2, TFE-FTP_TRP.1.3)

4. The host sends the encrypted request to the TFE (with session key K_{H-TFE}).
5. The TFE sends a confirmation message to the host (about the taking in charge of the operation).
6. TFE and Policy Server exchange a session key (K_{TFE-PS}).
7. The TFE checks the eligibility of the request by querying the Policy Server (message encrypted with the session key K_{TFE-PS}).
8. The TFE generates an encryption key K used later by the user to decrypt the downloaded file.
9. TFE and File Server exchange a session key (K_{TFE-FS}).
10. The TFE questions the File Server about the file (required by the MYTH user) by sending a message containing the encryption key K encrypted with the public key of the File Server. Before sending the message the TFE calculates the message digest, signs it, concatenates it to the message and sends the concatenation within an encrypted message (with the session key K_{TFE-FS}).
11. TSS and File Server exchange a session key (K_{TS-FS}).
12. The File Server encrypts the required file with the encryption key K and sends it to the TSS in a message that is encrypted with the key K_{TS-FS} , hashes it and signs the digest. (**FS-FCS_COP.1**)
13. The TSS checks the message hash and signature.
14. The TSS sends to the File Server an acknowledgement (encrypted with K_{TS-FS}).
15. The File Server sends a message to the TFE in which it confirms the processing of the request (encrypted with K_{TFE-FS}).
16. The TSS saves the file in a secure partition that is appropriate to the classification level of the specific data.
17. The TFE sends a message to the MYTH user containing the result of the request processing (provided by TSS). The message is encrypted with K_{H-TFE} .
18. The TFE sends the encryption key K to the host (encrypted with the public key of the MYTH user) in a message encrypted with K_{H-TFE} .
19. The MYTH receives the requested resource in encrypted form.
20. The MYTH decrypts K with the private key of the MYTH user.
21. The MYTH user decrypts the file with K .

22. The TSS safely removes the file after a predefined interval of time. (TSS-FDP_RIP.2.1)

Extensions (alternate streams):

- * **a.** At any time the system fails. To ensure the security of data the user administrator performs *Managing Security Subsystems* in order to check the integrity of the operating system and its components.

1a The user fails to authenticate.

1. The user requires the assistance of the user administrator.

3a. The host receives no response from the TFE.

1. The host forwards the request again after a predetermined period of time:
 - 1a.** The host does not receive any message.
 - 2a.** The host notifies the user the impossibility of the action.

7a. The request is not allowed.

1. The TFE sends a notification message to the host.

10a. The requested file is not available.

1. The TFE sends a notification message to the host.

15a. The File Server does not receive a confirmation message from the TSS.

1. The File Server submits again the file after a predetermined period of time:
 - 1a.** The File Server does not receive a confirmation message from the TSS.
 - 2a.** The File Server send a notification message to the TFE about the impossibility of providing the service.
 - 3a.** The File Server removes safely the encrypted copy of the file.

3.1.3.3 Hybrid Security Requirements Correlation

With our hybrid approach, we tied together the formalism required by the ISO/IEC 15408 standard with the agile development modeling (Unified Process) that provides us a rapidly progression of incremental system improvements. In this way, we immediately

insert the certification process from the beginning in the architectural design instead of postponing at the end of it. The solution is not only designed from the scratch in the defense-in-depth paradigm, but also in the Common Criteria certification perspective. The security evaluation “thinking” is pushed in the design process from the beginning. This approach, undoubtedly, will decrease the time and the effort of a security certification, increasing the certification success factor.

In addition, the “divide et impera” approach of MILS paradigm, based on the separation of security mechanisms and issues into independent and manageable components, perfectly adheres to the agile process and simplifies the specification, design and analysis of a multilevel solution.

TABLE 3.1: Hybrid Security Requirements Correlation Matrix

Security Functional Component	Use Case UC1	Use Case UC2
MYTH-FCS_COP.1	X	
FS-FCS_COP.1		X
SG-FDP_ITC.1.1	X	
SG-FDP_ITC.1.2	X	
SG-FDP_ITC.1.3	X	
TSS-FDP_RIP.2.1		X
TFE-FTP_TRP.1.1		X
TFE-FTP_TRP.1.2		X
TFE-FTP_TRP.1.3		X

We could, for example, extend the security requirement correlation matrices required by the Common Criteria standard using some elements created in the artifacts of the agile development process. This “joint perspective” can improve the check of the completeness of the fulfilled requirements. Table 3.1 shows an example of requirement correlation matrix limited to the use cases and the security functional components introduced in the previous paragraphs.

3.1.4 Cipher Suites

In order to address all the security functionalities needed by our multilevel solution and fulfill the security requirements, simultaneously with the use case definition we chose the Cipher Suites (CS) to be used. A CS comprises a set of protocols and cryptographic algorithms on which agree the participants of a communication in order to:

- share a secret key,
- encrypt messages,

- generate hashes of messages, protect the integrity of messages and provide source/destination authentication.

A CS defines the algorithms used to realize the aforementioned tasks and it is usually defined in a form like the following: `RSA_WITH_AES_256_CBC_SHA`. This example shows that the communication is protected by using RSA for key sharing, AES 256-bit key with CBC mode for encryption, and SHA for message authentication. Many combinations can be chosen according to specific needs and requirements. Our intention was to create different security layers in our Distributed MILS Architecture, in order to protect the classified data in the defense-in-depth paradigm. In particular we decided to use IPsec and IKEv2 protocols at ISO/OSI level 3 and TLS protocol at a higher level of the ISO/OSI model. For each of these layers we proposed a specific cipher suite according to the following considerations:

1. compliance with international rules established for the treatment of classified information,
2. differentiation of security mechanisms and algorithms.

According the first point we decided to adhere to the NSA-approved Cipher Suite B. It has been defined to protect foreign releasable information, US-Only Information and Sensitive Compartmented Information (SCI). In [66] the NSA Committee on National Security Systems establishes which standards may be used for cryptographic protocol and algorithm interoperability to protect National Security Systems (NSS). In particular, in [Appendix B](#), are detailed the cryptographic algorithms approved by NSA with the level of classifications they are supposed to operate with.

We also decided to use different CSes for IPsec and TLS protocols, in order to avoid that, in case of *zero-day* protocol security vulnerabilities, a security breach could invalidate our security chain. After Snowden's revelations about the NSA and its international partners' Global Surveillance, we also decided to modify the Cipher Suite suggested by the United States National Security Agency with new algorithms. The new algorithms are ChaCha20, a new high-speed stream cipher and Poly1305 that is a high-speed message authentication code. ChaCha20 has naturally very good performance and with some optimization, as presented in [67], it became a fast and secure alternative to AES in the TLS protocol. It is proposed in RFC7539 [68] together with the use of Poly1305, both as stand-alone algorithms and as a "combined mode", or Authenticated Encryption with Associated Data (AEAD) algorithm. Such algorithms have been recently adopted by Google.

For the aforementioned reasons we chose TLS v. 1.2 together with the cipher suite:

TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA384

where Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) and Elliptic Curve Digital Signature Algorithm (ECDSA) use Curve P-384 to protect up to TOP SECRET information, and the ChaCha20 algorithm uses 256-bit keys.

In relation to IPsec and IKE protocols we opted for different algorithms as listed below.

- IPsec v.3:
 - Transport Mode,
 - Encapsulating Security Payload (ESP),
 - AES-NI-256-GCM with 128 bit Integrity Check Value (ICV).
- IKEv2:
 - AES-NI-256-CBC⁴ (encryption),
 - HMAC-SHA-384 (pseudo-random function),
 - HMAC-SHA-384-192 (integrity),
 - 384 bit Random ECP Group (Diffie-Hellman group).

where ECDSA is used for signature.

An interesting resource is [73], that shows the status of recommended elliptic curve domain parameters conformant with standard protocols.

3.2 Evaluation

After introduced our novel Distributed MILS Architecture, our design approach and the chosen cipher suites, we now present a performance evaluation of the solution. Our goal is to show how the architecture enforces the Bell-La Padula model and to determine the latency costs due to the application of different multiple security layers.

⁴Some specific attacks have been presented for AES 256-bit key in [69, 70, 71]. A famous security professional suggests people do not use it. According to him, AES-128 provides more than enough security margin for the foreseeable future [72]. Also, AES-128 is faster than its 256-bit key homologous. Even if the speed usually is desirable for such algorithms, a slower one could be considered more secure against a brute-force attack. Even if the aforementioned attacks are better than brute force, they are still far beyond our capabilities of computation. When we consider Quantum Computation, that theoretically can perform several computations simultaneously, an attack to n -bit symmetric cryptography (e.g. symmetric encryption with a n -bit key) can be completed in $2^{n/2}$ elementary quantum operations. To account for quantum computers it would be a reasonable choice to double the key length. Hence, AES with a 256-bit key.

In order to evaluate our solution, that is quite complex because of its distributed nature and the presence of multiple levels of encryption built upon a Separation Kernel Hypervisor, we decided to operate in two different ways. We first performed a theoretical evaluation of the overall MILS architecture considered as a proof-of-concept, in order to analyze the security goals and its general performance. Thereafter, we built a prototype of a specific component of the architecture, the SoftGap, that we further tested with security protocol model checkers and vulnerability assessment and penetration tests.

3.2.1 Security

In this section, we first define the considered attack model then, we perform a security analysis of our solution. We refer to the use case [UC2](#). Later on, in the SoftGap prototype description, we will perform a security evaluation specifically for that component.

3.2.1.1 System Model

According to the system user we assume that:

- administrators are non-hostile and appropriately trained,
- each user has public/private key pair (provided by smartcard/token),
- no more than one user can work concurrently on a computer machine.

According to the system environment we make the following assumptions:

- MILS components with two network cards can connect to different subnetworks only in an *exclusive-or* way (only one card can be connected at any time),
- the Separation Kernel Hypervisor (SKH) used to virtualize the computer environment is secure⁵,
- the algorithms for digest, signature, symmetric and asymmetric encryption are secure.

The assumptions for the preexisting scenario defined in Par. [3.1](#) are still valid in our system model.

⁵ It is designed to satisfy the Common Criteria SKPP v1.03 [\[45\]](#) and is based on a security enhanced version of a Separation Kernel [\[48\]](#)

3.2.1.2 Attack Model

Different kind of attackers are considered in our scenario:

- external attacker,
- insider that wiretaps the network,
- insider with a certain access to the network,
- insider that gains access to an authorized user smartcard/token,
- authorized user trying to access data on which s/he does not have the proper clearance and need-to-know.

We also consider network attacks combined with the introduction of malware components to the targeted systems.

The main threats are the disclosure of information outside the protected domain (data exfiltration), the unauthorized access to data by personnel that has not the proper clearance and need-to-know, the accidental disclosure of data due to computer machine maintenance.

3.2.1.3 Security Analysis

Our distributed architecture foresees the exchange of different application layer messages in order to achieve the goal described in the use case UC2. These messages are used to define and perform the service required by the user, i.e. the download of a specific resource stored on a file server.

The first actor involved in our use case is MYTH that is in charge of authenticating the user through a preexisting Authentication, Authorization and Accounting Server, a smartcard/token, and credentials like username-password. This multi-factor authentication is necessary to avoid that an insider, even gaining access to one of the user's authentication components, could not gain access to the system resources.

The confidentiality of data is provided by different layers of encryption. In relation to the use case UC2, the resource required by a MYTH user is encrypted in the application layer before being sent from the file server to the Transitional Secure Server. On TSS the resource remains encrypted so that, even if an attacker could gain access or compromise that host, the confidentiality of data is still safeguarded. The same resource is encrypted on the MYTH and can be read only by the user that required it. In fact the encryption key K , used to encrypt the resource, is encrypted with the user public key. If a system crash occurs during the resource transmission, no clear data is present on

that machine. All the MILS hosts are characterized by a sanitization mechanism that deletes in a secure way sensitive data on the machine memory. It guarantees that the downloaded resource is securely deleted after being read and only the original encrypted one can be locally stored.

Also the confidentiality of messages exchanged by all the MILS components is protected by the combined use of TLS and IPsec protocols. The IPsec protocol is used in Transport Mode ESP that provides a secure end-to-end communication with protection of confidentiality, authentication, and integrity. In our solution the IP payload is protected by the TLS protocol, that adds another layer for identification, authentication, confidentiality, and integrity of the upper layer data. The use of these protocols reduces the ability of an insider to wiretap the network, modify data payloads, impersonate a host, and exploit man-in-the-middle techniques in order to access information illicitly. The use of ECDHE provides also the relevant feature called Perfect Forward Secrecy.

In our application layer protocol, the TFE sends the filepath necessary to download the requested resource from the TSS to the MYTH. In order to avoid the possibility of guessing a correct path by an insider, we applied two different countermeasures. The TFE sends the filepath and the digest of the required resource calculated by the file server to the MYTH. Only the originator of the request can have the proper path and file digest in order to retrieve the resource. We also included in the message an expiration time (*exp_time*) that defines a time window for the download of data. After this period of time, the resource becomes no more available.

All the MILS hosts that are based on a Separation Kernel Hypervisor and different guest operating systems foresee a security feature that, in case of system failure, reboots the VMs, checks their integrity, and restores a secure configuration state.

All the events are recorded by an Audit & Log mechanism.

The Separation Kernel Hypervisor performs a robust partitioning through isolation of memory, I/O interfaces, CPU runtime, etc. and preventing that low-clearance components could access resources of high-clearance ones. Different classification level data flows are processed in a separate way and cannot interfere with each other. The SKH also provides an independent runtime environment on which high-assurance security mechanisms are built. These mechanisms run directly on the CPU cores without relying on the assistance of the guest operating systems.

Our architecture foresees a hardware firewall used to enable a one-way communication from the System-high HIGH Side network towards the TSS. Together with the *exclusive-or* mechanisms used to enable the network cards present on the TSS machine, the firewall avoids the possibility of direct communications between the MILS HIGH Side and the System-high HIGH Side subnetwork.

The SoftGap is the single point of access from the Internet for attackers. The security evaluation for this component will be presented in the [SoftGap](#) paragraph.

3.2.2 Performance

The performed security analysis showed that the confidentiality of exchanged data is guaranteed by different security mechanisms that overlap and intersect each other. The distributed nature of the solution requires the exchange of messages between components that affects the architecture performance. The multiple encryption layers contribute increasing the latency of the overall solution. We developed a theoretical model in order to determine such latency as described in the following paragraphs.

3.2.2.1 Message Overhead

In order to calculate the message overhead introduced by our novel architecture in the path between host and file server we refer to the use case [UC2](#), where all the MILS devices are necessary to perform the required operations. In addition to the assumptions made in paragraph [3.2.1.1](#) we consider the following:

- no switch/router is considered in the proof-of-concept,
- there are two kind of application layer messages: the ones related to the protocol operations (*operational_message*) and the one used to transmit the requested file (*file_message*),
- no retries, packet losses or other events occur,
- TCP and IP packet headers of 40 Bytes (no TCP options),
- use of a full-duplex communications medium, i.e. the full bandwidth is available both upstream and downstream, at the same time.

We also assume that in the scenario where no MILS architecture is implemented (referred as **scenario A**), the resource is encrypted by the file server before being sent to the user's machine and the user's clearance is verified querying a Policy Database. Also the application layer messages are encrypted using the provided session-keys. Because such operations are also performed in our novel architecture, the latency introduced by them is not considered in both scenarios.

In scenario A, the download operation foresees two high level messages.

host - file server:

host → *file server*: *send(download_resource)*

file server → *host*: *reply(encrypted_file)*

host → *file server*: *send(ack)*

Hence, we have:

- number (#) of operational_messages: 2
- number (#) of file_messages: 1

When our MILS Distributed Architecture is implemented (**scenario B**), we have an increased number of exchanged messages between MYTH host (referred to as host in the following) and TFE, TFE and PS, TFE and file server, file server and TSS, and between host and TSS, respectively.

host - TFE:

host → *TFE*: *send(service_request)*

TFE → *host*: *send(ack)*

TFE → *host*: *send(filepath)*

host → *TFE*: *send(ack)*

with:

- # of operational_messages: 4
- # of file_messages: 0

TFE - PS:

TFE → *PS*: *send(verification_request)*

PS → *TFE*: *send(verification_result)*

TFE → *PS*: *send(ack)*

with:

- # of operational_messages: 3
- # of file_messages: 0

TFE - file server:

TFE → *file server*: *send(service_request)*

file server → *TFE*: *send(ack)*

with:

- # of operational_messages: 2
- # of file_messages: 0

file server - TSS:

file server → *TSS*: *send(file)*

TSS → *file server*: *send(filepath)*

file server → *TSS*: *send(ack)*

with:

- # of operational_messages: 2
- # of file_messages: 1

host - TSS:

host → *TSS*: *send(file_request)*

TSS → *host*: *send(file)*

with:

- # of operational_messages: 1
- # of file_messages: 1

The total message overhead is:

- # of operational_messages: 12 (+500%)
- # of file_messages: 2 (+100%)

Actually, to compute the message overhead between host and file server we have to consider that in our novel architecture the network and its relating traffic is split by the presence of the TSS and TFE machines. Because of the TSS double network cards, the total traffic generated on the MILS HIGH Side subnetwork (host-file server path) is composed by five operational messages (+150%) and one (+0%) file message. The remaining part of the generated traffic affects the System-high HIGH Side subnetwork and does not limit the bandwidth of other MYTH hosts' users. Also, it could be possible to decrease the System-high HIGH Side subnetwork overhead by caching already-made security policy checks and/or inserting the PS directly as a virtualized subject within the TFE.

3.2.2.2 IKE and TLS Message Overhead

In addition of the aforementioned overhead, IKEv2 (in the following referred as IKE) and TLS protocols add their own message overhead. It is caused primarily by the handshake messages exchanged to negotiate and agree upon the cipher suites to be used [74].

IKE performs mutual authentication between two parties and establishes a Security Association (SA) that includes shared secret information used in the IPsec Encapsulating Security Payload (ESP). The traffic overhead from IKE handshake comprises two types of message exchange. In the so-called `ISA_SA_INIT` Exchange the peers negotiate cryptographic algorithms, nonces, and do a Diffie-Hellman exchange. It usually foresees 4 messages. `IKE_AUTH` Exchange performs mutual authentication of the peers that establish a child-SA through the exchange of digital signed certificates (e.g. RSA or ECDSA authentication). After this initial handshake, additional requests can be initiated and consist of either informational messages or requests to establish another child Security Association. All the messages exchanged in `ISA_SA_INIT` and `ISA_SA_AUTH` are encrypted according to the negotiated security parameters and algorithms [75]. The overhead generated by the TLS protocol pertains to the handshake necessary for the host's authentication (at least the server must authenticate itself) and the agreement of a specific cipher suite. The TLS foresees two protocol layers: the *TLS Handshake* and the *TLS Record*. The former is used to negotiate the cipher suite and to authenticate the peers, whereas the latter is used to encapsulate various other higher protocols and to provide all the mechanisms for connection security.

The TLS Handshake Protocol involves the following steps [76]:

- exchange hello messages to agree on algorithms, exchange random values, and check for session resumption;
- exchange the cryptographic parameters to allow the hosts to agree on a pre-master secret;
- exchange certificates and cryptographic information to allow hosts mutual authentication;
- generate a master secret from the pre-master secret and exchanged random values;
- provide security parameters to the record layer;
- allow the hosts to verify the correct security parameters calculation and that the handshake occurred without tampering by an attacker.

The set of messages of the TLS handshake protocol represents the TLS message overhead present in our solution. In particular, using Elliptic Curve Diffie-Hellman to exchange ephemeral keys, the handshake is slightly different and consists of the following steps. Client and server share basic protocol information via the *ClientHello* and *ServerHello* messages, then the server sends its certificate to the client. A *ServerKeyExchange* message, containing either the parameters of a DH group or of an elliptic curve, paired with

an ephemeral public key calculated by the server, is also sent to the client, followed by a *ServerHelloDone* message that signals that no further messages will be sent until the client responds. The client computes an ephemeral public key compatible with the negotiated parameters and sends it to the server. Knowing their private keys and the public key of other machine, both sides share the same pre-master secret and can derive a shared master secret. The mutual, or at least the server authentication is realized by certificates signed by a Certification Authority. To authenticate the connection the server signs the parameters contained in *ServerKeyExchange* with the its private key. The client verifies the signature and only then proceeds with the handshake.

3.2.2.3 Further Overhead Elements

Other elements contribute to the overhead of our solution. They can be led back to several different groups:

Latency and processing overhead from IKE handshake

Processing overhead from IKE

Traffic overhead from IPsec ESP Transport Mode

Latency overhead from IPsec ESP Transport Mode

Processing overhead from from IPsec ESP Transport Mode

Latency overhead from TLS handshake

Traffic overhead from TLS record layer

Processing overhead from TLS handshake

Processing overhead from TLS record layer

The significance of these elements is directly connected to the choices made during the IKE and TLS handshakes, through which the cipher suites are negotiated and agreed between hosts for securing the communications. In such negotiations it is very important to choose the most robust security algorithms, and above all, the length of the cryptographic keys that plays a very important role as security parameter. In [77] it is possible to see a comparison of international organizations' recommendations of the minimum key size requirements.

Looking at the certificates used in the authentication phases of our IKE and TLS protocols, currently most of the involved organizations suggest an RSA key length of 2048 bits. It represents an important factor of traffic and processing overhead. According to the German Federal Office for Information Security (BSI) [78] and [79], a 2048-bit RSA key gives is equivalent to a 112-bit symmetric encryption key and to 224-bit elliptic curve algorithm key. Such considerations, their related performance issues, and our security

goals drove our decision of inserting very specific cipher suites in our MILS Distributed Architecture. In particular, in order to provide very high security algorithms without deteriorating the performances of the hosts and the entire network, we decided to opt for a fast encryption algorithm (i.e. AES-NI) and for Elliptic Curve Cryptography for both IKE/IPsec and TLS.

AES-NI is a new instruction set extension developed by Intel to optimize AES implementations. The new six instructions allowed Intel to significantly improve IPsec performances of a new AES-NI driver in Linux (about 400%) compared to a non-AES-NI enabled software solution implemented on the same platform [80]. In the same document the boost of performance of an IPsec tunnel is presented (see Fig. 3.5).

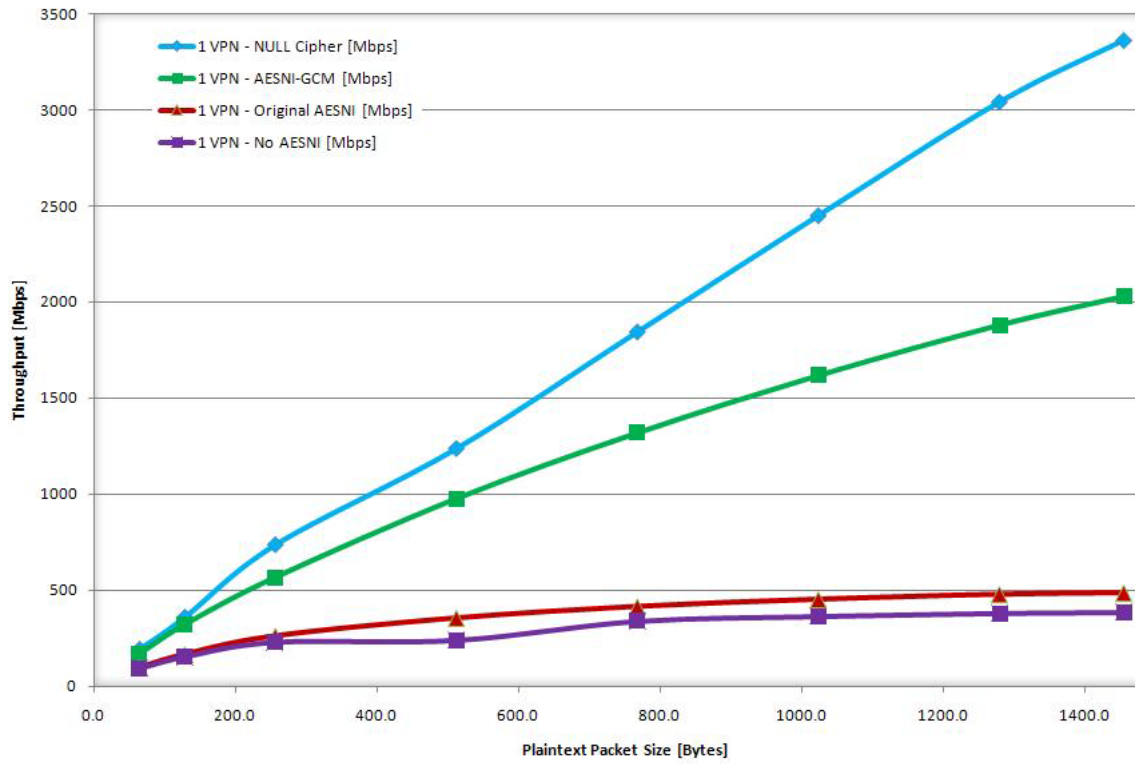


FIGURE 3.5: Performance in Mbps for an IPsec Tunnel with AES-NI

They show a substantial improvement of the overall performances due to the fact that with AES-NI, both the original Linux AES-NI crypto driver and the new AES-NI developed by Intel, the majority of hosts CPU core cycles are spent in the non-crypto portion of the workload. The latency due to AES-NI operations can be then considered negligible in our evaluation.

Elliptic Curve Cryptography (ECC) is a new approach of asymmetric cryptography that is based on elliptic curves over finite fields. It can be used for Diffie-Hellman key exchange and also for digital signature. We used the Elliptic Curve Digital Signature

Algorithm (ECDSA) both in the IKE and on the TLS protocols in order to take advantage of their peculiarities.

One of the advantages of such cryptography is that it allows smaller keys compared to traditional public-key cryptography (i.e. RSA) in order to provide an equivalent level of security. Fig. 3.6 provides comparable minimum bits of security for ECDSA key size and message digest algorithms [81]. *Ceteris paribus*, considering a 2048 bit RSA key, an ECC key is only the 11% of an RSA key length. In [82] the authors show that TLS server throughput can be increased by 11% to 31% using ECDSA over RSA. In [83] it is shown that, for **equivalent** security levels, ECDHE-192 outperforms RSA-1776 key exchange. [84] observes that ECDHE-ECDSA performs faster than RSA key exchange and signature and that an ECDHE-based key exchange can be faster than basic RSA 2048-bit key exchange.

Minimum Bits of Security	ECDSA Key Size	DH/DSA/RSA	Message Digest Algorithms	Curves
80	160–223	1024	SHA-1 SHA-224 SHA-256 SHA-384 SHA-512	sect163k1 secp163r2 secp192r1
112	224–255	2048	SHA-224 SHA-256 SHA-384 SHA-512	secp224r1 sect233k1 sect233r1
128	256–383	3072	SHA-256 SHA-384 SHA-512	secp256r1 sect283k1 sect283r1
192	384–511	7680	SHA-384 SHA-512	secp384r1 sect409k1 sect409r1
256	512+	15360	SHA-512	secp521r1 sect571k1 sect571r1

FIGURE 3.6: Minimum bits of security comparison

Table 3.2 shows a comparison between ECDSA and RSA signature [85].

TABLE 3.2: Comparison between ECDSA and RSA signature

ECDSA	RSA
Keys require less bandwidth	Keys require more bandwidth
Signatures are smaller	Signatures are bigger
Verification is little slower	Verification is little faster

Another extremely important feature deriving by our cipher suite choice is the so-called Perfect Forward Secrecy (PFS). By the use of ephemeral Diffie-Hellman and elliptic curve Diffie-Hellman key exchange, the hosts involved in our novel architecture will generate a new set of Diffie-Hellman parameters for each communication session. It means that the compromise of a single session key will not affect any data other than those exchanged

in that specific session. No past encrypted communications can be read even if the private key of the server is stolen. This use of ephemeral Diffie-Hellman and elliptic curve Diffie-Hellman leads to compute new keys for each new session, but this limiting factor is balanced from the reduced key size and from the enhanced level of security provided by our solution.

In order to define an order of magnitude of latency and processing overheads, we decided to perform a theoretical evaluation supported by measurements of the delay introduced by these security protocols present in the Literature. In particular, since TLS and IPsec/IKE work on different layers, we can assume in our evaluation that the delay of both protocols is additive.

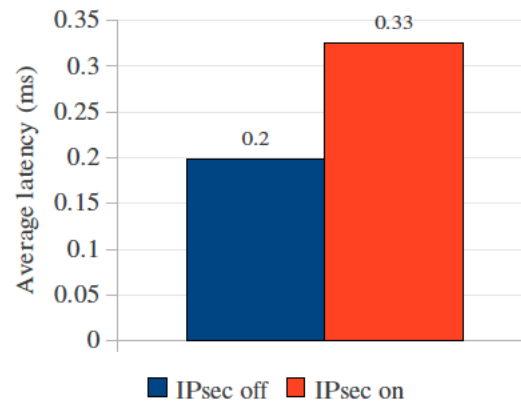


FIGURE 3.7: Connection latency with communication within a domain

Many research works provide data on latency and overhead of IPsec. Such latency and the protocol processing overhead depend on the chosen cipher suite, and obviously on the performance of the machine hardware resources (CPU, RAM, switches, kernel and configuration). For example in [86] the speed performance of a file transfer operation within a single domain with and without IPsec implementation is presented. It is based on [87] that used IPsec ESP tunnel mode with AES-128 bit key in CBC mode, HMAC-SHA1, and public key authentication with IKE. In Fig. 3.7 are depicted these values, that are approximately 0.33 ms for the latency, whereas the throughput is 321 Mbit/s with single connection and 304 Mbit/s for multiple connections (see Fig. 3.8).

Another paper ([88]) shows the average delay for IKE/IPsec Security Association setup in ESP mode that, depending on the specific used cipher suite, is in the range between 896 ms and 1461 ms. Authors in [89] present analogous results that are comprised between 615 ms and 719 ms (IKE + IPsec). The order of magnitude of IPsec throughput is confirmed by [90] that shows the results for a specific IPsec implementation (AES-CBC mode) running on an IBM System x3550 with quad cores. The paper underlies the big improvement of IPsec performances when AES-NI hardware is used. The throughput with AES-NI is about 865 Mbit/s against the 480 Mbit/s without AES-NI kernel.

The throughput is also dependent on packets size. In [91] the results about IPsec VPN connections using ESP with CBC-AES-192 bit key with HMAC-SHA1 authentication on a Gigabit network are presented. The throughput value is aligned with data shown in the aforementioned works and the best performance is obtained with packet size greater than 500 bytes (see Fig. 3.9). White paper [92] shows the performance of different hardware accelerated IPsec implementations (see Fig. 3.10). Another interesting data source is [93] that confirms the previous data and gives a comparison of performance of IPsec implemented using different cipher suites.

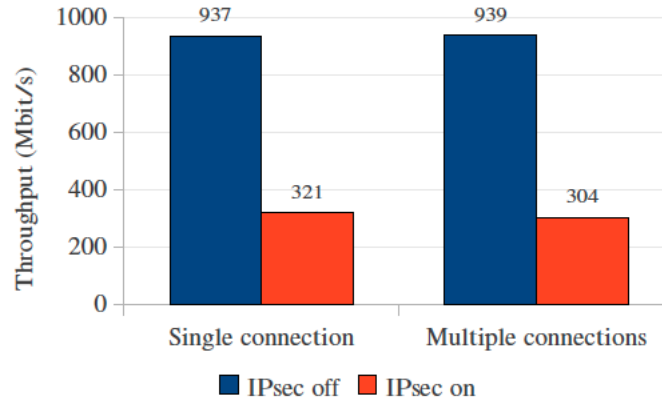


FIGURE 3.8: Throughput with and without IPsec with a single TCP connection

The main overhead of TLS is the handshake. Here the expensive asymmetric cryptography plays a big role. After negotiation, relatively efficient symmetric ciphers are used. In [84] the authors show that the latency due to the setup phase of a TLS connection using ECDHE-ECDSA (with 256-bit key and SHA-256) over the Internet is comprised between 3300 ms and 7600 ms. In [94] the computational overhead using TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 in or low-rate wireless personal area networks is about 2020 ms. Authors of [95] show that the latency of TLS handshake using ECDH-ECDSA with 224-bit key in a heterogeneous wireless sensor network is about 5530 ms and 12000 ms without and with client authentication, respectively. The portion of time used in the TLS handshake process for ECDHE and ECDSA computations is illustrated by [96]. The authors performed different tests using various ARM Cortex-M processors and various elliptic curves. For example the calculation performed on a Cortex-M3 processor using the curve secp224r1 (security level equivalent to a 2048-key RSA) with NIST optimization takes about 717 ms. Such value becomes 1194 ms using the curve secp384r1 necessary to operate in our architecture on TOP SECRET information.

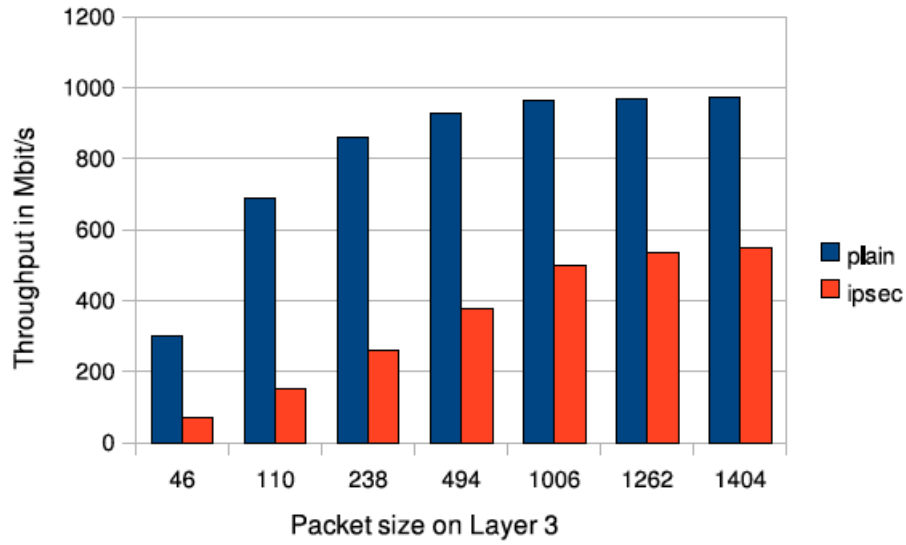


FIGURE 3.9: Performance of IPsec tunnel at different packet size

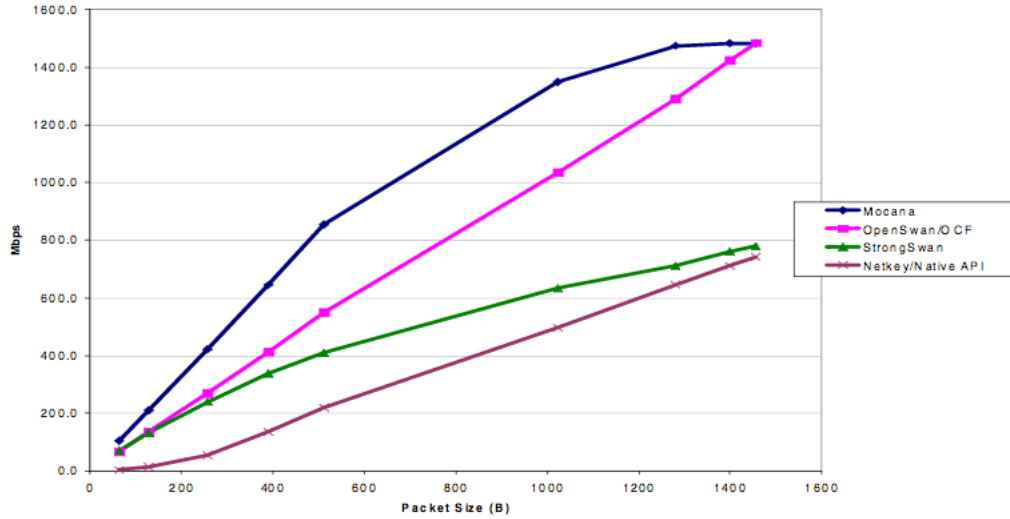


FIGURE 3.10: Performance of IPsec hardware accelerated implementations

3.2.2.4 Message Size Overhead

As shown in [97] many measurement studies before 2004 describe the Internet packet size distribution is trimodal. More recently works [98, 99, 100] found that the distribution of packet size became bimodal with 44% of high number of small packets (less than 100 bytes) and 37% of very large packets (1300 bytes). The large packet size of the distribution is connected to the fact that this value is the MTU recommended one by VPN vendors in order to stay within ethernet MTU after VPN encapsulation (see [101]).

The application protocol defined for our architecture foresees the following data to be exchanged by the MILS components:

- user: 30 bytes
- id: 16 bytes
- service: 2 bytes
- filename: 255 bytes
- nonce: 16 bytes
- timestamp: 8 bytes
- reply: 1 byte
- exp_time: 8 bytes
- digest: 48 bytes
- filepath: 4096 bytes
- file encryption key: 64 bytes
- level: 1 byte
- compartment: 1 byte
- file: n bytes

In our architecture the maximum message size at the application layer is comprised between 25 bytes and $16 \cdot n$ bytes, where n is the size of the downloaded file. In the MILS HIGH Side subnetwork the message sizes are instead comprised between 41 and $16 \cdot n$ bytes.

Because of the presence of the TLS protocol, we have to consider its overhead. In Fig. 3.11 is shown the TLS Record that foresees TLS header, followed by an Initialization Vector/Nonce, the encrypted content, a message authentication code and a padding.

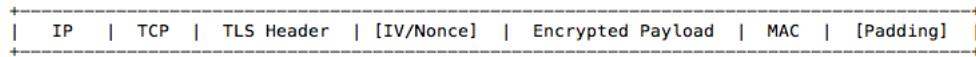


FIGURE 3.11: TLS Packet Structure

For our Cipher Suite that use ChaCha20_Poly1305 the per-packet overhead consists of 5 bytes TLS header, 8 bytes explicit nonce and 16 bytes for the Poly1305, that means 29 bytes introduced by the use of TLS protocol [102].

To Transmit the original 25 bytes of data, we have first to add the 29 bytes due to TLS overhead and then add the overhead relating to the IPsec protocol. It means that transmitting $25 + 29 = 54$ bytes using IPsec, we have to add (see [103, 104, 105]):

4 Bytes for the Security Perimeter Index (SPI)

4 Bytes for the Sequence Number Field

12 Bytes for the Initialization Vector⁶

0 Byte for AES-GCM padding

1 Byte for the Pad Length

1 Byte for the Next Header

16 Bytes for the ICV

Total packet size (minus TCP/IP headers) is now: 92 Bytes \rightarrow an increase of about **268%** respect to the original payload.

Transmitting 1300 bytes of data using TLS together with IPsec implies the addition of:

4 Bytes for the Security Perimeter Index (SPI)

4 Bytes for the Sequence Number Field

12 Bytes for the Initialization Vector

1 Bytes for AES padding to reach the 16 Byte AES block size

1 Byte for the Pad Length

1 Byte for the Next Header

16 Bytes for the ICV

Total packet size (minus TCP/IP headers) is now: 1368 bytes \rightarrow an increase of about **5%** respect to the original payload.

In our architecture all application level messages, except the one considered in the previous paragraph and the acknowledgment ones, are bigger than 300 bytes. Hence, we can estimate that the bimodal packet distribution is shifted to the bigger percentage of large packets described in [98, 99, 100].

Following the aforementioned theoretical evaluation, we can speculate some results that we could expect from our MILS Architecture for a generic use case UC2. We can notice that most of data provided by the Literature that we used for our evaluation, are referred to specific networks with low performances and often the devices used for computations and testing are not so well-performing because, for example, the specific paper topics were Wireless Sensor Networks, Internet of Things, and so on. In a real scenario of usage, where is important to protect classified information providing fast dissemination of data, networks and devices would be highly performing. It means that we can assume that the values provided by the aforementioned Literature constitute an **upper bound** that

⁶NIST Special Publication 800-38D [106] suggests to use 12 bytes for the Initialization Vector

we can use as reference to infer some performance. Such statement is also supported by the fact that none of those papers using AES algorithm implemented AES-NI for crypto-operation, that is much more faster than the implementations without hardware optimization.

According to the latency introduced by the IKE/IPsec and TLS protocols, we can then suppose for a single user session:

- max IKE handshake latency: $\sim 1000\ ms$,
- max TLS handshake latency: $\sim 5000\ ms$.

These values are calculated considering the average latency measured in the papers introduced in the previous paragraph for each protocol (IKE/IPsec/TLS), and choosing the biggest ones as upper limits. The upper bound of the delay due to TLS, IKE and IPsec is about 6000 ms in case the architecture is in an operating state, that is, all the secure connections among the MILS devices have been already setup with the exception of the MYTH-TFE one. We do not consider the delay due to the application layer key exchange performed in conjunction with PKI and AS servers, because in our assumptions it is present also in the scenario without MILS architecture (scenario A). We have also to consider the time necessary for the exchange of the application level messages foreseen by our solutions, that can roughly amount to some hundreds of ms.

An extra delay has to be considered if the architecture is in an operating state but the MYTH performs its operation after, for example, a start up. Such delay is due to the IKE and TLS handshakes necessary to set up the secure connection with the TSS. Its upper bound is the same one taken into consideration for the MYTH-TFE path.

The worst scenario is in the setup phase of our MILS Architecture, where all the devices need to perform the operations necessary to the working of the IKE/IPsec and TLS protocols. Such operations, however, can be made in parallel for some elements.

Our novel architecture introduces overhead elements that decrease the performance of the network particularly in the communication setup process. But for the specific use case UC2, that does not require real-time or near real-time services, such overhead is still acceptable for the user. As seen in the previous paragraphs the overhead introduced by TLS and IPsec for big packet size is around the 5% more than the original scenario. The download operation does not introduce big delay in the file transmission component respect to the scenario A. It has impact on the time necessary to start the download operation that, in an ordinary network, occurs no later than after 6-7 seconds.

3.3 SOFTGAP: a novel MILS Cross-Domain solution

Once defined our MILS Distributed Architecture, we decided to start the detail design of the SoftGap component that fulfill the requirements stated in use case UC1. As already explained in the previous Chapters, such component is very important for many contexts that currently enforce the unidirectional information flow control policy though systems like Air-Gap and/or Data-Diode.

In distributed multilevel systems the cross-domain solutions (CDS) offer the security mechanisms to allow controlled information flows among different security levels domains. A CDS can allow two different modes of data transfer:

- unidirectional, and
- bidirectional.

In the first mode, information is transferred from a low level domain towards a higher level one in order to be compliant with the Bell-La Padula model. The reverse path is normally prohibited because could lead to an exfiltration of information. In the second mode data can move in both directions, but in the high-to-low direction further security mechanisms must be applied in order to sanitize the flow from data that can not be read by the lower level domain. It means that the cross-domain solution must have the capability to inspect data and filter them according to the established security policy. It would be also desirable that a CDS could dynamically change its configuration and its filtering decisions in order to meet the needs of fast-changing cooperative networks. In the panorama of COTS cross-domain solutions, few companies started offering products based specifically on the Separation Kernel concept and than we can assume adherent to the MILS paradigm.

Rockwell Collins developed a suite of software components called SecureOne Domain Technologies running on the Wind River VxWorks platform [107]. The main purpose is to equip aircrafts so that they can securely process, communicate and display hierarchical data using a single MILS architecture machine instead of physically separated devices. For example the SecureOne Guard component can be combined with a high-assurance programmable cryptographic product called Rockwell Collins KOV-7 providing a MILS distributed architecture capable of protecting Unclassified through Top Secret data simultaneously. Other solutions owned by Rockwell Collins, called Turnstile High Assurance Guard and MicroTurnstile Cross Domain Solution are built on the AAMP7 microprocessor [108], that provides a partitioning mechanism to enforce an explicit communication policy among applications.

Thales offers a system called Themis to enable access and communications on multiple domains that is based on the PolyXene solution. It is not simple to get detailed

information about these solutions, because vendors make only generic and high level descriptions available on their websites. To find more detailed information it is often necessary to infer data from the poor illustration given by datasheets, if they exist, and from information provided by academic and open-source communities.

Besides the awareness that there is a lack of MILS CDSs in the international marketplace, we decided to start developing the SoftGap components for further motivations. Our MILS Distributed Architecture is quite complex due to the presence of many specific-purpose components that use different technologies. The most interesting and new one was the Separation Kernel Hypervisor that represents the basic element upon which we built our security functionalities. It was important to verify the effectiveness of the offered security mechanisms and also to become acquainted with the SKH itself. Another reason was represented by the actual need of such solution in important contexts like, e.g., the NATO IEG.

3.3.1 SoftGap Architecture

In order to contribute in filling up the lack of MILS cross-domain solutions, we proposed in the framework of our distributed MILS architecture an innovative component called SoftGap [109, 110]. This component was designed from the scratch in the MILS and defense-in-depth paradigm for a specific security application. Our objective was to build a high-assurance system that allowed controlled unidirectional data flow from a low classification level domain (e.g. the Internet) to a high classification level one (e.g. a protected LAN). Actually we designed our solution in order to provide in the future a bidirectional data flow control, and also extend its capability within a single domain where further security functionalities are required to allow data exchange among actors with different clearance and need-to-know. The proposed MILS architecture is depicted in Fig. 3.12.

SoftGap enforces information flow control and separation of networks through the use of a Separation Kernel Hypervisor (SKH) and different Security Enforcing Functions (SEF). In the specific configuration depicted in Fig. 3.12 running on a Quad Core CPU, one core is assigned to each Subject (guest OS) whereas two CPU cores are assigned to the Separation Kernel Hypervisor to perform its specific SEFs. As already explained, a Separation Kernel Hypervisor is the virtualization platform that performs hardware resource protection from unauthorized access using different mechanisms. It gives to security engineers a refined control over CPU scheduling, memory and hardware resources to enforce the separation of processed information. There are two virtualized subjects, one connected to the low classification level network, in the following referred as Low-Side Subject (LSS), the other connected to the high classification level network, known

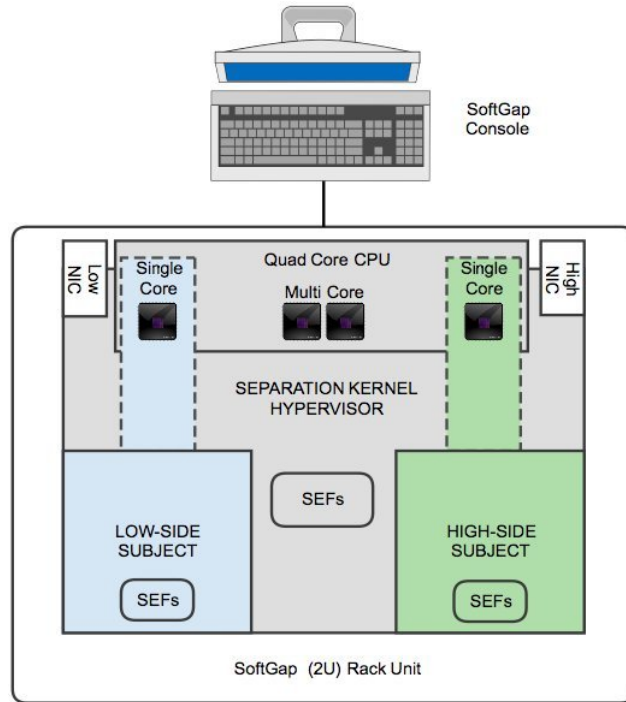


FIGURE 3.12: SoftGap Architecture

as High-Side Subject (HSS). Our SoftGap foresees two different network cards for that purpose.

3.3.2 System Model

According to the system, we make different assumptions for our solution. According to the system user, we assume that:

- administrators are non-hostile and appropriately trained;
- each user has public/private key pair (provided by smartcard/token);
- there is only one user at the same time on the machine;
- no more than one user can work concurrently on the machine.

According to the software/hardware environment, we assume that:

- we use a single computing machine;
- there are two different classification levels: high and low;
- the machine has two network cards;

- the network connection is made in an *exclusive-or* way: only one network card can be connected either to the low-side or the high-side at any time;
- the Separation Kernel Hypervisor (SKH) used to virtualize the computer environment is secure (it is designed to satisfy the Common Criteria Separation Kernel in High Robustness v1.03 functional and information assurance requirements [45] and it is based on a security enhanced version of Separation Kernel [48]);
- the Separation Kernel Hypervisor has public/private keypair;
- the algorithms for digest, signature, symmetric and asymmetric encryption are secure;
- there are two antivirus software regularly updated, that are different on the two sides;
- files downloaded from the Internet are saved on a specific location on the low-side partition (Low-Side Subject) and, they will be copied on a specific place on the high-side partition (High-Side Subject) by the SKH. Both locations are shared among users, because the SKH does not know the users of the Subjects.

3.3.3 Attack Model

The expected attacks can originate on the low-side network and the high-side network. Three attack scenarios are considered where attackers can control:

- only the Low-Side Subject (outside attacker);
- only the High-Side Subject (inside attacker);
- both subjects (a hybrid attacker).

The main threat is represented by the exfiltration of data from the high classification level network to the low classification level network that subvert the Bell-La Padula security model. Attackers could try to import malware/viruses inside the High-Side Subject. An attacker could also try to subvert the security policies, e.g. appending malicious code to a downloaded file on the Low-Side Subject. Such code could be later used in the High-Side Subject to open a backdoor towards the low classification level network. Another attack vector could make the Low-Side Subject or the High-Side Subject crash in order to steal information from resources not properly managed.

3.3.4 Security Enforcing Functions Details

We defined the following Security Enforcing Functions and their respective security mechanisms in the defense-in-depth paradigm:

SEF 1 checks the presence of virus/malware on the downloaded files

SEF 2 ensures the file archive integrity

SEF 3 ensures the source integrity

SEF 4 generates a symmetric encryption key

SEF 5 ensures the integrity of the symmetric encryption key originator

SEF 6 protects the confidentiality of the symmetric encryption key

SEF 7 declassifies the signed symmetric encryption key

SEF 8 imports the (signed and encrypted) symmetric encryption key

SEF 9 protects the confidentiality of the message containing the file archive

SEF 10 deletes in a secure way the data on the subject (key, files and/or archive)

We realized the security mechanisms implementing our SEFs both on the subjects and on the software environment directly provided by the Separation Kernel Hypervisor. A security function can be realized by one or more security mechanisms, e.g. the SEF 2 uses a mechanism that calculates the digest of the file archive on both Subjects and another mechanism that verifies the digest on the High-Side Subject. Likewise, SEF 6 and SEF 9 are composed by two mechanisms that perform the encryption and decryption operations, respectively.

3.3.5 Sequence of Operations

The sequence of operations is detailed in Table 3.3. The related UML sequence diagram is also depicted in Fig. 3.13. The sequence of operations on the HSS and the LSS are also depicted in Fig. 3.14 and Fig. 3.15. Here, we describe all the steps foreseen in the protocol. The SoftGap enforces the identification and authentication (I&A) policies on both subjects (low-side and high-side) by two-way I&A mechanisms (password and smartcard/token).

The user must log on the Low-Side Subject in order to use the system. Initially the Soft-Gap is an isolated environment with two disconnected network cards. After a successful log-in the SKH connects the Low-Side Subject network card to the low classification level network. The user can now download files from the low-side network through a strictly controlled environment that saves them in a secure folder. In this phase, an attacker could load malware in the Low-Side Subject but Low-Side Subject antivirus checks this likelihood. Even if the antivirus is not able to detect it, or if an attacker

introduces malicious software after this security check, the High-Side Subject (HSS) can later detect it through a different antivirus software. An archive is created for perfor-

TABLE 3.3: Sequence of operations

Op	From	Action	Msg	To
00	user	$\log_on(pwd_1)$	pwd_1	LSS
01	SKH	$connect_LSS_network_card$	-	LSS
02	user	$F := files_to_import$	-	LSS
03	antivirus 1	$check_files(F)$	-	-
04	LSS	$A := create_archive(F)$	-	-
05	SKH	$disconnect_LSS_net_card$	-	-
06	LSS	$D := calculate_digest(A)$	-	-
07	user	$S_D := encrypt(D, K_{Priusr})$	-	-
08	SKH	$K_{rnd} := generate_random_key$	-	-
09	SKH	$S_{Krnd} := encrypt(K_{rnd}, k_{Priskh})$	-	-
10	SKH	$E_{SKrnd} := encrypt(S_{Krnd}, K_{Pubusr})$	-	-
11	SKH	$downgrade(E_{SKrnd})$	-	-
12	SKH	$send(E_{SKrnd})$	-	LSS
13	LSS	$S_{Krnd} := decrypt(E_{SKrnd}, K_{Priusr})$	-	-
14	LSS	$K_{rnd} := decrypt(S_{Krnd}, K_{Pubskh})$	-	-
15	user	$\log_off()$	-	LSS
16	user	$switch()$	-	HSS
17	user	$\log_on(pwd_2)$	pwd_2	HSS
18	LSS	$M := encrypt(A S_D, K_{rnd})$	-	-
19	SKH	$secure_delete(F, A)$	-	LSS
20	SKH	$secure_delete(K_{rnd})$	-	LSS
21	LSS	$send(M)$	M	HSS
22	SKH	$secure_delete(K_{rnd})$	-	SKH
23	HSS	$S_{Krnd} := decrypt(S_{Krnd}, K_{Priusr})$	-	-
24	HSS	$K_{rnd} := decrypt(S_{Krnd}, K_{Pubskh})$	-	-
25	HSS	$A S_D := decrypt(M, K_{rnd})$	-	-
26	HSS	$D' := calculate_digest(A)$	-	-
27	HSS	$check\ D == D'$	-	-
28	HSS	$F := unzip_archive(A)$	-	-
29	antivirus 2	$check_files(F)$	-	HSS
30	HSS	$send(OK)$	OK	user
31	SKH	$connect_HSS_net_card$	-	HSS

mance reasons, in order to apply the cryptographic operations on a single file container.

The SKH disconnects the Low-Side Subject network card. The Low-Side Subject is now a closed environment on which operations can be securely performed.

The Low-Side Subject calculates then the hash value for the archive using e.g. SHA-3. Such value and its signature allow the High-Side Subject to detect any unauthorized change in the archive. The Separation Kernel Hypervisor generates a random key necessary to encrypt the message composed by the archive and its signed hash value. To allow the Low-Side Subject to read the symmetric key, such information must be declassified by the SKH that represents the originator of the key. In order to declassify the key, SKH signs the symmetric encryption key with its own private key, encrypts the signed key with the public key of the user provided by the token/smartcard and stores it in a strictly controlled memory location. Now the Low-Side Subject can read, verify the signature, and decrypt the symmetric encryption key. The user logs off the Low-Side Subject, switches to the High-Side Subject and logs on it. Specific SEFs on the LSS securely perform the encryption of the archive and the signed digest with a symmetric encryption algorithm (e.g. AES with Galois Mode or AES-XTS Mode).

The encryption is necessary to enforce the rule that, once the archive is encrypted and the original files/archive are securely deleted, the formerly Low-Side Subject data now become High-Side Subject data. Being High-Side Subject data now it is not allowed to be in cleartext on Low-Side Subject any more. On the Low-Side Subject the symmetric encryption key is securely deleted. The SKH sends now the signed and encrypted symmetric encryption key to the High-Side Subject and securely deletes it from the SKH. The High-Side Subject performs the cryptographic operations in the reverse order to obtain the symmetric encryption key, and then the archive with its original hash value. Furthermore, it calculates the digest of the received data and compares it with the original one. If the check is valid, the High-Side Subject antivirus performs a malware/viruses scan. The High-Side Subject antivirus is different from the Low-Side Subject one to increase the potential detection of viruses. The SKH connects the High-Side Subject network card to the high classification level network.

At this point, the downloaded files can be used on the High-Side Subject and/or disseminated on the high classification level network. In a different use case, instead of performing the decryption operations on the High-Side Subject, the symmetric encryption key and the imported message could be sent in consecutive phases to a host connected to the high classification level network. In this scenario if a system crash occurs, it is impossible to find cleartext on the SKH and the High-Side Subject because data are stored in encrypted form.

3.3.6 Design Approach Rationale

Being an applied research project, we thought that it could not be led according to the procedures used in the engineering of classic ICT systems. In particular, the waterfall design process is not the best applicable choice for this context, where the research character requires the formulation of hypotheses, the creation of a theoretical model in order to verify their correctness, and subsequently the field test of the effectiveness of the model in the real use case. In this perspective a classic approach is counter-productive because scarcely ever the initial developing stages of a project of this nature will remain the same over time. On the contrary, they will follow the natural evolution guided by experimentation.

For the aforementioned reasons we decided to adopt an iterative software development process through an agile approach. Such process is known as *Unified Process*. It is iterative, incremental (or evolutionary), risk-driven and architecture-centered. In the waterfall model, classic phases of definition of the requirements, architecture design, development and testing follow one another in a single cycle which should lead to a complete system deliverable in the final stage. Unified Process uses an incremental and iterative approach based on short, fixed-length time-boxed projects. Each project or cycle includes partial progresses of the architecture that represent production-grade subsets of the final system. Regardless of the nature of a research project, various studies evaluating the correspondence between software system development processes and quality of the final product show that the waterfall model leads to the creation of systems where 40% of the functionalities is never used, and where 19% is rarely used [111]. Also a value between 25% and 35% of the requirements that were considered initially frozen inexorably varies during the evolution of the project [112]. In the agile perspective the document production itself follows a different development process, driven by the inability to freeze *a priori* the set of requirements. The documentation, following a development corresponding to the design and implementation of the system gains an added value in the agile time-boxed iterations. It means that it can neither be produced nor even used to describe in advance, in a fully detailed way, what the product of the research project will be eventually able to realize.

3.3.7 Security Analysis

To verify the validity of our solution, we used two different tools specifically designed for the analysis of security protocols:

- *Automated Validation of Internet Security Protocols and Applications (AVISPA) tool;*

- *Scyther*.

We modeled our protocol according to the language used by these tools. Avispa tool [113] uses the High Level Protocol Specification Language (HLPSL) defined in [114], whereas Scyther [115] uses the a Security Protocol Definition Language (SPDL), a formal semantics of security protocol based on [116]. Considering the HLPSL language, the Low-Side subject is modeled as:

```

role subject_low (A,H,B: agent,
                  Kap: public_key,
                  Hash: hash_func,
                  RCV_HA, SND_AH: channel (dy))
  played_by A def=

    local
      State : nat,
      M: message,
      Hash: hash_func,
      Digest: hash(message),
      Network_card_ON : bool,
      K: symmetric_key

    const
      false : bool,
      h_a_key, h_a_m_digest: protocol_id

    init
      State := 0

    transition

      0. State = 0 /\ RCV_HA(start) =|>
      State' := 2 /\ K' := new()
      /\ SND_AH({A.K'}_inv(Kap))
      /\ M' := new()
      /\ Digest' := Hash(M')
      /\ SND_AH({A.M'.Digest'}_K')
      ...

```

Avispa and Scyther then analyzed our protocol giving evidences that the claimed security properties were satisfied. In the near future we will extend the formal proofs using

the Isabelle system with higher-order logic (Isabelle/HOL) as reported in [117].

A set of vulnerability and penetration tests have been also performed on the Softgap. Nessus [118] and tool from the Kali Linux Distribution [119] were used to detect the presence of known vulnerabilities or protocol design errors that could be exploited to subvert the security policies. The preliminary results showed that no disclosure of information occurred on the low classification level network even by using trojans/backdoors in the High-Side Subject.

3.3.8 Security Evaluation Considerations

High-assurance systems need to be evaluated by an accredited evaluation body in the framework of a certification scheme in order to assure the completeness and effectiveness of the security functionalities they claim to fulfill. The problem becomes more complex because ICT security tendency is moving from the single system perspective towards the system of systems one. The MILS paradigm tends to simplify the achievement of a security certification by treasuring the evaluation experiences of the previous classical multilevel solutions. The combined use of a “divide et impera” strategy and a compositional approach should make feasible to reach high levels of assurance and the related security certifications [40].

The Common Criteria standard introduced in version 3.1 a new assurance class and a package that address the requirements necessary to ensure that two or more previously certified components can be integrated in a secure manner [61]. The introduced Composition Class (ACO) and the Composed Assurance Package (CAP) allow to reuse the evidences and the results of previous certifications, instead of performing a new evaluation of the integrated system but is not trivial. Such approach raises new problems to be solved, for example how to combine previously certified products in a scientific-based framework, or how to determine the dependability attributes of the composite system starting from the components properties. These challenges constitute some topics of the MILS programs introduced in the next paragraph. Even if the possibility to achieve very high assurance certification levels is the milestone of the MILS effort and even if it meets the receptiveness of the Common Criteria standard, its realization has not been reached so far. The participants of the Common Criteria Recognition Agreement (CCRA) are required, “*inter alia*” to define a National Scheme and precise guidelines for the specific country they represent. Until now no national organization in charge of promoting the compositional approach in the Common Criteria Scheme made efforts in this direction, even if there is some evaluations have been already conducted according the new approach [120].

Also the CCRA participants mutually recognize only Common Criteria Certificates that refer to the following groups:

- “a collaborative Protection Profile (cPP), developed and maintained in accordance with Annex K, with assurance activities selected from Evaluation Assurance Levels up to and including level 4 and ALC_FLR, developed through an International Technical Community endorsed by the Management Committee; ” or
- “Evaluation Assurance Levels 1 through 2 and ALC_FLR” [121].

As compared with the previous version of CCRA that foresaw a mutual agreement on product certification until the EAL4, this choice seems to go in the MILS opposite direction. There is another mutual agreement among European countries (the SOG-IS European Mutual Recognition Agreement [122]) that still provides the higher levels of mutual recognition (until EAL7) for defined areas if schemes have been approved by the management committee.

We think that higher levels of evaluation can still be reached by a MILS solution through:

- a NEAT compliant minimized Separation Kernel Hypervisor (100-200 KB) providing data isolation, information flow control and damage limitations;
- simple and independent bare-metal Security Enforcing Functions running directly on the CPU core without the assistance of a guest operating system;
- use of Common Criteria compositional approach for interfaces between the SKH and the SEFs
- use of certified cryptographic and hashing algorithms;
- formalization of the security key-components [123];
- formalization of interactions between the security system and the attacker
- use of security properties proof verifiers.

Finally we think that the current approaches to security certification should be reexamined in order to meet the real needs and levels of assurance required to protect sensitive information. Apart from the certification goal, the usage of a MILS architecture fosters:

- the reduction of physical devices. It is no more necessary to have separate machines to guarantee the security and the separation of classified and unclassified domains (through a classic Air-Gap). That means a big saving in terms of Size, Weight and Power (SWaP) but also improves the effectiveness and the reduction of message delivery time.

- simplified information control and management among different domains,
- lower costs to develop high secure and reliable systems and quicker deployment,
- increasing of the overall system security through the stricter controls required by a high level of assurance certification,
- lower need to design again systems in case of new security requirements set up to counter new threats,
- integration of legacy applications in a secure environment with the assurance that the overall security features are satisfied.

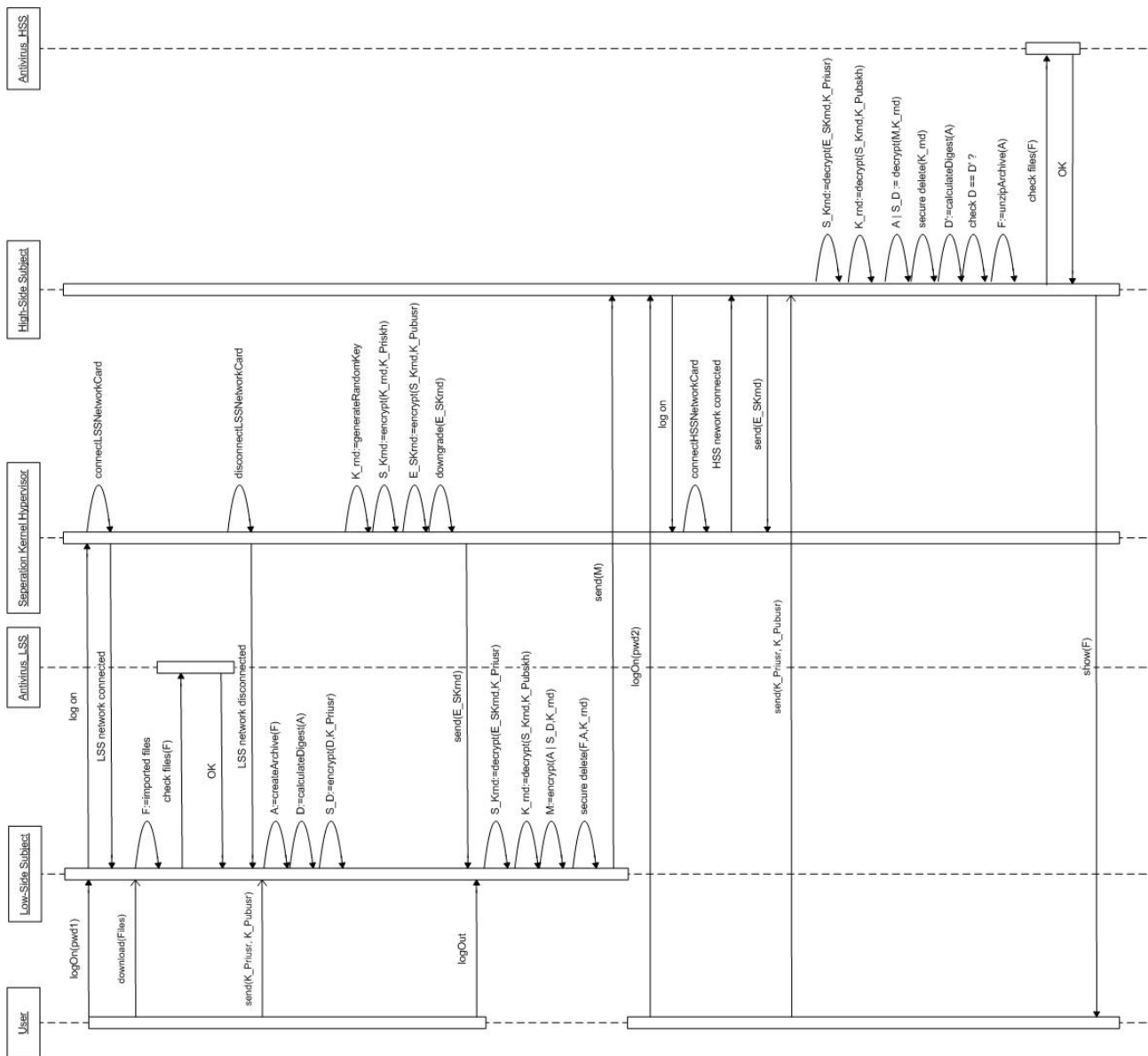


FIGURE 3.13: SoftGap Sequence of Operations

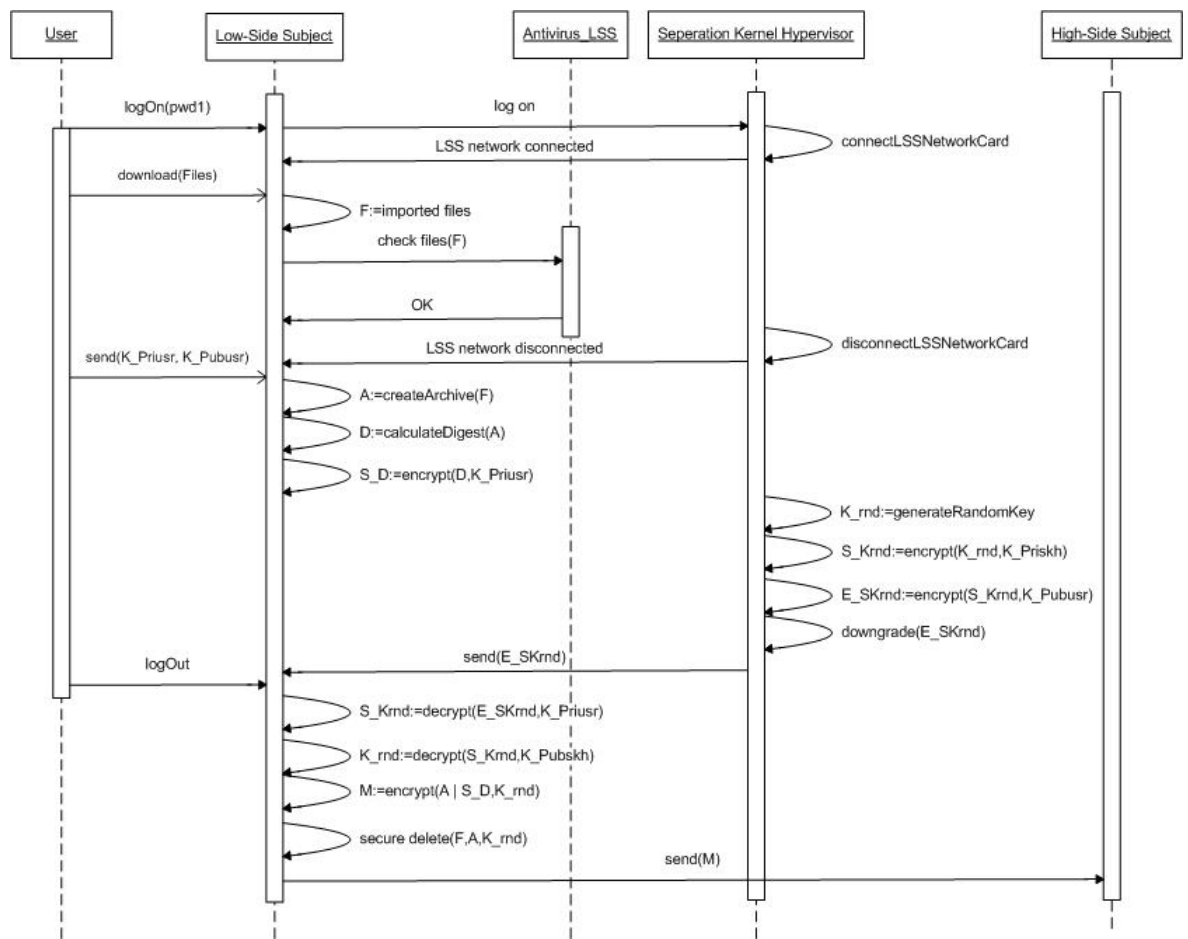


FIGURE 3.14: LSS sequence diagram

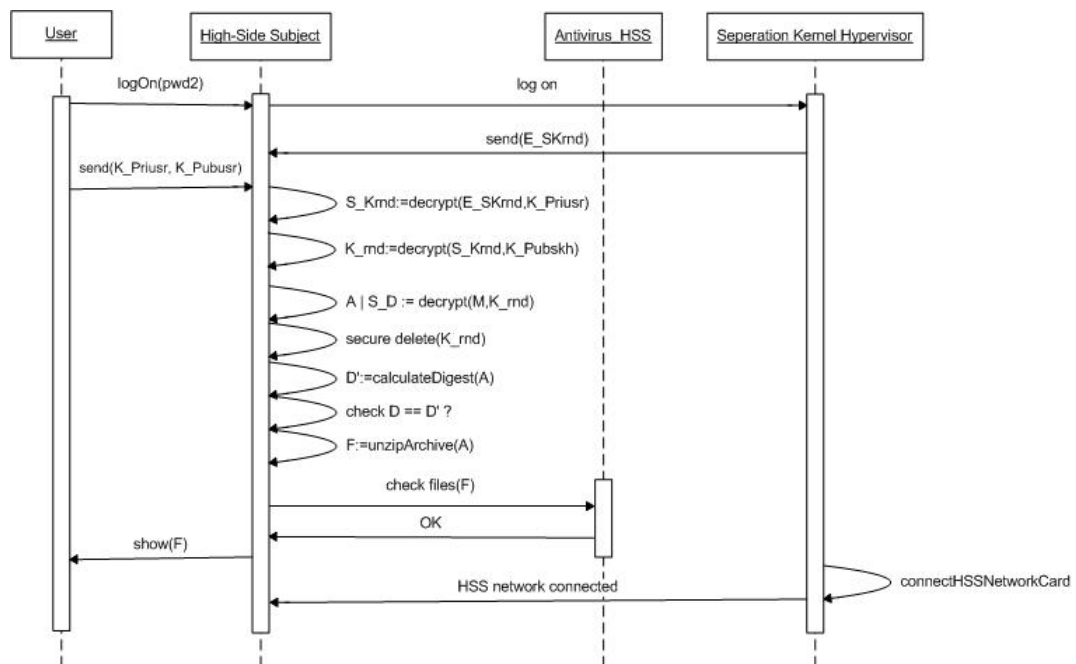


FIGURE 3.15: HSS sequence diagram

Chapter 4

COVERT CHANNEL DETECTION

IT IS COMMONLY KNOWN that even the best engineered software can contain vulnerabilities that can be exploited by attackers. In case of distributed multilevel systems is extremely important to enforce a controlled sharing of information across networks. Often such data sharing requires a guaranteed one-way flow of information. The unidirectional nature of these communication channels is challenging in the design of MLS systems. Most communication mechanisms, even those enforcing a unidirectional data flow, usually foresee an acknowledgment mechanism to communicate back to the sender. Without this back-channel most existing software and network protocols do not work unmodified. The overt channels are arbitrary communication paths that are intentionally enabled in a system or network. They can represent a risk for the controlled sharing of information in distributed environments. Even when overt information flows are controlled and properly managed in a communication channel, unintentional flows can still be there. These flows are called covert channels and are typically present when the sender can influence in some way the data provided to the receiver. Such operational ways can be summarized by the Prisoners' Problem.

4.1 The Prisoners' Problem

The prisoners' problem (not to be confused with the counterpart "dilemma" proposed by Albert Tucker as a problem of game theory in the '50s) was submitted by Simmons in 1983 and represents the *de facto* model for communications through covert channels [124]. The actors, Alice and Bob, are two prisoners intending to escape that need to

communicate with each other to reach their goal. The problem is the presence of a guard (Walter) who can read the exchanged messages and decide either to forward directly or to modify them (Fig. 4.1). For that reason Alice and Bob have to hide their actual communication in a manner that keeps Walter unaware about the very existence of the hidden message.

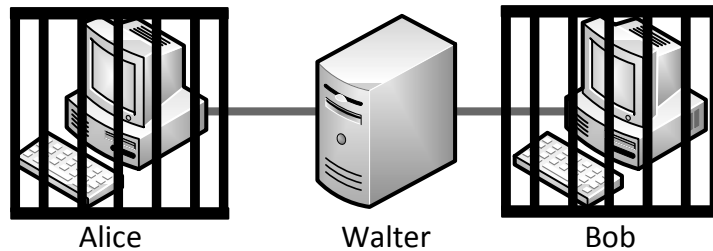


FIGURE 4.1: Prisoners' Problem: Alice and Bob represent hosts that exchange data through a channel that is hidden inside licit communications. Warden can read, drop and manipulate the licit communication.

By extending the concept to computer systems, Alice and Bob represent processes or hosts that exchange data through a channel that is hidden inside licit communications and therefore must share a secret to encode and decode the information. Walter instead is the one who manages and checks the system or the network trying to detect, control and eventually eliminate all forms of covert channels.

A scenario could be the following: Alice, a user with specific clearances logged on a host is exchanging information with Bob, another host characterized by the same classification level (Fig. 4.2).

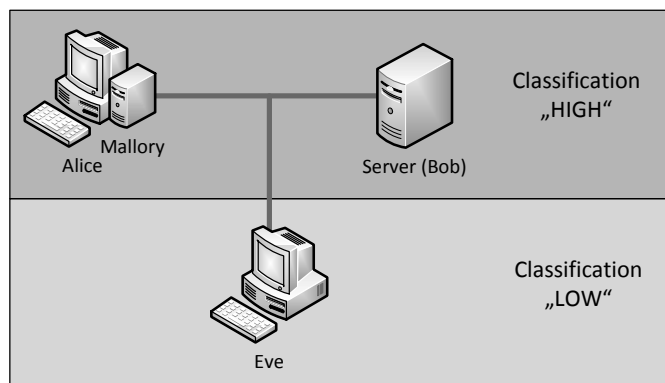


FIGURE 4.2: System Model: Alice and Bob could be for example a client requesting services and an internal Web Server replying to these requests.

The hosts could be for example a client requesting services and an internal web server replying to these requests. In such scenario a covert channel could be used by a third user, Mallory, to send arbitrary secret data to a lower clearance user (Eve in the picture), contravening surreptitiously the security policies. Through a malware that could operate as a covert channel proxy, Mallory could send secret data to Eve without any

modification of the original packets but just delaying them and, more important, without leaving any kind of evidence about the occurred information disclosure.

4.2 Covert Channel Definition and Taxonomy

According to the Common Criteria standard, a covert channel is an an enforced, illicit signaling channel that allows a user to surreptitiously contravene the multilevel separation policy and unobservability requirements of the system under evaluation. In presence of security objectives established to prevent users from observing the activity associated to other users, a covert channel analysis is foreseen to estimate its capacity and, in case, to reduce it. A covert channel, if present, could be used in high secure environments to disclose sensitive data towards unauthorized users even if such data flows are prohibited by security policies. The problem, known initially in the context of monolithic systems such as mainframes, has increased its popularity thanks to the spread of distributed systems and the growing number of protocols created in order to meet the requirements imposed by computer networks. The large amount of data has transformed the Internet in a high bandwidth medium for this type of channels that usually, to remain secret, can use only few bits within protocols where they are hidden. In addition, the increased attention payed by companies to protect their “open” channels (e. g. e-mail, USB memory sticks, etc.) made attackers more interested in this type of channels to pursue their fraudulent activities (industrial espionage, cyberattacks and so on).

Scientific literature offers varied records of covert channels, where authors focus on the correct nomenclature and precise definition of covert channels and propose detection methods they have test on their own covert channel implementations. In the following we try to clarify the main concepts and present a covert channel taxonomy that collects and reflects the most accepted view of the scientific world.

The term covert channel was introduced by Lampson [125] in 1973, and subsequently treated by several authors including [126] and [127] that over time identified it as:

- a transmission channel that has not been designed to directly transfer information,
- a channel in which the transmission of information takes place by storing the state of resources in variables,
- a channel originated from applying and administering the resource policies allocation,
- a channel using entities, that normally are not considered data objects, to transfer information between different subjects.

Usually, in the covert channel characterization, the basic distinction is between Covert Storage Channels (CSC) and Covert Timing Channels (CTC), although the definitions given often make them overlap and can lead readers to misunderstanding or confusion [128].

A covert channel is called “storage” if its scenario of use requires the presence of a subject that has the power to read/write memory locations or communication protocol header fields that may be accessed by another subject. A covert channel is called “timing” if a subject signals information by modulating the use of system/network resources and hence by manipulating the response time observed by a second subject.

A covert channel could also be active or passive. An **active** covert channel generates its own traffic in order to convey the hidden message whereas a **passive** one uses pre-existing communication traffic for its operations, improving in this way its *stealthiness* to the detriment of speed [129]. The former property, called also *covertiness*, pinpoints the ability of the covert channel to keep hidden the very existence of the communication and represents the most important feature an effective covert channel should have. The latter property is related to the covert channel capacity that, in turn, is bonded to the channel noisiness.

4.2.1 Covert Storage Channels

Most of covert channels fall into the *direct noiseless storage* category, it means that for example information flows directly between the involved parties without intermediate actors through an error-free path. Sender and receiver have access to the same portion of memory. These channels encode information in fields of protocols/operations that are not specified for example in RFCs or by exploiting semantic ambiguities. Hidden messages can be encoded in unused or reserved bits in frame headers of specific protocols. There is a great potential for such channels in all those protocols that do not dictate specific values for header fields, or in case intermediate devices or recipients do not check standard values. Authors of [130] for instance, proposed a covert channel that exploits unused IP header “Type Of Service” bit (TOS) or certain fields of TCP. In [131] is instead proposed to use the “Do not Fragment” IP header bit to convey information. Authors in [132] have proposed the use of covert channel inside VoIP streams. Data transmission is normally based on Real-time Transport Protocol (RTP) and control information is exchanged separately through the Real Time Control Protocol (RTCP). Instead of using different RTCP flows, Mazurczyk et al. [132] proposed to encapsulate the control information within RTP streams: the unused bits in IP, UDP and RTP headers identify the type of parameters whereas the corresponding values are inserted as watermark in voice data.

Normally protocols define header extensions to transport on request not mandatory information. In [133] potential covert channels are identified in IPv6 header extensions for routing, fragmentation and authentication.

Hidden information can be inserted as padding in frames or packets. Since the Ethernet frame must be at least 60 bytes long, if the protocol does not specify standard values for padding, and if a frame does not meet the minimum allowed length, it can be used to convey any type of data. Authors of [134] developed a method for exchanging covert messages through TCP “Timestamp” headers. Information is hidden in the timestamp least significant bit of the sender side, since it is assumed that they are random for slow TCP connections. Instead of modifying directly the timestamp, the algorithm slows the TCP stream so that the timestamp of the packet is still valid when they are sent. The algorithm checks if the least significant bit is equal to the covert bit to be sent and if the condition is true, the packet is immediately sent, otherwise it is slowed down with a predefined delay. Authors of two different works, [135] and [136], proposed to encode information either directly in the IP address fields or by modulating the order of valid addresses in subsequent transmissions. The capacity of this channel depends on the number of valid IP the sender can use but in general the technique works effectively even using the “Protocol” or “Port number” field. In [137] is proposed a framework, called Infranet, which uses an HTTP tunnel to provide a channel for secret communication between client and server. Upstream, Infranet clients send secret messages to servers by associating a meaning to the sequence of exchanged HTTP requests. Downstream, the Infranet servers reply hiding data with steganographic techniques within pictures.

Many other covert channels were discovered in the analysis of ad-hoc networks and in general in IEEE 802.2, 802.3, 802.4 and 802.5 communication standards [138]. Since wireless networks have variable error rates, such feature provides opportunities to insert corrupted frames. In [139] is showed a covert channel exploited by sending frames created intentionally with incorrect checksum. Authors of [140] proposed to send hidden information in IEEE 802.11 networks within not required ACK messages or invalid frames with deliberately incorrect checksums. The sender encodes data in the payload along with a magic number contained in the receiver address field. The receiver decodes data from the frame containing the magic number. In [141] is proposed a covert channel in IEEE 802.11 networks that uses corrupted frames: the sender encodes bits by duplicating some of the frames of specific connections and the recipient decrypts the covert bits according to such duplication.

Payload tunnels are covert channels that insert contents within the payload of another protocol in order to bypass firewall/routers policy. Many of these channels do not aspire to be undetectable, they rather aim at maximizing the transmission capacity. Nowadays there are a multitude of tools that use these mechanisms on protocols that are not normally blocked as ICMP and HTTP. It is also possible to use a fake DNS where

covert information is encoded in legitimate requests and the same type of covert data can be received by the following replies [142]. Some other covert channels are hidden in the pseudo-random data (even encrypted data). For example, the IP packet “Identification” (ID) header, used to reassemble fragmented IP packets, belongs to this group. The “Fragment Offset” (FO) header is used instead to determine the correct order of message packets. In [143] it is shown that, using a man-in-the-middle technique, it becomes feasible for an attacker placed in the middle of a transmission to convey information using the ID and the FO fields and setting a bit in the “Flag” one. The recipient can reconstruct the fragmented information distinguishing licit fragments having “More Fragment” bit set to zero, compared to those used for the covert communication.

There is a type of covert channels called *noisy* that exploits fields not accurately specified in protocol specifications or semantic ambiguities where these fields are changed on the path between sender and receiver. These changes can cause errors on the covert channel, and this is the reason why they can show some noise. Noise reduces their capacity, but potentially improves covertness. Compared to the noiseless channels there are few implementation of them and they are much less known. [144] proposed to use the IP Time To Live (TTL) header field to track flows without using the source addresses. The router changes the packet TTL field so that the downstream receivers can unambiguously identify their upstream router. Whereas the channel in [144] is used exclusively to mark packets, authors in [145] proposed a covert channel in the TTL field used for general communications. Since the TTL field, like the IPv6 Hop Limit Field counterpart, is modified by network nodes in the transition from the sender to the recipient, and because of packets can follow different paths, this channel is kind of noisy. A kind of covert *direct noisy* channel is also the one proposed in [146] that implemented a covert channel with high capacity in the physical layer protocols of wireless communications. The authors shown that using a constellation of symbols of the modulation system used by a Software Defined Radio realized with FPGA technology, it is possible for two devices to send hidden data that are confused as noise by legitimate devices.

Storage *indirect* channels belong to a different type that uses an intermediate unaware node to exchange illegal information. This type of channels shows a higher robustness to detection but it is more difficult to implement and has lower capacity. In [147] Rowland described an indirect channel, called “bounce” channel, which acts as follows: instead of sending a TCP SYN packet directly to the receiver with a particular Initial Sequence Number (ISN) and with hidden data, the sender sends the packet to a drop host with a spoofed source IP address set with the address of the desired destination. The drop host then sends a SYN/ACK or SYN/RST packet to the receiver with the sequence number equal to $(ISN + 1)$. The receiver decreases the ACK number to decode the hidden information. In [148] Zelenchuk implemented an indirect IP tunnel using the ICMP protocol. The sender forwards echo request packets to a drop host with spoofed source address

set to the recipient address and hides data encoding them in the payload. The drop host then sends the echo reply answers to the recipient with the same payload. In [149] Danezis proposed an indirect channel using the ID field of the IP protocol. This channel requires an unaware host equipped with an operating system that globally increases the counter ID field for outgoing packets. Furthermore, sender and recipient must be able to force the intermediary to receive packets and forward them back (for example, using the ping command). In each interval time, the sender forwards n packets to the intermediate node, in which n represents the length of the hidden information, forcing it to send back them again. The recipient can build the hidden information by calculating the difference between the values of two consecutive ID packets. In [150] Bauer proposed the use of web traffic channels to enable covert communications. Information is hidden inside JavaScript/HTML code and transported through the use of redirect JavaScript. An observer that is not able to examine the HTTP payload can not distinguish between web harmless users and malicious actors.

4.2.2 Covert Timing Channels

These type of channels are always *noisy* because of timing inaccuracies between sender/recipient and network jitter. Their capacity is typically lower than the noiseless storage one, but they are more difficult to detect. Channels disclose information by changing the rate of packets sent over the network. The recipient can reconstruct the covert message by measuring the rate in each interval time. In this type of channels the synchronization between subjects is therefore extremely important. In [151] Padlipsky et al. described a channel in which the sender encodes a bit of information simply transmitting or remaining silent in a predetermined time interval. This ON-OFF channel is a particular case of binary channel in which a given bit rate corresponds to the binary value 0 (OFF), while any other rate assumes the binary value 1 (ON). In [152] Cabuk et al. implemented an ON-OFF covert channel in which sender and receiver agree *a priori* on a time interval, using also the first packets bit to maintain synchronization. Later on the authors developed a more advanced CTC that uses pre-recorded sequence that are stored in two different partitions by a cut-off value. The sender replays using a randomly chosen value of one partition to transmit a 0 covert bit, or a value of the second partition to transmit a 1 covert bit.

In [153] Berk et al. described a timing covert channel that does not require synchronization because information is encoded within the interval between two consecutive packets. They were able to show that, by analyzing the values of channels characterized by only two gap values rather than multiple gap values, it is possible to choose the optimal symbol distribution once the peculiarities of the channel to be used are known. In [154]

Shah et al. developed a device that clips to the hardware connection between keyboard and computer and filters each keystroke modulating the time in which packets are sent over the wire (Jitterbug covert channel). In [155] Liu et al. presented a work in which the covert channel encodes data in a way that the inter-packet gap normal distribution is very close to the dispersal techniques used to increase the channel robustness. This feature makes the channel extremely difficult to detect with specific tests. Wolf [138] mentioned the possibility of building covert channels just modulating the use of the operations relating to a specific protocol. For example the recipient may acknowledge each frame separately or wait until the arrival of more frame before sending the ACK. Handel et al. [130] proposed a solution based on the modulation of Clear To Send (CTS) and Ready To Send (RTS) signals in serial communications. This technique can also be used in other protocols that use CTS/RTS, for example in WLAN. Eer et al. [156] implemented a timing channel through a web server and analyzed its capacity. In their scheme a web server carries hidden information delaying or not the answers, and associating to these events the logical values 1 and 0. Zou et al. [157] described a technique to insert a covert channel in the File Transfer Protocol (FTP). This solution transmits data by varying the number of commands “No Operation” (NOOP) sent during idle periods: the number of NOOP is equal the size of hidden information.

Servetto et al. [158] demonstrated that structured deletions of packets within a communication may be used as covert channel. The technique requires to number the packet sequence in a way that allows the recipient to detect the loss. Such deletions can be made artificially by the sender. Mazurczyk et al. [159] instead proposed a technique that uses packet losses and re-transmission. In their scheme the recipient does not send the ACK for a regularly received packet and waits until the sender re-transmits one in which, instead of the original data, covert information are sent using steganography. El-Atawy et al. [160] proposed a timing channel based on packet reordering in fake network traffic. The fake packets are used to encapsulate a sequence number within the IP packet payload.

Handel et al. [130] proposed to exploit the known mechanism of Ethernet Carrier Sense Multiple Access/Collision Detection (CSMA/CD). The covert sender begins to jam the communication of another user and waits for a time equal to zero or the maximum allowed value, so the packets sent by the user will be complete or segmented, thus providing the recipient a covert bit of information for each transmitted frame. The recipient can indeed reconstruct the hidden message detecting collisions and analyzing the order of arrival of the frame. Bhadra et al. [161] proposed a similar channel for ALOHA protocol while Dogu et al. [162] proposed a channel that uses splitting First Come First Serve (FCFS) algorithm used after the collision among user packets. Information is conveyed by observing the number of collisions detected in a predetermined time slot. The sender generates data just causing collisions whereas the recipient reconstructs them passively

controlling the channel and keeping track of the implemented collision resolution procedure. The timing *indirect* channels, like their storage channels counterpart, use an unaware intermediary to disclose data. This type of communication is very difficult to detect, and is also characterized by a relatively low capacity. Hintz [163] described a covert channel that uses a public server as an intermediary: the sender forwards a large number of requests or remains silent within a time slot, encoding information through this behavior (ON-OFF coding). The receiver sends periodic probes to the server by analyzing the response time of the sender.

Murdoch [164] instead developed a channel that combines the technique of packet rate and timestamp to modulate the covert channel. This technique requires an intermediary that receives and sends packets to the hidden communication subjects. The channel exploits the fact that the temperature of the CPU depends on the number of requests processed per time unit and the alteration of the host system clock depends on the temperature. The sender then forwards requests to the intermediary or remains silent, changing in this way the temperature of the clock and hence the system clock response. The recipient estimates the alteration of the clock according to the timestamp of the packets sent by the intermediary and decodes the hidden message. Gianvecchio et al. [165] proposed their Model-Based Covert Timing Channel (MBCTC) that selects a traffic model, like for example Exponential or Weibull, that fits with a legitimate traffic and chooses the one with the smallest Root Mean Square Error. The distribution of the generated pseudo-random inter-packet delays can then imitate that of the legitimate traffic.

4.3 Detection Algorithms

It is now clear that the employment of multilevel security countermeasures that prevent information flows from high-security domains to low-security domains are effective for traditional means like email, file transfer and so on. But all of them are still vulnerable and ineffective in presence of covert channels, that require specific countermeasures.

The problem is well-known by Certification schemes like the ISO/IEC 15408 standard or TCSEC, in fact they require evidences, through methodical analysis and testing, that the ICT solution under evaluation would not be affected by vulnerabilities that could be exploited by attackers to convey illicit data flows that contravene the security policies, i. e. through covert channels.

Covert channels opened new research directions focused on the identification, detection and mitigation of such vulnerabilities. Identification is performed in order to find the shared resources that can potentially lead to covert channel exploitation. It can be for

example done in the designing phase of an ICT solution. Detection is performed to detect covert channels that are already running and it usually falls in two main categories:

- misuse detection,
- anomaly detection.

Misuse of protocols and anomaly detection are normally performed by firewalls, intrusion detection and prevention systems. Both categories foresee a deep knowledge of protocols and system/network architectures, and the usage of different detection techniques.

4.3.1 Covert Storage Channel Detection Techniques

Misuse detection is particularly suited for CSCs that apply the mechanisms presented in the previous paragraphs. Some Covert Storage Channel can be directly eliminated through traffic normalizer devices. For example a firewall could zeroise option header fields of protocols that provide features not used in a specific environment (i.e. TOS field in IP headers). In cases where a CSC exploits header fields that are effectively used for the protocol operations, it is necessary to change strategy and focus on covert channel detection. For example ID field in IP packet is used to identify the packet and help re-assembling the original message. It could be manipulated in a way that uses the first byte as ID packet and the second byte as covert bits carrier. Observing the sequence of packets and the IP header fields can detect this kind of CSC. With the same approach it is possible to detect CSCs that use the flags Do-not Fragment (DF) and More Fragment (MF) to convey covert bits. If such mechanism is used then the presence of a covert communication can be suggested by the MF set to 1 even if the last fragment of the original message is less than the packet size and it should be set to zero [166]. Other more sophisticated techniques can be used. Joanna Rutkowska for example proposed an ISN generation model that predicts ISN values in TCP sessions based on previous ISNs generated by an operating system. Through the use of a neural network that is appropriately trained, it is possible to compare the predicted ISN value with the generated one and verify the difference. Large difference suggests that the value is not generated by the original stack and presumably a covert channel is set up [167]. Another technique takes advantage of Markov models for TCP protocol to check if the protocol rules are actually followed or not [168]. The paper shows that if the state transition of TCP protocol does not follow the rules, it could indicate the presence of a covert communication. The allowed TCP states are defined and they constitute the states of the Markov model. Depending on the specific application used (e.g. telnet, ftp, smtp etc.), different models are generated and used together with the Kullback-Leibler method

in order to check if the data sets present differences in relation to valid transitions. CSCs take place in IP and TCP protocol, but also in the application layer of the ISO/OSI stack. Application protocols offer many ways for covert channels that are difficult to detect and the detection methods are based on behavior analysis.

4.3.2 Covert Timing Channel Detection Techniques

Researches on Covert Timing Channels led to different solutions developed to detect, disrupt and also eliminate such channels. For example it is possible to add random delays on traffic or to buffer packets before sending them to the network. The problem with these type of solutions is that they affect also the licit traffic, reducing the overall performance of a system. The interest of Academia and Industry has then moved towards detection methods that evolved greatly in the last decade, like to underlying the general increasing of the perceived danger about these vulnerabilities. As already introduced, the popularity of covert channels increased together with the spread of distributed systems and the evolution of telecommunications networks that made such channels a medium with interesting bandwidth. Covert Timing Channels in particular represent vulnerabilities that are very difficult to detect, also respect to their storage homologous. This awareness lay the foundations for opening new research fronts that nowadays converge to use statistical tests that help in differentiating covert traffic from legitimate one.

Two are the main classes of detection tests:

- **shape tests**,
- **regularity tests**.

The shape of traffic refers to the shape of an histogram of a specific observed variable, e.g. the inter-packet delay¹ (IPD). It can be described for example by first-order statistics like mean, variance and distribution. Shape tests compare a distribution created from a sample to some known fingerprint of the legitimate overt distribution. Each shape test produces a metric that measures the difference between these two distributions. Generally speaking, a passive CTC is more prone to shape tests because it adds delay to legitimate traffic and then they change the shape of the original IPD distribution. A good quality active CTC can mimic the original stream distribution, hence the shape test is not suited for it. The effective test in this case is the regularity one, that is based on the observation of recurrences of specific patterns in the traffic distribution. This

¹Inter-packet delay, called also inter-arrival time, represents the time between transmission of consecutive packets.

type of tests assume that the covert data communication is more regular than the legitimate traffic. One example of such a test is the heuristic regularity test of [152], which examines whether the variance in the inter-arrival times remains small. Because CTCs are produced by computer programs, it is very difficult that they can produce the irregularity patterns present in legitimate traffic and the regularity test works starting by such assumption to perform their analysis. Mainly the regularity of traffic is described using second-order or higher-order statistics, like correlations in data sets.

According to the shape tests we find in the literature different solutions: the **Kolmogorov-Smirnov** test, the **entropy** test, the **chi-square** test, and the **ϵ -similarity** test, just to name but a few.

The Kolmogorov-Smirnov test (K-S test) described in [169] allows to measure the maximum distance between two empirical distribution functions:

$$K-S \text{ test} = \max |S_1(x) - S_2(x)|$$

where S_1 and S_2 represent the empirical distribution of the samples. Normally they are a training sample obtained by a legitimate traffic distribution and a test sample taken from the investigated distribution. S-K test is distribution independent, that means, it is applicable to any traffic distribution (e.g. Exponential, Weibull etc.). Once defined a threshold ζ , if the S-K test difference is less than ζ it means that the test sample belongs to a legitimate traffic distribution, otherwise it belongs to a different one, indicating the presence of a covert communication.

In [129] the authors present a detection method based on the Shannon entropy. The method organizes a sample of IPDs in bins with equal probability for legitimate traffic. Each bin should contain the same number of IPDs. If the calculated entropy of a sample of IPDs is less than the legitimate set, it is a sign that a covert communication occurred. Since in the building of the histogram of bins, most of them are empty, the authors improved the detection method using the Corrected Entropy (CE) that takes into account also such bins.

The chi-square test of [170] exploits IPDs to find out if they either conform to the Weibull distribution or not. The authors first estimate the Weibull parameters from the series X . Then, they apply the chi-square test to measure the degree of affinity. Finally, they verify the conformity of the sample data against a Weibull distribution fitted over the previously estimated parameters calculating the chi-square value according to the following formula:

$$chi-square = \sum_{i=1}^b \frac{(F_i^O - F_i^E)^2}{F_i^E}$$

where F_i^O and F_i^E are the observed and expected frequencies in the i -th bin, respectively, and b is the number of bins. If the result is less than a predefined threshold, then it shows the presence of a hidden communication.

In [152] two different methods are presented. The first one is a shape test, called ϵ -similarity, that is based on determining the proportion of similar inter-packet delays. Because an ON-OFF CTC creates groups of similar IPDs that are multiple of the chosen timing interval, many IPDs will be nearly the same and it indicates the presence of a covert communication. On the contrary, the second method belongs to the regularity tests group. It is based on the characteristic of most network traffic whose IPD variance changes over time whereas it is relatively constant in presence of covert channels. The sequence X of inter-arrival times is first separated in w different bins. Then, for each i -th bin, the standard deviation (STD) s_i is evaluated. Finally, the testing variable is the STD of the pairwise differences between each σ_i and σ_j for all the blocks $i < j$, and defined in the following formula:

$$regularity = STDEV\left(\frac{|\sigma_i - \sigma_j|}{\sigma_i}, i < j, \forall i, j\right)$$

where $|\cdot|$ is the absolute value (or modulus). The distribution with low variance probably hides a covert channel. The regularity measure is ineffective against covert timing channels where the model is changed more often than the size of the window or where the overt application is regular [171].

Among the aforementioned detection algorithms, Leo et al. proposed in their work [172] another method that targets a specific CTC called Cloak. This method, together with the ϵ -similarity test, are effective at detecting the specific CTCs they are designed for, and their scopes of detection are very limited.

Besides Cabuk's **heuristic regularity** test, another interesting test belongs to the same category: the **Corrected Conditional Entropy** test. CCE uses the Shannon entropy with corrected conditional entropy. The latter is able to detect CTCs with abnormal regularity, whereas the former can detect CTCs with abnormal shape. The combination of the two approaches can then be useful to detect a variety of covert channels whereas is not suited for CTCs with a distribution close to the legitimate traffic one.

Shape and similarity tests use the same procedure on different variables: first they calculate a metric, that is then compared with a proper preselected threshold. If the test metric is less than the specific threshold, this states that the test sample belongs to the set of legitimate traffic. Conversely, if the metric is above the threshold, the two samples belong to different distributions, thus suggesting the presence of a CTC in the test sample.

4.4 Novel Covert Timing Channel Detection Algorithm

In the panorama of CTC detection algorithms we devised a new testing procedure that measures the distance of a sample set of IPDs from the Weibull model. We called our procedure **Weibull-ness test**.

Our method discriminates between the presence or absence of a CTC, by measuring how much the series under investigation, i.e. the IPD, fits the Weibull distribution model.

4.4.1 System Model

Let X be the observed sequence of N IPDs (x_1, x_2, \dots, x_N) of a data communication. If the sequence X consists of samples of legitimate traffic, its probability density function (PDF) can be modeled as Weibull-distributed as shown in [129, 170, 171, 173]. A Weibull Distribution is defined as:

$$P_{X_i} = \begin{cases} \frac{k}{\lambda} \cdot \left(\frac{x}{\lambda}\right)^{k-1} \cdot e^{\left(-\frac{x}{\lambda}\right)^k}, & \forall x \geq 0 \\ 0, & \forall x < 0 \end{cases}$$

where $k > 0$ is the shape parameter and $\lambda > 0$ is the scale parameter of the distribution, respectively. Fig. 4.3² depicts the Weibull Probability Density Function with the change of shape and scale parameters.

Finding a distribution that fits the TCP flow inter-arrival times is an important issue in network-related applications and researchers tried to understand how much the IDP distribution in TCP communications is Weibull distributed. According to [174] and [175] TCP connection IPDs are statistically well modeled by distributions with heavy tails, such as the Weibull distribution.

According to some empirical studies, the paper [176] states that the TCP flow IPD distribution follows the Weibull distribution with a shape parameter close to 1.0 as the scale of network becomes larger, so the Weibull distribution degrades to the *Exponential distribution*. Moreover, the authors in [177] show that the marginal distribution of the inter-arrival times is piecewise Weibull distributed for wireless TCP/IP traffic.

We can state that the assumption of modeling the IPD distribution as Weibull is hence now widely accepted in the literature. Paper [171] chooses for example a Weibull distribution with a mean scale parameter $\lambda = 0.1279$ and a mean shape parameter $k = 0.4401$ to mimic legitimate traffic, whereas in [173], the authors select the same distribution with parameters $\lambda = 0.0020$ and $k = 0.4742$ for the legitimate traffic model.

In Appendix C details about the Weibull Probability Density Function are given.

²Di Calimo, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=9671814>.

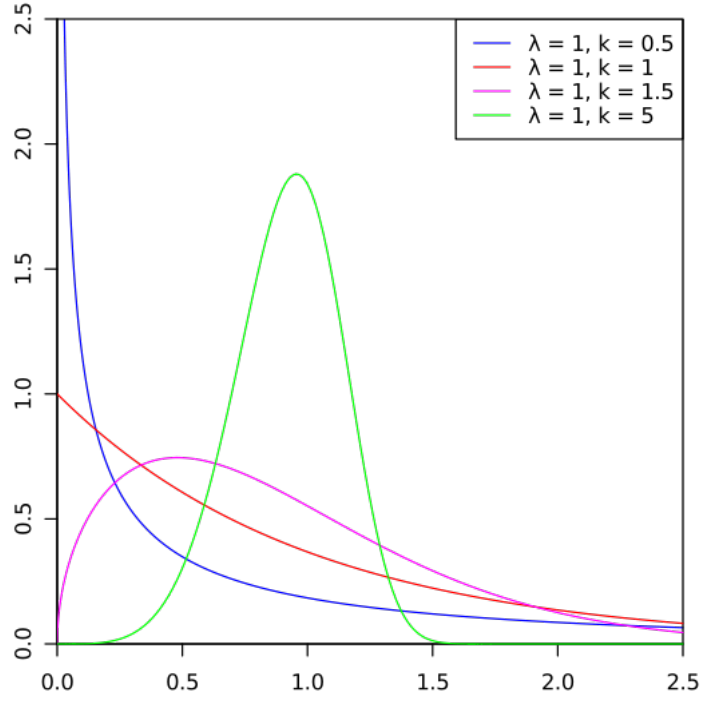


FIGURE 4.3: Weibull Probability Density Function

4.4.2 The Weibull-ness Test

Our detection method is able to detect the presence of a CTC by measuring how much the series under investigation (i.e. the packets inter-arrival times) fits the Weibull distribution model.

The two parameters of the Weibull distribution are first estimated, as done in [178], from the observed sequence of IPDs X . After that, a new sequence Y of N samples (y_1, y_2, \dots, y_N) is obtained from the sequence X , according to the following change of variable:

$$Y = \left(\frac{X}{\hat{\lambda}}\right)^{\hat{k}} \quad (4.1)$$

where $\hat{\lambda}$ and \hat{k} are the estimated scale and shape Weibull parameters, respectively.

As underlined in [179], the non-linear change of the variable expressed by 4.1 leads to a new sequence Y whose samples are distributed according to an exponential PDF $P_{Y_i}(y)$ of intensity 1 (for any $\hat{\lambda}$ and \hat{k}). This exponential PDF of intensity 1 is mathematically defined as follows:

$$P_{Y_i} = \begin{cases} e^{-y}, & \forall y \geq 0 \\ 0, & \forall y < 0 \end{cases} \quad (4.2)$$

The moments (of order n) of the PDF in 4.2 are all theoretically known according to the following:

$$M_n = E[Y^n] = n! \quad (4.3)$$

where M_n is the n -th order moment, while $E[\cdot]$ represents the expectation operator. Now, let us evaluate the testing variable needed by our Weibull-ness test.

If the sequence X represents the inter-arrival times of a legitimate communication, the sequence Y is exponentially distributed with known moments. In particular, since we estimated the two Weibull parameters exploiting the first two moments of X , we have now to focus on the first moment of Y different from the mean and variance. Hence, we decided to exploit the third order moment M_3 , whose value is theoretically evaluated as follows:

$$M_3 = E[Y^3] = 3! = 6 \quad (4.4)$$

Finally, we introduce a new variable Z defined as follows:

$$Z = M_3 - 6 \quad (4.5)$$

Now, its estimation \hat{Z} is used as the testing variable for discriminating about the presence and absence of the covert channel. Hence, the new testing variable is estimated according to the following:

$$\hat{Z} = \hat{M}_3 - 6 \quad (4.6)$$

where the estimate of the third order moment is obtained as:

$$\hat{M}_3 = \frac{1}{N} \sum_{i=1}^N y_i^3 \quad (4.7)$$

and y_i is the i -th sample of the transformed sequence Y . Then, considering a threshold h , the test is finally expressed as follows:

$$\begin{aligned} H_0 : \hat{Z} < \eta, & \text{ presence of CTC} \\ H_1 : \hat{Z} \geq \eta, & \text{ absence of CTC} \end{aligned} \quad (4.8)$$

This means that, if the testing variable is greater than the threshold value η , the algorithm decides for the hypothesis H_1 (i.e. presence of covert traffic). Otherwise, the choice is for H_0 (i.e. absence of covert traffic). Finally, we can use any higher order moment instead of the third one, in the evaluation of the Weibull-ness testing variable. However, exploiting higher order moments results in a greater variance of the testing variable (corresponding to very poor detection performance), as theoretically shown in the following section.

4.4.3 Performance Analysis and Numerical Results

In this section, we evaluate the performance of the proposed method, theoretically evaluating the bias and the variance of the testing variable in 4.6. In particular, for the bias we obtain:

$$bias(\hat{Z}) = E[\hat{Z}] = E[\hat{M}_3 - 6] = E\left[\frac{1}{N} \sum_{i=1}^N y_i^3 - 6\right] = 3! - 6 = 0 \quad (4.9)$$

Then, the variance is evaluated as follows:

$$\begin{aligned} var(\hat{Z}) &= E[\hat{Z}^2] = E[(\hat{M}_3 - 6)^2] = \\ &= E\left[\frac{1}{N} \sum_{i=1}^N y_i^3\right] \cdot E\left[\frac{1}{N} \sum_{i=1}^N y_i^3\right] - 12 \cdot E\left[\frac{1}{N} \sum_{i=1}^N y_i^3\right] + 36 = \\ &= \frac{1}{N^2} E\left[\sum_{i=1}^N \sum_{j=1}^N y_i^3 \cdot y_j^3\right] - 36 = \dots = \frac{1}{N} M_6 - \frac{1}{N} (M_3)^2 \end{aligned} \quad (4.10)$$

where M_6 is the sixth order moment expressed by:

$$M_6 = E[Y^6] = 6! \quad (4.11)$$

Finally, the variance of the testing variable is as follows:

$$var(\hat{Z}) = \frac{6! - 36}{N} = \frac{684}{N} \quad (4.12)$$

The variance of the testing variable expressed in 4.12 tends to zero, increasing the number N of observed samples. In addition, if in evaluating the testing variable, see 4.5, we exploit higher order moments (higher than M_3), this will result in a variance of the testing variable that is greater than the one expressed by 4.12. This finally results in a performance worsening of the method. Then, the CFAR procedure (usually used to solve detection issues in radar and telecommunications fields) is here employed to perform the test: first, a threshold is determined to limit the false-alarm probability (P_{FA}), at a given chosen value under the H_0 hypothesis (i.e. absence of the hidden communication). Then, the detection probability (P_D) is evaluated under the hypothesis H_1 (i.e. presence of the CTC) for the previously determined threshold. In addition, the testing variable in 4.6 is asymptotically ($N \rightarrow \infty$) Gaussian as a direct consequence of the central limit theorem. Hence, the test threshold can be asymptotically tuned from a straightforward evaluation of the Gaussian integral for a fixed P_{FA} , under the null-hypothesis [180]:

$$\eta = bias(\hat{Z}) + \frac{1}{\sqrt{2} \cdot var(\hat{Z})} \cdot erf c^{-1}(2 \cdot P_{FA}) \quad (4.13)$$

where $\text{erfc}^{-1}(\cdot)$ is the inverse of the complementary error function. Then, the P_D is determined in the H_1 hypothesis as:

$$P_D = \frac{1}{2} \text{erfc} \left[\frac{\eta - \text{bias}(\hat{Z})}{\sqrt{2 \text{var}(\hat{Z})}} \right] \quad (4.14)$$

with $\text{bias}(\hat{Z})$ and $\text{var}(\hat{Z})$ expressed by 4.9 and 4.12, respectively. To verify the detectability of CTCs in data communications we compared legitimate traffic to traffic containing a covert channel. To test the performance of all the considered detection methods, we exploit two types of simulated traffic:

- legitimate traffic, generated using the Weibull distribution model, and
- covert traffic, simulated by embedding covert communications in the legitimate traffic exploiting a passive CTC, called JitterBug [154].

JitterBug inserts an additional delay into the traffic generated by the transmitter. The delays imposed by the transmitter are limited to at most w milliseconds (ms). A covert bit 0 is encoded by increasing the IPD of the packet to a value modulo $\lceil w/2 \rceil$ ms, where $\lceil \cdot \rceil$ represents the ceiling operation. Conversely, a covert bit 1 is transmitted by increasing the inter-arrival time of the packet to a value modulo w ms. To avoid creating a pattern of inter-arrival times at multiples of w and $\lceil w/2 \rceil$, an additional random delay is considered.

In our results, we set the parameter $w = 20\text{ms}$, as determined in the hardware implementation of JitterBug in [154]. First, we evaluated the P_D of the innovative Weibull-ness test both in an analytic way, i.e. using equation 4.10, and by means of simulations. In Fig. 4.4 we report the theoretical and simulated results for the P_D of the Weibull-ness, versus the number of transmitted packets and varying the number of covert bits. In all the considered cases, the simulation results (dotted lines) well match the theoretical ones (solid lines), thus validating the correctness of the mathematical analysis performed. We also evaluated the detection performance of our method versus other conventional approaches. In particular, inter-arrival times from the legitimate and covert traffic were used as inputs to the three considered CTC detection methods: regularity, chi-square and Weibull-ness tests.

In Tab. 4.1, Tab. 4.2 and Tab. 4.3 we show the results for all the analyzed procedures when a covert message was inserted in flows consisting of 10000 packets. As in [178], we randomly injected in each of these flows an external trigger activating covert transmission consisting of 20 (small), 40 (medium), or 80 (large) packets in every interval over 250, 500, 1000 or 2000 packets. Differently from [178], we inserted the covert bits in non-consecutive packets in order to enhance the stealthiness of the CTC. The CFAR

procedure was exploited with a fixed $P_{FA} = 10^{-2}$, and the results are averaged over 10000 trials. In all cases, the Weibull-ness test provides the best detection performance, even improving the detection of about 24% versus the chi-square approach in the worst case of 20 covert bits over 250 packets, while the regularity test fails.

TABLE 4.1: P_D of the analyzed methods for a fixed $P_{FA} = 10^{-2}$ and small case

P_D small case (20 covert bit)			
n. of packets	weibull-ness test	chi-square test	regularity test
250	90.5%	73.1%	67%
500	35.8%	11.4%	32%
1000	4.8%	3%	11%
2000	3%	2.9%	1%

TABLE 4.2: P_D of the analyzed methods for a fixed $P_{FA} = 10^{-2}$ and medium case

P_D medium case (40 covert bit)			
n. of packets	weibull-ness test	chi-square test	regularity test
250	100%	99.5%	98.2%
500	90%	62.9%	5.5%
1000	40.5%	9.8%	3.8%
2000	8.2%	3.6%	2.4%

TABLE 4.3: P_D of the analyzed methods for a fixed $P_{FA} = 10^{-2}$ and large case

P_D large case (80 covert bit)			
n. of packets	weibull-ness test	chi-square test	regularity test
250	100%	99.4%	99%
500	100%	99.2%	98.2%
1000	97.3%	91.8%	96.7%
2000	57.5%	18.3%	3%

This behavior is a consequence of the fact that the JitterBug timing channel shifts the traffic IPDs distribution by increasing the delays to embed the covert message. The regularity tests will not detect statistical regularities in JitterBug traffic because the traffic is not produced by a statistical model [181]. Hence, in order to detect a shift of the distribution, the best detection method is represented by shape tests, and in particular, by the Weibull-ness test here devised.

Finally, in order to stress the detection performance of our test, we have analyzed what happens in the presence of a shortlived CTC, namely a *Needle* channel [182]. This is a more realistic scenario in which a small secret (e.g. a password or private key) is hidden in the data communication. To minimize the risk of detection, the attacker

toggles its use of the covert channel, transmitting a single bit once every M packets. In Fig. 4.4, we illustrate the detection performance of the Weibull-ness and chi-square tests, embedding the data every $M = 50$ and 100 packets. The results of the regularity test are not reported since this test again completely fails in detecting the covert traffic. This is because this type of CTC does not change high-level traffic statistics very much, making it very difficult to detect. However, Fig. 4.4 shows that the curves referring to the proposed approach are always higher than the ones of the conventional method, thus proving the efficiency and effectiveness of the Weibull-ness test in detecting CTCs. Our test is able to reach reliable detection (of about 90%) observing 400 packets, while the conventional solution needs at least 800 packets to declare correct detection of the covert traffic.

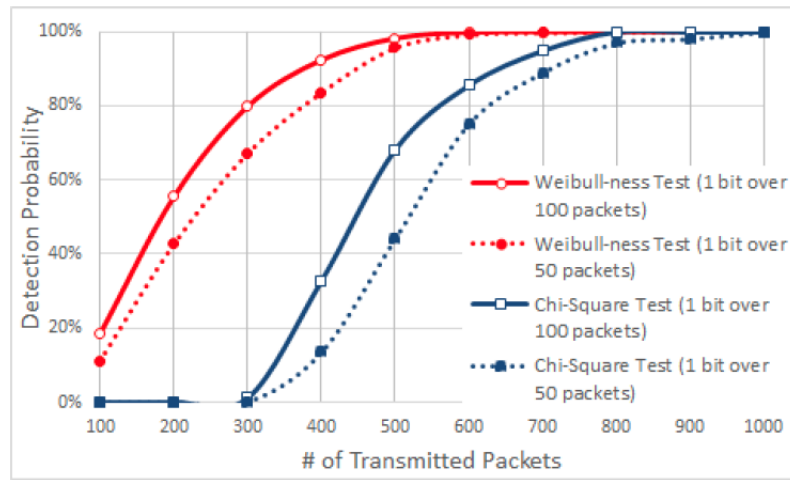


FIGURE 4.4: P_D of Weibull-ness and chi-square tests

4.5 Open Source Covert Timing Channel

A specific taxonomy of covert channels is present in many scientific papers and surveys, and many implementations are available on the Internet. But, according to the best of our experience, there is no public realization of a specific covert channel family, the Timing Covert Channel type. In our opinion a CTC is more difficult to detect than CSC and for this reason it could be used profitably as a starting reference model by manufacturers aiming at testing their security solutions against such vulnerabilities. In our case we needed an effective implementation to test our detection algorithm in a real scenario³. After a long research on the Internet, we found many implementations of Storage Covert Channels (an exhaustive list is present in [183]) hence, we decided to implement our own CTC. In the following section we will explain the features of our

³At the time this Thesis is being written, we started collecting real data with our covert channel in order to conduct preliminary tests on our detection algorithm.

channel and the implemented technological solutions through small code snippets. The source code is further detailed in the Appendix A and is also available at [184] under the license described in [185] (Apache Licence 2.0).

A covert channel is used to transfer information in a manner that conceals the very existence of the communication (unlike the encrypted channels whose purpose is to make the content of the transmission unintelligible). A CTC could create its own traffic, e.g. sending random messages and hiding a covert message by just manipulating their inter-arrival packet time. In case of a passive CTC, it uses some pre-existing traffic as a carrier of the covert message. In both cases, the packets payload is meaningless from the covert message receiver perspective, and the carrier itself could be even encrypted without compromising the effectiveness of the covert channel. Because of the asynchronous nature of this kind of communication, the main issue of CTCs is related to the synchronization between sender and receiver. In order to solve this non-trivial problem, we set up different mechanisms necessary to keep synchronization or, in the worst case, to re-establish it as soon as possible.

Our **Open-Source Covert Timing Channel (OSCTC)** implementation takes the cue from a paper written by S. Gianvecchio et al. [186], that introduces an entropy-based approach for detecting CTCs. Gianvecchio's paper introduces four different CTCs and we designed three different implementations of the first one, described formerly by Cabuk et al. [128], modifying the original one and adding further functional mechanisms.

In all our implementations the OSCTC uses a binary modulation of the inter-arrival packet delay in a TCP connection (ON-OFF coding). When the covert message source wants to send the bit 1 (ON), it transmits packets on the TCP connection, otherwise it remains silent signaling in this way the covert bit 0 (OFF).

We also developed two different operation modes: active and passive. In *active mode*, the OSCTC uses an own predefined message to forward the covert one. In contrast, in *passive mode*, it uses data exchanged in a different communication as carrier for the covert message. All the implementations resemble a client-server architecture. The covert bit period, called *TIME_SLOT* in our code, specifies the time within which a packet is transmitted or not. It's expressed in milliseconds (ms). Sender and receiver define the same function `calc_time` that is used to determine the local time on each machine and represents the starting synchronization point for both the hosts. On the receiver this function is called by two threads, one taking responsibility of reading the covert bits, that are organized in bytes, the other managing the *TIME_SLOT* sampling.

Function *calc_time*

```

1 double calc_time(){
2     double time;
3     char usec[7];
4     double microsec;
5     struct timeval tv;
6     time_t curtime;
7     gettimeofday(&tv, NULL);
8     curtime=tv.tv_sec;
9     sprintf(usec, "%i", tv.tv_usec );
10    microsec = atoi(usec);
11    time = curtime * 1000 + microsec * 0.001;
12    return time;
13 }

```

We implemented a first simple way to keep synchronization by the function `read_message`, that sets the packet received time at the center of the TIME_SLOT after each received packet, moving in that way the time window forward or backward with respect to the expected value.

Function *read_message*

```

1 void *read_message(void *socket){
2     char buffer[43];
3     do{
4         int bytesReceived=recv(*(int*)socket, buffer, sizeof(buffer),0);
5         if (bytesReceived > 0){
6             lasttime = calc_time();
7             start_time = lasttime - time_slot / 2;
8         }
9         else if (bytesReceived < 0){
10            perror("cannot read from socket");
11            break;
12        }
13        else{
14            running = false;
15        }
16    } while (running);
17    return NULL;
18 }

```

The three implementations of active OSCTC present different features. The first one, called Simple OSCTC, uses a predefined pattern to keep synchronization between sender and receiver. The second version implements a Manchester Code to add a further synchronization layer and the last one adds an Hamming Code for error correction

goals.

On the sender side we used the function `send_message` to send the covert message that is passed by a text file. With respect to the implementation explained in Cabuk et al. [128], we use the complete covert byte to carry information and we don't need the first Start Of Frame (SOF) bit to keep synchronization.

Our carrier is a TCP message saved in the variable `message`.

Function *send_message*

```

1 void send_message(char * covert_message, int client){
2     unsigned int coded_c = 0;
3     unsigned char c;
4     char * message;
5     usleep(time_slot*500);
6     message = "all work and no play makes jack a dull boy\0";
7     int length = strlen(message);
8     double start_time;
9     int waitsyncsymbol = 0;
10    for (int i =0; i < strlen(covert_message); i++){
11        c = covert_message[i];
12        coded_c = HammingTableEncode(c);
13
14        //send encoded message with Manchester code
15        for(int bitno = 0; bitno < CODE_BITS; bitno++){
16            start_time=calc_time();
17            if ((coded_c & 0x800) == 0x800){
18                send_xbytes(client, length, message);
19                do{
20                    ;
21                } while(calc_time()<=(start_time+time_slot));
22                start_time=calc_time();
23            }
24            else{
25                do{
26                    ;
27                } while(calc_time()<=(start_time+time_slot));
28                start_time=calc_time();
29                send_xbytes(client, length, message);
30            }
31            coded_c = coded_c << 1;
32            do{
33                ;
34            } while(calc_time()<(start_time+time_slot));
35        }
36        waitsyncsymbol=(waitsyncsymbol+1) % 3;
37        if(waitsyncsymbol == 0){
38

```

```

39      //each 3 bytes we send synchronization symbol
40      for(int bitno = 0; bitno < 4; bitno++){
41          start_time=calc_time();
42          do{
43              ;
44          } while(calc_time() < (start_time+time_slot));
45          send_xbytes(client, length, message);
46      }
47  }
48  }

```

This choice allows us to use the extended ASCII code and to perform an efficient error correction code in the third OSCTC implementation, as explained in the following sections.

4.5.1 Manchester Coding

In order to keep synchronization and, in case of loss, to restore it as soon as possible we implemented two mechanisms: synchronization pattern and Manchester coding. The OSCTC uses a synchronization pattern composed by 4 additional covert bits, repeated every 3 covert bytes. Such a solution allows the covert channel to track rapidly back the correct synchronization after loss. Manchester coding is a line code used in TLC that allows subjects involved in a communication to recover the synchronization from the encoded data. Each data element is encoded using at least one transition, e.g. bit 0 is expressed by a low-high transition and bit 1 by a high-low transition. That means that in our implementation each covert bit is represented by a couple 01 or 10 whereas the other two combinations (00 and 11) are not valid in our scheme. Such a mechanism allows the OSCTC to avoid synchronization losses due to long sequences of 0s and 1s that could be present in the covert message, even if it doubles the bits necessary to send the covert communication. Sender and receiver implement this code in the functions `sample` and `send_message`, respectively (see [184]).

4.5.2 Hamming Code (12,8)

Hamming code is an error correction technique invented by R. Hamming in 1950 [187]. An *Hamming Code* (m,n) uses n bits to check the m bits of information and consequently detect all double-bit errors and detect and correct correct all single-bit errors. In our OSCTC we implemented the Hamming Code (12, 8). Such a scheme encodes 8 covert bits in in 12 bits, adding 4 redundant bits. In this way, with a 50% of increase of the word length, we are able to detect 2 covert bit errors and correct 1 covert bit error.

Cabuk et al. [152] used an Hamming Code (7, 4) that represented a pondered choice for their covert byte that used 1 bit as Start Of Frame, 4 information bits and 3 correction bits. But it increases the 75% of the word length.

We pre-calculated the matrices necessary to the Hamming coding. Those matrices are quickly accessed by the functions `hamming_table_encode` and `hamming_table_decode` present on the sender and the receiver side, respectively.

Function *hamming_table_encode*

```
1 unsigned int hamming_table_encode(unsigned char data){
2     return hammingCodes[data];
3 }
```

Function *hamming_table_decode*

```
1 unsigned char hamming_table_decode(unsigned int code){
2     return hammingDecodeValues[code];
3 }
```

4.5.3 Development and Target environment

We provided two different ways of operation for our OSCTC: active and passive mode. The former uses a predefined message in a TCP connection to convey the illicit one, whereas the latter uses a picture that is downloaded by an unaware user that is browsing, for instance, an intranet portal. In the passive CTC, as a start, we decided to emulate a web server inside the OSCTC sender, that builds the HTTP message to be sent to the user browser. The source code for the passive OSCTC is also in Appendix A.

According to the passive mode, we used an image, *Lena.jpeg*, that provides the overt data used by our covert channel to convey the illicit message. In this case the system model could be the following: Alice downloads a picture (e.g. a Common Operational Picture or a Consistent Tactical Picture⁴ from a compromised web server (Bob) within a classified domain. Alice receives the image from Bob whereas Eve, a lower clearance user wiretaps the communication and reconstructs the covert message. It's important to underline that for passive mode, according to the scenarios depicted in the previous paragraphs, it's not necessary to have the receiver part of our implementation. Eve could simply record through a sniffer the communication between Alice and Bob, and re-build the covert messages decoding the information of the inter-packet delays.

In Fig. 4.5 is shown the picture used for the overt communication. In the presence

⁴Common Operational Pictures (COP) and Consistent Tactical Pictures (CTP) represent data sent to war-fighters and operators necessary to accomplish their missions.

of our OSCTC, the download is affected by a slowing down due to the inter-packet delays inserted by our covert channel. If the picture is part of a web page composed by different frames and containing images and writings, we can state that the user will difficulty become suspicious because of the delay of our image.

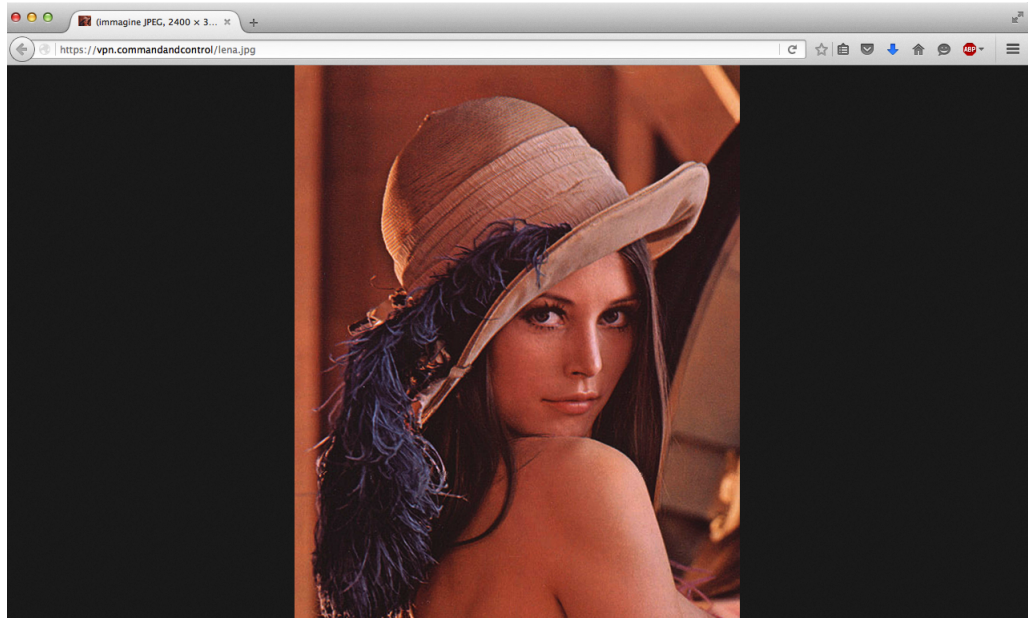


FIGURE 4.5: Web Browser downloading the picture used by our passive OSCTC

After deploying the OSCTC implementations, we tested their performances in terms of speed and correctness of the received message.

The OSCTC was implemented in C, using the Berkeley socket library offered by this language.

We tested our OSCTC in different scenarios. The first scenario is represented by sender and receiver sited on the internal network, that is a Virtual Private Network (VPN), of the Department of Applied Information Security (AIS) of Kassel University. The second scenario is a path between the same Department and the Department of Distributed Systems of University Duisburg-Essen. The third scenario is characterized by the path between AIS and the laboratory of Signal Processing for Telecommunications and Economics sited in the Engineering Department of the University Roma Tre.

4.5.3.1 Sender

The sender is a host sited in the network of University of Kassel (Germany), Department of Applied Information Security, equipped with *Debian 7.6* operating system in all the scenarios.

TABLE 4.4: Distance (hops) and RTT value (ms) from the sender

Receiver	Hop	RTT (min. - max.)
Kassel	1	0.005 - 0.009
Duisburg	11	8.8 - 11
Rome	18	39 - 70

4.5.3.2 Receivers

Receivers are three hosts sited in different networks and equipped with the following operating systems:

- Kassel - *Debian 6.0.10*
- Duisburg - *Debian 7.8*
- Rome - *Ubuntu 14.04.1*

4.5.3.3 Data test

We used a 795 characters covert channel message that we sent 100 times between sender (Kassel) and receiver (Kassel and Duisburg) and 30 times between sender (Kassel) and receiver (Rome) for each TIME_SLOT in a range between 1 and 40 milliseconds. This choice allowed us to perform the test during all the network conditions.

For example in the case *Kassel-Duisburg* with a TIME_SLOT range between 2 and 12 milliseconds, a single test run lasted more than 25 hours, whereas in the case *Kassel-Rome* with a TIME_SLOT range between 25 and 40 milliseconds, a single test run lasted more than 87 hours.

4.5.4 Empirical results

The distances between sender and receiver with their minimum and maximum average Round Trip Time (RTT) values are listed in the Table 4.4.

As shown in [164], the performances of a CTC vary according to the traffic network conditions and the length of the covert bit time interval (TIME_SLOT). According to the passive mode of operation, in which the user perception of the OSCCTC is related to the image visualization that is slowed down by the covert channel mechanism, another important factor is the buffer size used in the HTTP response.

Our OSCTC sends actually 76 bits each 3 covert message bytes, according to the following formula:

$$3 \cdot \{[8(bits) + 4(hamming)] \cdot 2(Manchester)\} + 4(syncro) = 76 \quad (4.15)$$

4.5.4.1 Network conditions

The figures below show the correctness of the received covert message using our three OSCTCs in all the different network scenarios. We show in Fig. 4.6, Fig. 4.7 and Fig. 4.8 respectively the three different OSCTC behaviors when sender and receiver are in:

1. Kassel network (Fig. 4.6)
2. Kassel - Duisburg (Fig. 4.7)
3. Kassel - Rome (Fig. 4.8)

The lines represent the following OSCTC implementations:

1. *Sync*, only the synchronization pattern is used
2. *Manch+Sync*, Manchester Code is added
3. *Manch+Sync+Hamm*, Hamming Code is added

We use the Levenshtein distance [188] as the metric for the correctness of the received message. The Levenshtein distance measures the difference between two different sequences. It is defined as the minimum number of edits necessary to transform one string into the other, with insertion, deletion, or substitution of a single character.

On the x-axis there are the TIME_SLOTS expressed in ms whereas on the y-axis there's the covert message error rate in terms of percentage. Results show that the OSCTC implementation with Hamming Code has the best performances in terms of errors both in the path *Kassel-Duisburg* and the path *Kassel-Rome*. Different behavior is shown in Fig. 4.6, we think because of the VPN that manipulates our TCP packets in a way that disturbs the covert channel.

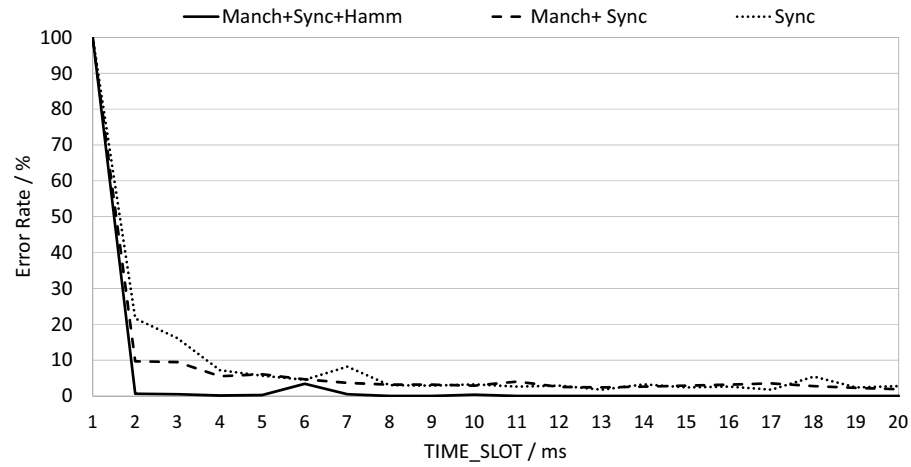


FIGURE 4.6: Covert message error rate percentage at different TIME_SLOT in Kassel network

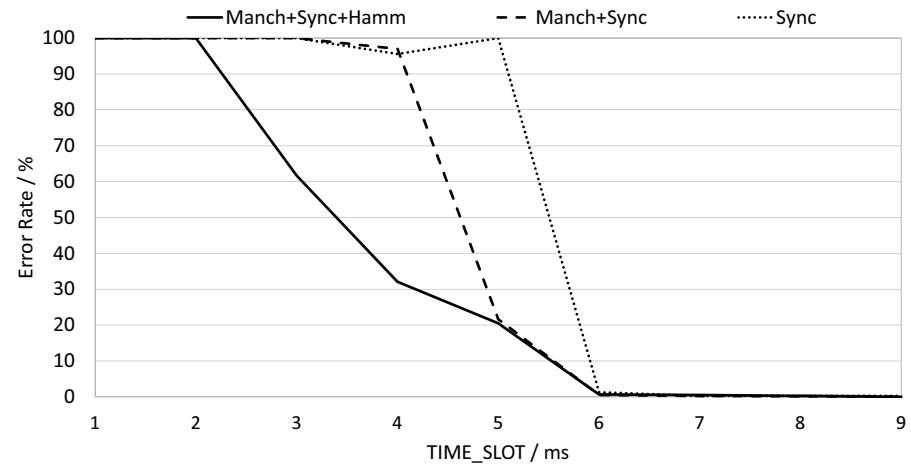


FIGURE 4.7: Covert message error rate percentage at different TIME_SLOT in path Kassel - Duisburg

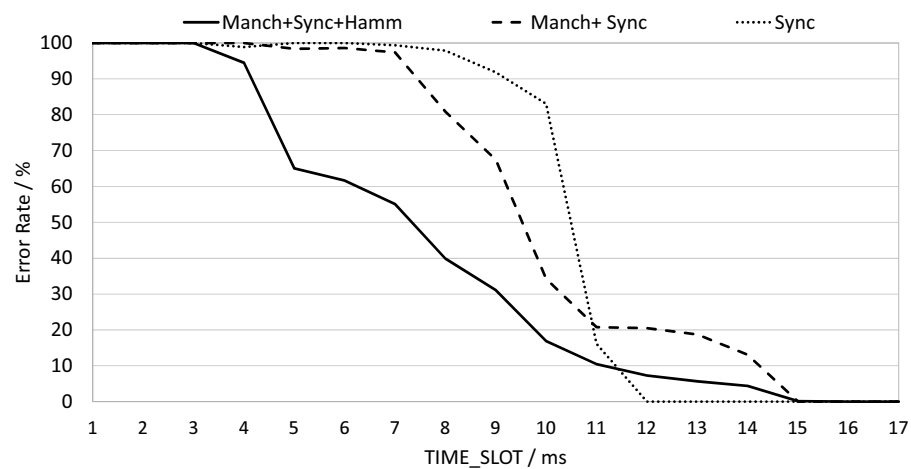


FIGURE 4.8: Covert message error rate percentage at different TIME_SLOT in path Kassel - Rome

4.5.4.2 Covert bit interval time

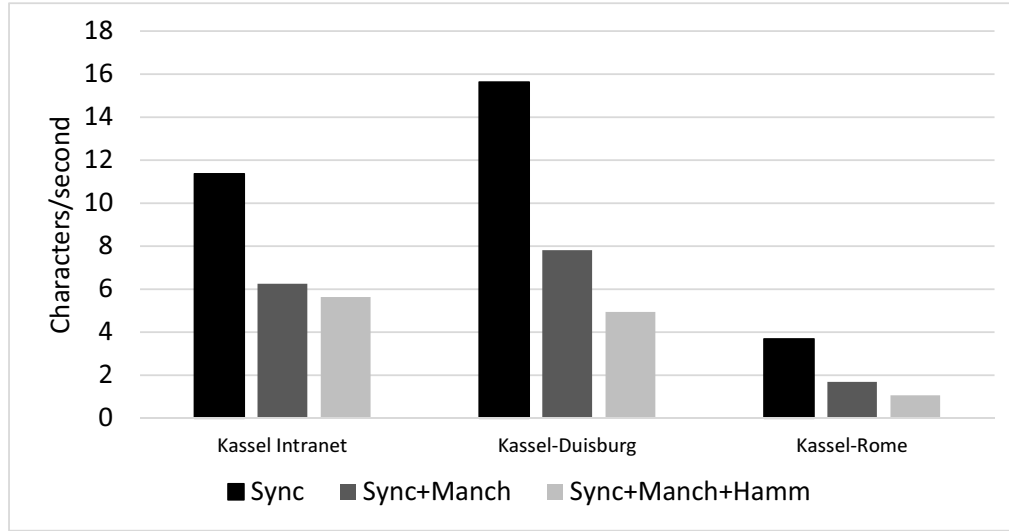


FIGURE 4.9: Characters-rate in different scenarios

Fig. 4.9 depicts the speed of our OSCTC implementations in all the scenarios at different TIME_SLOTS when the error rate is below the 3%. It is interesting to notice that, concerning only the data rate, the Simple OSCTC has the best performances. In fact, even if the implementation with Hamming Code reaches an error free message earlier than the others, the amount of covert bits necessary to encode data is much higher.

NOTE: At the time this Thesis is being written, we started collecting real data with our covert channel implementations in order to conduct further tests on the proposed detection algorithm.

Chapter 5

CONCLUSIONS

THE THESIS presented novel architectures specifically designed to face the multilevel security problem. The study was set out to explore new paradigms that overcame the limitations of the solution proposed nowadays, designed typically for the Defense sector. The research underlined the multilevel nature of many non-military communications that are used in different contexts requiring a strong protection of sensitive data. Why the multilevel security problem remained for a long time a topic circumscribed to the military community? Could it bring benefits to other sectors that usually deal with sensitive information on which different type of users, with different authorizations and need-to-know operate? Does multilevel security offer an adequate trade-off between the provided security and its costs in terms of investment, effort and performance? The study was set out to answer these and other important questions, that recent security-related events like Panama Papers, Vatileaks and Snowden's disclosures on the NSA global surveillance brought to prominence in the global panorama of data privacy and security.

5.1 Summary of Contributions

Even though the MLS problem represents a topic faced since years by the military community, it is not yet properly solved due to the multiple facets it requires to be taken into consideration.

The study has identified such aspects and analyzed the motivations that made the actual multilevel solutions unsatisfactory in meeting the multilevel requirements.

The study also sought to understand whether the inability of finding a multilevel “silver bullet” would owe to either technological or other kind of problem. Definitely the

combination of different factors limited the success of the Industry, Academia and Defense multilevel answer. The research tried to dissect each identified factor in order to understand the underlying problems and the cross-correlated implications.

The first important element that affects the MLS is the capability of a multilevel system to be certified according to worldwide recognized security certification schemes like the ISO/IEC 15408. A **new hybrid design methodology** has been proposed in this Thesis that bonds the systematic process of a security certification together with the lightweight and the capability of an agile development modeling to rapid delivery business value. The proposed methodology pushes, from the scratch, the certification process into the system design. Such approach from one side decreases the time and the effort of a security evaluation, from the other side it increases the certification success factor and eases a more correct and security-oriented design.

The way the MLS problem has been faced in the past is another element that had a big role in the partial failure of the multilevel solutions. The classic approach demonstrated that the multilevel problem should not be handled in a monolithic way, through components like Security Kernel and Trusted Computing Base that expanded incorporating even more system functionalities. The Thesis showed that a new approach, the Multiple Independent Levels of Security/Safety (MILS), is more suitable for such complex systems. In particular, the study drew attention to the synergy between the “divide et impera” approach of MILS paradigm, based on the separation of security mechanisms and issues into independent and manageable components, and the iterative nature of agile development. A **novel MILS Distributed Architecture proof-of-concept** has been proposed in this doctoral research, designed according to the aforementioned hybrid methodology in the defense-in-depth concept. The idea beyond this architecture has been the possibility to extend MILS features to contexts that require high-assurance systems, but do not have the economical resources of defense sector. The use case example chosen to illustrate the working principles of the MILS architecture has shown the potentiality and the feasibility of the proposed solution that hugely increases the security of exchanged sensitive data.

Technology is another factor that affected the effectiveness of standard multilevel solutions. Even though the MILS paradigm has been proposed years ago for the avionic sector, it was long limited to the hardware architectures designed specifically for such environment. The technological improvement of hardware components, in particular CPUs, made possible the implementation of MILS features that were only theorized short time ago. The **realization of a prototype of a specific architectural component, the SoftGap**, has demonstrated in this study the maturity and the effectiveness of the MILS paradigm in real scenarios.

The certification-related aspect and the multilevel nature of the problem analyzed in the Thesis do not allow to disregard an important issue of multilevel solutions: covert channels. The study deeply analyzed such class of vulnerabilities, and categorized the adequate countermeasures for different types of covert channel. A **novel higher order moments based test, the Weibull-ness test**, has been developed to detect a specific class of channels called Covert Timing Channels. The performance of the detection algorithm has been evaluated theoretically versus the conventional solutions. In order to test the algorithm with real data, a set of three Covert Timing Channels has been developed and released under the Apache License 2.0. It is freely available to researchers and developers that want to study the covert channel problem and test their security solutions against such vulnerabilities.

5.2 Lessons Learned

The study highlighted the main facets characterizing the multilevel problem, and the main empirical findings have been summarized in the previous paragraph. Many teachings have been assimilated.

According to the certification factor, it is necessary that the choice of the desired level of assurance must be performed after a risk analysis phase rather than the pure consideration of the classification level of data to be protected. A good risk analysis takes into consideration the assets that must be protected together with the specific environment and specific attacker types. Lessons learned: a top secret information should not be necessarily protected by systems evaluated at EAL6 or EAL7 of the Common Criteria scheme. When the asset is used in specific contexts, for example military enclaves with very restricted access control, a lower evaluation assurance level could be enough.

The proof-of-concept and the prototype realized in this study showed that the current lack of MILS solutions, particularly outside the military sector, is not a technological problem but mainly depends on the certification scheme and its application. The Common Criteria Recognition Arrangement members still do not fully share scheme interpretations and do not incorporate scheme modifications necessary to boost the evaluation of MILS solutions.

The MILS paradigm, as demonstrated by the theoretical evaluation of the performance of the MILS distributed architecture, showed that it is applicable in general purpose networks at the cost of increasing the overall network latency. Nevertheless, such latency can still be considered acceptable for the user in relation to the hugely increased level of security provided.

Another important teaching is related to the security protocols and algorithms used to protect sensitive data in daily life applications. A plethora of solutions exists and their effectiveness depends on a delicate balance among key size, performance, and modes of operation. It is not possible to become a security engineer overnight. Standards and recommendations like those provided by organizations as NIST, ANSSI, etc. are good reference points. But experience, good practice, and deep knowledge of security protocols, algorithms, and patterns should go along with a continuous study and attention to new attacks and vulnerabilities.

5.3 Open Challenges

The study has offered an evaluative perspective on an important topic that affects much more than military security. Since most programs concerning the multilevel security problem belong to the Defense world, it has been very difficult to enter in such closed subject. The discussion with some Italian Common Criteria Evaluation Centers and the German Federal Office for Information Security (BSI - Bundesamt für Sicherheit in der Informationstechnik) helped in such direction, but only partially.

As a direct consequence of this restricted access to military information, the study encountered a number of limitations, which need to be considered. The research is based on assumptions that arose in the military context. Such assumptions, provided by the contact with the aforementioned entities, could not be supported by evidences in a real military scenario.

Another limitation due mainly to the intrinsic complexity of the proposed MILS Distributed Architecture and to time constraints refers to the evaluation of the solution that has been performed in a theoretical way and could be supported also by difficult, but realizable, simulations. In order to fill such a gap and demonstrate the effectiveness of the MILS and the defense-in-depth paradigms, the study presented a working prototype of the SoftGap component that includes all the main ideas and technologies foreseen for the distributed architecture.

The study lays the foundations for the extendability and improvements. It would be interesting to analyze the possibility to extend the MILS architecture to mobile devices like smartphones. Mobile technology become always more invasive in the daily life operations and its use for applications that process and exchange sensitive data is increased over the time. The Separation Kernel Hypervisor chosen for the SoftGap prototype has already the ability to run fully virtualized Android requiring no changes to the guest

OS. The problem is the underlying hardware that has to meet very strict features in order to run the SKH and it is yet not fulfilled by any commercial device.

The presented MILS Distributed Architecture foresees the presence of two subnetworks, one working at system-high security mode of operations and one enabling effectively a multilevel communication. The extension of this capability also to the former subnetwork would be desirable to simplify the management of the domain security policy. For such goal the application servers, like for example the file servers, should enforce robust partitioning and resources separation through the use of an SKH.

Another important element for future research could be the use of the multi-homing features provided by a relatively new transport protocol called Stream Control Transmission Protocol (SCTP). Multi-homing SCTP is the ability for a single SCTP endpoint to support multiple IP addresses. It could improve the survivability of sessions in the presence of network failures and provide redundant paths to increase network resilience and reliability.

5.4 List of Publications

5.4.1 International Journals

- A. Tedeschi, A. Liguori, and F. Benedetto. Information Security and Threats in Mobile Appliances. *Recent Patents on Computer Science*, 7(1):3–11, June 2014
- A. Liguori. From Multilevel Security to MILS: the Evolution illustrated through a Novel Cross-Domain Architecture. *International Journal of Mobile Network Design and Innovation*, FORTHCOMING

5.4.2 International Conferences

- A. Liguori. A novel Multiple Independent Levels of Security/Safety Cross Domain Solution. In IEEE, editor, *Military Communications Conference (MILCOM), 2015 IEEE Conference on*, pages 1578–1583, October 2015. doi: 10.1109/MILCOM.2015.7357670
- F. Benedetto, G. Giunta, A. Liguori, and A. Wacker. A novel method for securing critical infrastructures by detecting hidden flows of data. In IEEE, editor, *Communications and Network Security (CNS), 2015 IEEE Conference on*, pages 648–654, September 2015. doi: 10.1109/CNS.2015.7346881

- A. Liguori, F. Benedetto, G. Giunta, N. Kopal, and A. Wacker. Analysis and monitoring of hidden TCP traffic based on an open-source covert timing channel. In IEEE, editor, *Communications and Network Security (CNS), 2015 IEEE Conference on*, pages 667–674, September 2015. doi: 10.1109/CNS.2015.7346885
- A. Liguori, F. Benedetto, G. Giunta, N. Kopal, and A. Wacker. SoftGap: A Multi Independent Levels of Security Cross-Domain Solution. In IEEE, editor, *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*, pages 754–759, August 2015. doi: 10.1109/FiCloud.2015.84

5.4.3 Books

- A. Simonetta, M. C. Paoletti, and A. Liguori. *Testing di oggetti matematici in java. Introduzione a JUnit*. UNIVERSITALIA, 1 edition, 9 2013. ISBN 978-8865075531

Appendix A

Covert Timing Channel Source Code

IN THIS APPENDIX the C source code of our OSCTCs for active and passive mode is presented.

A.1 Simple OSCTC

The Simple OSCTC is an active Covert Timing Channel that sends a covert message modulating the inter-packet delay of a TCP communication. It uses a predefined pattern to keep synchronization between sender and receiver. The covert message is ASCII-7 bit encoded.

A.1.1 Simple OSCTC Client

```
1 #include <sys/types.h>
2 #include <sys/socket.h>
3 #include <netinet/in.h>
4 #include <netinet/tcp.h>
5 #include <arpa/inet.h>
6 #include <netdb.h>
7 #include <stdio.h>
8 #include <unistd.h> /* close */
9 #include <string.h>
10 #include <stdlib.h>
11 #include <time.h>
12 #include <math.h>
```

```
13
14 #define SERVER_PORT 20001
15 #define MAXMSG 256
16 #define TIME_SLOT 20
17
18 #ifndef NULL
19 #define NULL ((void *) 0)
20 #endif
21
22 typedef int bool;
23 #define true 1
24 #define false 0
25
26 FILE *input;
27 int time_slot = TIME_SLOT;
28
29 //send overt message
30 void send_xbytes(int socket, unsigned int x, void * buffer)
31 {
32     int nSent = 0, nIndex = 0;
33     int nLeft = x;
34     while (nLeft > 0){
35         nSent = send(socket, buffer, nLeft, 0);
36         if (nSent < 0){
37             perror("cannot send message");
38             close(socket);
39             exit(-1);
40         }
41         nLeft -= nSent;
42         nIndex += nSent;
43     }
44 }
45
46 double calc_time(){
47     double time;
48     char usec[7];
49     double microsec;
50     struct timeval tv;
51     time_t curtime;
52     gettimeofday(&tv, NULL);
53     curtime=tv.tv_sec;
54     sprintf(usec, "%i", tv.tv_usec);
55     microsec = atoi(usec);
56     time = curtime * 1000 + microsec*0.001;
57     return time;
58 }
59 }
60
```

```

61 //send covert message in ON-OFF mode
62 void send_message(char * covert_message, int client){
63     char c;
64     char * message;
65     usleep(time_slot*500);
66     message = "all work and no play makes jack a dull boy\0";
67     int length = strlen(message);
68     double start_time;
69     int waitsyncsymbol = 0;
70     for (int i = 0; i < strlen(covert_message); i++)
71     {
72         c = covert_message[i];
73         c = c << 1;
74         for(int bitno = 0; bitno < 7; bitno++)
75         {
76             start_time=calc_time();
77             if ((c & 0x80) == 0x80){
78                 //printf("10, %f\n", calc_time());
79                 //printf("1\n");
80                 send_xbytes(client, length, message);
81             }
82             c = c << 1;
83             do{
84                 ;
85             }while(calc_time()<(start_time+time_slot));
86         }// end for(int bitno = 0;bitno < 7;bitno++)
87         start_time=calc_time();
88         do{
89             ;
90         }while(calc_time()<(start_time+time_slot));
91         waitsyncsymbol=(waitsyncsymbol+1) % 3;
92         if(waitsyncsymbol == 0){
93             //after sending 3 complete bytes we send our sync symbol
94             for(int bitno = 0;bitno < 8;bitno++)
95             {
96                 start_time=calc_time();
97                 send_xbytes(client, length, message);
98                 do{
99                     ;
100                 }while(calc_time()<(start_time+time_slot));
101             }
102         }// end for(int bitno = 0;bitno < 8;bitno++)
103     }
104 }// end for (int i =0; i < strlen(covert_message); i++)
105 }
106
107 /* main() */
108 int main (int argc, char *argv[]) {

```

```

109
110     int client; /* client socket */
111     int rc;
112     struct sockaddr_in local_addr, serv_addr;
113     struct hostent *host;
114
115     if(argc < 3 || argc > 4) {
116         printf("usage: %s <server> | [<\"> input_file_name.txt <\">] | [
time slot]\n",argv[0]);
117         exit(-1);
118     }
119     if (argv[3]){
120         time_slot = atoi(argv[3]);
121         printf("time_slot: %i\n\n", time_slot);
122     }
123     /* get host address from specified server name */
124     host = gethostbyname(argv[1]);
125     if (host == NULL)
126     {
127         printf("%s: unknown host '%s'\n", argv[0], argv[1]);
128         exit(-1);
129     }
130     /* now fill in sockaddr_in for remote address */
131     serv_addr.sin_family = host->h_addrtype;
132     /* get first address in host, copy to serv_addr */
133     memcpy((char *) &serv_addr.sin_addr.s_addr, host->h_addr_list[0], host->
h_length);
134     serv_addr.sin_port = htons(SERVER_PORT);
135     memset(serv_addr.sin_zero, 0, 8);
136     /* create local stream socket */
137     client = socket(AF_INET, SOCK_STREAM, 0);
138     if (client < 0) {
139         perror("cannot open socket");
140         exit(-1);
141     }
142     /* bind local socket to any port number */
143     local_addr.sin_family = AF_INET;
144     local_addr.sin_addr.s_addr = htonl(INADDR_ANY);
145     local_addr.sin_port = htons(0);
146     memset(local_addr.sin_zero, 0, 8);
147     rc = bind(client, (struct sockaddr *) &local_addr, sizeof(local_addr));
148     int on = 1;
149     setsockopt(rc, IPPROTO_TCP, TCP_NODELAY, &on, sizeof(on));
150     if (rc < 0)
151     {
152         printf("%s: cannot bind port TCP %u\n",argv[0],SERVER_PORT);
153         perror("error");
154         exit(-1);

```

```

155     }
156     /* connect to server */
157     rc = connect(client, (struct sockaddr *) &serv_addr, sizeof(serv_addr));
158     if (rc < 0)
159     {
160         perror("cannot connect to server");
161         exit(-1);
162     }
163     long f_size;
164     char* covert_message, *code;
165     size_t code_s, result;
166     input = fopen(argv[2], "r");
167     if( input == NULL )
168     {
169         perror("Error while opening the file.\n");
170         exit(EXIT_FAILURE);
171     }
172     fseek(input, 0, SEEK_END);
173     f_size = ftell(input);
174     fseek(input, 0, SEEK_SET);
175     code_s = sizeof(char) * (f_size+1);
176     covert_message = malloc(code_s);
177     fgets(covert_message, code_s, input);
178     printf("covert_message = %s\n", covert_message);
179     fclose(input);
180     send_message(covert_message, client);
181     close(client);
182     return 0;
183 }

```

A.1.2 Simple OSCTC Server

```

1  #include <sys/types.h>
2  #include <sys/socket.h>
3  #include <sys/wait.h>
4  #include <netinet/in.h>
5  #include <netinet/tcp.h>
6  #include <arpa/inet.h>
7  #include <ctype.h>
8  #include <netdb.h>
9  #include <stdio.h>
10 #include <unistd.h>
11 #include <time.h>
12 #include <pthread.h>
13 #include <string.h>
14 #include <signal.h>

```

```
15 #include <stdlib.h>
16 #include <string.h>
17
18 #define SUCCESS 0
19 #define ERROR 1
20
21 typedef int bool;
22 #define true 1
23 #define false 0
24
25 #ifndef NULL
26 #define NULL ((void *) 0)
27 #endif
28
29 #define SERVER_PORT 20001
30 #define MAXMSG 256
31 #define TIME_SLOT 20
32
33 unsigned int seq_time_slot;
34 bool running = true;
35 unsigned char previous = 0;
36 unsigned char current = 0;
37 unsigned char count = 0;
38 int slot = TIME_SLOT; //msec
39 unsigned int current_symbol = 0;
40 unsigned char count_bits=0;
41 FILE *outputfile;
42 double start_time=0;
43 int received_ones=0;
44 double lasttime;
45
46 int time_slot = TIME_SLOT;
47
48 void error(const char *msg)
49 {
50     perror(msg);
51     exit(1);
52 }
53
54 double calc_time(){
55     double time;
56     char usec[7];
57     double microsec;
58     struct timeval tv;
59     time_t curtime;
60     gettimeofday(&tv, NULL);
61     curtime=tv.tv_sec;
62     sprintf(usec, "%i", tv.tv_usec );
```

```

63     microsec = atoi(usec);
64     time = curtime * 1000 + microsec*0.001;
65     return time;
66 }
67
68 //pthread 1 send one tick every ON-OFF time slot
69 void *sample(void *rate){
70     unsigned char output = 0;
71     seq_time_slot = 0;
72     do{
73         start_time = calc_time();
74         do{
75             if(lasttime > start_time & lasttime <= start_time + time_slot){
76                 current = 1;
77             }
78         }
79         while(calc_time() <= (start_time + *(int*)rate));
80         //increment our time
81         seq_time_slot++;
82
83         if(current == 1){
84             received_ones++;
85         } else {
86             received_ones = 0;
87         }
88         if (seq_time_slot < 8){
89             current_symbol = current_symbol << 1;
90             current_symbol = current_symbol | current;
91         }
92         current = 0;
93         //we received 8 ones, so we received our sync symbol
94         // => reset everything to start
95         if(received_ones == 8){
96             received_ones=0;
97             seq_time_slot = 0;
98             //printf ("Received 8 ones\n");
99             current_symbol = 0;
100         }
101         //check if we received 8 bits
102         if(seq_time_slot==8){
103             output = (char)current_symbol;
104             printf("We received a complete symbol: %c\n", output);
105             if(output <32 | output >127){
106                 fprintf(outputfile, "-");
107             }
108             else{
109                 fprintf(outputfile, "%c", output);
110             }

```

```

111     seq_time_slot = 0;
112     current_symbol = 0;
113 }
114 } while (running);
115 return NULL;
116 }
117
118 //pthread 2 reads covert message in ON-OFF mode
119 void *read_message(void *socket){
120     char buffer[43];
121     do{
122         int bytesReceived = recv(*(int*)socket, buffer, sizeof(buffer), 0);
123         if (bytesReceived > 0){
124             lasttime = calc_time();
125             start_time = lasttime - time_slot / 2;
126         }
127         else if (bytesReceived < 0){
128             perror("cannot read from socket");
129             break;
130         }
131         else
132         {
133             running = false;
134         }
135     } while (running);
136     return NULL;
137 }
138
139 int main(int argc, char *argv[])
140 {
141     int sockfd, newsockfd;
142     socklen_t clilen;
143     char buffer[MAXMSG];
144     struct sockaddr_in serv_addr, cli_addr;
145     int n;
146     pthread_t p1, p2;
147     if(argc > 2) {
148         printf("usage: %s [time slot]\n", argv[0]);
149         exit(-1);
150     }
151     if (argv[1]){
152         time_slot = atoi(argv[1]);
153         slot = time_slot;
154     }
155     sockfd = socket(AF_INET, SOCK_STREAM, 0);
156     if (sockfd < 0){
157         perror("cannot open socket ");
158         return ERROR;

```

```

159     }
160     bzero(&serv_addr, sizeof(serv_addr));
161     serv_addr.sin_family = AF_INET;
162     serv_addr.sin_addr.s_addr = INADDR_ANY;
163     serv_addr.sin_port = htons(SERVER_PORT);
164     if (bind(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) <
165         0){
166         perror("cannot bind port");
167         return ERROR;
168     }
169     listen(sockfd, 5);
170     while(1){ // infinite loop
171         int pid;
172         clilen = sizeof(cli_addr);
173         newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
174         printf("%s: accepted new connection with socket id: %i\n", argv[0],
175             newsockfd);
176         printf("%s: waiting for data on port TCP %u\n", argv[0], SERVER_PORT);
177         //create child processto manage the client
178         pid = fork();
179         if (pid == 0){
180             outputfile = fopen("output_ham.txt", "w");
181             // pthread p1
182             if(pthread_create(&p1, NULL, sample, &slot)) {
183                 fprintf(stderr, "Error creating thread p1\n");
184                 return 1;
185             }
186             // pthread p2
187             if(pthread_create(&p2, NULL, read_message, &newsockfd)) {
188                 fprintf(stderr, "Error creating thread p2\n");
189                 return 1;
190             }
191             if(pthread_join(p1, NULL) && pthread_join(p2, NULL)) {
192                 fprintf(stderr, "Error joining thread p1 or p2\n");
193                 return 2;
194             }
195             printf("%s: waiting for data on port TCP %u\n", argv[0], SERVER_PORT);
196             ;
197             fprintf(outputfile, "\n");
198             fclose(outputfile);
199         }
200     }
201     close(newsockfd);
202     close(sockfd);
203     return 0;
204 }

```

A.2 OSCTC with Manchester Coding

In the enhanced version of our OSCTC we added a further synchronization layer through the use of the Manchester Coding. Since each data element is encoded using at least one transition, i.e. bit 0 is expressed by a low-high transition and bit 1 by a high-low transition we decide to use the couple 01 to denote a covert bit 0 and the couple 10 for the covert bit 1. The other two combinations (00 and 11) are not valid in our scheme.

A.2.1 OSCTC with Manchester Client

```

1 #include <sys/types.h>
2 #include <sys/socket.h>
3 #include <netinet/in.h>
4 #include <netinet/tcp.h>
5 #include <arpa/inet.h>
6 #include <netdb.h>
7 #include <stdio.h>
8 #include <unistd.h> /* close */
9 #include <string.h>
10 #include <stdlib.h>
11 #include <time.h>
12 #include <math.h>
13
14 #define SERVER_PORT 20001
15 #define MAXMSG 256
16 #define TIME_SLOT 20
17 #ifndef NULL
18 #define NULL ((void *) 0)
19 #endif
20 typedef int bool;
21 #define true 1
22 #define false 0
23
24 FILE *input;
25 int time_slot = TIME_SLOT;
26
27 //invia messaggio
28 void send_xbytes(int socket, unsigned int x, void * buffer)
29 {
30     int nSent = 0, nIndex = 0;
31     int nLeft = x;
32     while (nLeft > 0){
33         nSent = send(socket, buffer, nLeft, 0);
34         if (nSent < 0){
35             perror("cannot send message");

```

```

36     close(socket);
37     exit(-1);
38 }
39 nLeft -= nSent;
40 nIndex += nSent;
41 }
42 }
43
44 double calc_time(){
45     double time;
46     char usec[7];
47     double microsec;
48     struct timeval tv;
49     time_t curtime;
50     gettimeofday(&tv, NULL);
51     curtime=tv.tv_sec;
52     sprintf(usec, "%i", tv.tv_usec );
53     microsec = atoi(usec);
54     time = curtime * 1000 + microsec*0.001;
55     return time;
56 }
57
58 //send covert message ON-OFF mode
59 void send_message(char * covert_message, int client){
60     char c;
61     char * message;
62     usleep(time_slot*500);
63     message = "all work and no play makes jack a dull boy\0";
64     int length = strlen(message);
65     double start_time;
66     int waitsyncsymbol = 0;
67     for (int i =0; i < strlen(covert_message); i++)
68     {
69         c = covert_message[i];
70         for(int bitno = 0; bitno < 8; bitno++)
71         {
72             start_time=calc_time();
73             if ((c & 0x80) == 0x80){
74                 send_xbytes(client, length, message);
75                 do{
76                     ;
77                 }while( calc_time()<=(start_time+time_slot));
78                 start_time=calc_time();
79             }
80             else{
81                 do{
82                     ;
83                 }while( calc_time()<=(start_time+time_slot));

```

```

84         start_time=calc_time();
85         send_xbytes(client, length, message);
86     }
87     c = c << 1;
88     do{
89         ;
90     }while(calc_time()<(start_time+time_slot));
91 }// end for(int bitno = 0;bitno < 8;bitno++)
92 waitsyncsymbol=(waitsyncsymbol+1) % 3;
93 if(waitsyncsymbol == 0){
94     //after sending 3 complete bytes we send our sync symbol
95     for(int bitno = 0;bitno < 4;bitno++)
96     {
97         start_time=calc_time();
98         do{
99             ;
100         }while(calc_time()<(start_time+time_slot));
101         send_xbytes(client, length, message);
102     }// end for(int bitno = 0;bitno < 6;bitno++)
103 }
104 }// end for (int i =0; i < strlen(covert_message); i++)
105 }
106
107 /* main() */
108 int main (int argc, char *argv[]) {
109     int client; /* client socket */
110     int rc;
111     struct sockaddr_in local_addr, serv_addr;
112     struct hostent *host;
113     if(argc < 3 || argc > 4) {
114         printf("usage: %s <server> | [<\">> input_file_name.txt <\">>] | [
time slot]\n",argv[0]);
115         exit(-1);
116     }
117     if (argv[3]){
118         time_slot = atoi(argv[3]);
119         printf("time_slot: %i\n\n", time_slot);
120     }
121     /* get host address from specified server name */
122     host = gethostbyname(argv[1]);
123     if (host == NULL)
124     {
125         printf("%s: unknown host '%s'\n", argv[0], argv[1]);
126         exit(-1);
127     }
128     /* now fill in sockaddr_in for remote address */
129     serv_addr.sin_family = host->h_addrtype;
130     /* get first address in host, copy to serv_addr */

```

```

131 memcpy((char *) &serv_addr.sin_addr.s_addr, host->h_addr_list[0], host->
    h_length);
132 serv_addr.sin_port = htons(SERVER_PORT);
133 memset(serv_addr.sin_zero, 0, 8);
134 /* create local stream socket */
135 client = socket(AF_INET, SOCK_STREAM, 0);
136 if (client < 0) {
137     perror("cannot open socket");
138     exit(-1);
139 }
140 /* bind local socket to any port number */
141 local_addr.sin_family = AF_INET;
142 local_addr.sin_addr.s_addr = htonl(INADDR_ANY);
143 local_addr.sin_port = htons(0);
144 memset(local_addr.sin_zero, 0, 8);
145 rc = bind(client, (struct sockaddr *) &local_addr, sizeof(local_addr));
146 int on = 1;
147 setsockopt(rc, IPPROTO_TCP, TCP_NODELAY, &on, sizeof(on));
148 if (rc < 0)
149 {
150     printf("%s: cannot bind port TCP %u\n", argv[0], SERVER_PORT);
151     perror("error");
152     exit(-1);
153 }
154 /* connect to server */
155 rc = connect(client, (struct sockaddr *) &serv_addr, sizeof(serv_addr));
156 if (rc < 0)
157 {
158     perror("cannot connect to server");
159     exit(-1);
160 }
161 long f_size;
162 char* covert_message, *code;
163 size_t code_s, result;
164 input = fopen(argv[2], "r");
165 if( input == NULL )
166 {
167     perror("Error while opening the file.\n");
168     exit(EXIT_FAILURE);
169 }
170 fseek(input, 0, SEEK_END);
171 f_size = ftell(input);
172 fseek(input, 0, SEEK_SET);
173 code_s = sizeof(char) * (f_size+1);
174 covert_message = malloc(code_s);
175 fgets(covert_message, code_s, input);
176 printf("covert_message = %s\n", covert_message);
177 fclose(input);

```

```
178     close(client);
179     return 0;
180 }
```

A.2.2 OSCTC with Manchester Server

```
1 #include <sys/types.h>
2 #include <sys/socket.h>
3 #include <sys/wait.h>
4 #include <netinet/in.h>
5 #include <netinet/tcp.h>
6 #include <arpa/inet.h>
7 #include <ctype.h>
8 #include <netdb.h>
9 #include <stdio.h>
10 #include <unistd.h>
11 #include <time.h>
12 #include <pthread.h>
13 #include <string.h>
14 #include <signal.h>
15 #include <stdlib.h>
16 #include <string.h>
17
18 #define SUCCESS 0
19 #define ERROR 1
20
21 typedef int bool;
22 #define true 1
23 #define false 0
24
25 #ifndef NULL
26 #define NULL ((void *) 0)
27 #endif
28
29 #ifndef MANCHESTER_H
30 #define MANCHESTER_H
31 #endif
32
33 #define SERVER_PORT 20001
34 #define MAX_MSG 256
35 #define TIME_SLOT 20
36
37 unsigned int seq_time_slot;
38 bool running = true;
39 unsigned char previous = 0;
40 unsigned char current = 0;
```

```

41 unsigned char count = 0;
42 int slot = TIME_SLOT; //msec
43 unsigned int current_symbol = 0;
44 unsigned char count_bits=0;
45 FILE *outputfile;
46 double start_time=0;
47 int received_ones=0;
48 double lasttime;
49
50 int time_slot = TIME_SLOT;
51
52 void error(const char *msg)
53 {
54     perror(msg);
55     exit(1);
56 }
57
58 double calc_time(){
59     double time;
60     char usec[7];
61     double microsec;
62     struct timeval tv;
63     time_t curtime;
64     gettimeofday(&tv, NULL);
65     curtime=tv.tv_sec;
66     sprintf(usec, "%i", tv.tv_usec );
67     microsec = atoi(usec);
68     time = curtime * 1000 + microsec*0.001;
69     return time;
70 }
71
72 //send message
73 void send_xbytes(int socket, unsigned int x, void * buffer)
74 {
75     int nSent = 0, nIndex = 0;
76     int nLeft = x;
77     while (nLeft > 0){
78         nSent = send(socket, buffer, nLeft, 0);
79         if (nSent < 0){
80             perror("cannot send message");
81             close(socket);
82             exit(-1);
83         }
84         nLeft -= nSent;
85         nIndex += nSent;
86     }
87 }
88

```

```

89 //pthread 1 send one tick every ON-OFF time slot
90 void *sample(void *rate){
91     unsigned char output = 0;
92     seq_time_slot = 0;
93     do{
94         start_time = calc_time();
95         do{
96             if(lasttime > start_time & lasttime <= start_time + time_slot){
97                 current = 1;
98             }
99         }
100         while(calc_time() <= (start_time + *(int*)rate));
101         //check manchester code 10=1 01=0 11=e 00=e
102         if (previous == 1 & current == 0 & count == 1){
103             current_symbol = current_symbol << 1;
104             current_symbol = current_symbol | 1;
105         } else if (previous == 0 & current == 1 & count == 1){
106             current_symbol = current_symbol << 1;
107         } else if (previous == 1 & current == 1 & count == 1){
108             current_symbol = current_symbol << 2;
109             seq_time_slot++;
110             count = (count + 1) % 2;
111         } else if (previous == 0 & current == 0 & count == 1){
112             current_symbol = current_symbol >> 1;
113             count = (count - 1) % 2;
114         }
115         //increment our time
116         seq_time_slot++;
117         count = (count + 1) % 2;
118         //go on in time with manchester detection
119         if(current == 1){
120             previous = 1;
121             received_ones++;
122         } else{
123             previous = 0;
124             received_ones = 0;
125         }
126         current = 0;
127         //we received 4 ones, so we received our sync symbol
128         // => reset everything to start
129         if(received_ones == 4){
130             received_ones=0;
131             previous = 0;
132             seq_time_slot = 0;
133             current_symbol = 0;
134         }
135         //check if we received 16 bits (8 bit data * 2)
136         if(seq_time_slot==16){

```

```

137     output = (char)current_symbol;
138     printf("We received a complete symbol: %c\n", output);
139     if(output <32 | output>127){
140         fprintf(outputfile, "-");
141     }
142     else{
143         fprintf(outputfile, "%c", output);
144     }
145     seq_time_slot = 0;
146     current_symbol = 0;
147 }
148 } while (running);
149 return NULL;
150 }
151
152 //pthread 2 reads covert message in ON-OFF mode
153 void *read_message(void *socket){
154     char buffer[43];
155     do{
156         int bytesReceived = recv(*(int*)socket, buffer, sizeof(buffer), 0);
157         if (bytesReceived > 0){
158             lasttime = calc_time();
159             start_time = lasttime - time_slot / 2;
160         }
161         else if (bytesReceived < 0){
162             perror("cannot read from socket");
163             break;
164         }
165         else
166         {
167             running = false;
168         }
169     } while (running);
170     return NULL;
171 }
172
173 int main(int argc, char *argv[])
174 {
175     int sockfd, newsockfd;
176     socklen_t clilen;
177     char buffer[MAXMSG];
178     struct sockaddr_in serv_addr, cli_addr;
179     int n;
180     pthread_t p1, p2;
181     if(argc > 2) {
182         printf("usage: %s [time slot]\n", argv[0]);
183         exit(-1);
184     }

```

```

185  if (argv[1]){
186      time_slot = atoi(argv[1]);
187      slot = time_slot;
188      //printf("time_slot: %i\n\n", time_slot);
189  }
190      sockfd = socket(AF_INET, SOCK_STREAM, 0);
191
192      if (sockfd < 0){
193          perror("cannot open socket ");
194          return ERROR;
195      }
196      bzero(&serv_addr, sizeof(serv_addr));
197      serv_addr.sin_family = AF_INET;
198      serv_addr.sin_addr.s_addr = INADDR_ANY;
199      serv_addr.sin_port = htons(SERVER_PORT);
200      if (bind(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) <
201  0){
202          perror("cannot bind port");
203          return ERROR;
204      }
205      listen(sockfd, 5);
206      while(1){ // infinite loop
207          int pid;
208          clilen = sizeof(cli_addr);
209          newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
210          printf("%s: accepted new connection with socket id: %i\n", argv[0],
211  newsockfd);
212          printf("%s: waiting for data on port TCP %u\n", argv[0], SERVER_PORT);
213          //creo processo figlio per gestire il client
214          pid = fork();
215          if (pid == 0){
216              outputfile = fopen("output.ham.txt", "w");
217              // pthread p1
218              if(pthread_create(&p1, NULL, sample, &slot)) {
219                  fprintf(stderr, "Error creating thread p1\n");
220                  return 1;
221              }
222              // pthread p2
223              if(pthread_create(&p2, NULL, read_message, &newsockfd)) {
224                  fprintf(stderr, "Error creating thread p2\n");
225                  return 1;
226              }
227              if(pthread_join(p1, NULL) && pthread_join(p2, NULL)) {
228                  fprintf(stderr, "Error joining thread p1 or p2\n");
229                  return 2;
230              }
231          }
232          printf("%s: waiting for data on port TCP %u\n", argv[0], SERVER_PORT)
233  ;

```

```

230     fprintf(outputfile, "\n");
231     fclose(outputfile);
232 }
233 }
234 close(newsockfd);
235 return 0;
236 close(sockfd);
237 return 0;
238 }

```

A.3 OSCTC Manchester + Hamming

In the last version of our OSCTC we added the Hamming Code (12, 8). Such a scheme encodes 8 covert bits in 12 bits, adding 4 redundant bits. In this way, with a 50% of increase of the word length, we are able to detect 2 covert bit errors and correct 1 covert bit error. To implement such code we used two different matrices necessary to encode (client side) and decode (server side) the covert message.

A.3.1 OSCTC Manchester + Hamming Client

The Hamming Code matrix necessary to encode the covert message is used on the client in the `HammingTableEncode` function:

```

1 unsigned int HammingTableEncode(unsigned char data)
2 {
3     return hammingCodes[data];
4 }

```

Referring to the OSCTC Manchester Client source code, such function is called in the `send_message` function through the command `coded_c=HammingTableEncode(c);` where the variable `c` is defined as: `int coded_c = 0;`.

A.3.2 OSCTC Manchester + Hamming Server

The Hamming Code matrix necessary to decode the covert message is used on the client in the `HammingTableDecode` function:

```

1 unsigned char HammingTableDecode(unsigned int code)
2 {
3     return hammingDecodeValues[code];
4 }

```

Referring to the OSCTC Manchester Server source code, such function is called in the `sample` function through the command `output=HammingTableDecode(current_symbol);`. Here is important to notice that a covert byte, after the Manchester coding, raises to 16 bits. When applying our Hamming coding, we have 8 bits of data + 4 parity bits that become, after the Manchester coding, 24 bits. It means that in the `sample` function we have to modify the lines:

```
1 //check if we received 16 bits (8 bit data * 2)
2     if(seq_time_slot==16){
3         output = (char)current_symbol;
```

with the following:

```
1 //check if we received 24 bits (8 bit data + 4 parity bit)
2     if(seq_time_slot==24){
3         output = HammingTableDecode(current_symbol);
```

A.4 Passive OSCTC

A.4.1 Passive OSCTC Client

The Passive OSCTC Client exactly corresponds to our Active OSCTC that uses the Manchester Coding and the Hamming Code.

A.4.2 Passive OSCTC Server

The Passive OSCTC Server has the same structure of our Active OSCTC that uses the Manchester Coding and the Hamming Code, but with some differences. One difference is the presence of the new function `create_html_output_for_binary()`, used in our case to simplify our implementation. With this function we emulated a web server, building a specific HTTP message that is sent to the user browser. After run the server and the client, to visualize the picture is necessary to point the browser to the URL: `127.0.0.1:<port_number>`. *Function create_html_output_for_binary*

```
1 //=====generate http response for binary data=====
2 char *create_html_output_for_binary() {
3
4     FILE *file ;
5     char *file_buf;           // image is copied in this buffer
6
```



```

6      struct sockaddr_in client_addr;           // structure to hold client
        address
7      unsigned int cli_size;                   // length of address
8      pthread_t p1;
9      pthread_t p2;
10     printf("\n\t***-***PROXY SERVER STARTED***\n");
11     // check arguments
12     if( argc == 2 )
13     {
14         portnum = atoi(argv[1]);
15     }
16     else
17     {
18         printf("usage: %s <server_port>\n", argv[0]);
19     }
20
21     client_sockfd = socket(AF_INET, SOCK_STREAM, 0);
22     if(client_sockfd < 0){
23         printf("server - socket() error\n");    // debugging
24         exit(1);
25     }
26     else
27         printf("socket\t\tcreated\n");
28     //set info of server
29     bzero(&server_addr , sizeof(server_addr));
30     memset(buffer , 0, MAXBUF);
31     memset(buffer2 , 0, MAXBUF);
32     fwrite(buffer2 , 1, sizeof(buffer2), stdout);
33     server_addr.sin_family=AF_INET;
34     server_addr.sin_addr.s_addr = INADDR_ANY;
35     server_addr.sin_port = htons(portnum);
36     client_sockfd = socket(AF_INET, SOCK_STREAM, 0);
37     if(bind(client_sockfd , (struct sockaddr *)&server_addr , sizeof(
        server_addr)) < 0){
38         printf("binding failed\n");
39         exit(1);
40     }
41     else printf("binding\t\tsuccess\n\n");
42     // socket on listening mode
43     listen(client_sockfd , 5);
44     printf("waiting for client\n");
45     while(1)                                   // infinite loop
46     {
47         int pid;
48         cli_size = sizeof(client_addr);
49         client_new_fd = accept(client_sockfd , (struct sockaddr *)&client_addr ,
            &cli_size);

```

```

50     printf("server got connection from %s & %d\n\n",inet_ntoa(client_addr.
sin_addr), client_new_fd);
51     pid = fork();
52     if (pid == 0){
53         int bytes_read;
54         bytes_read = recv(client_new_fd, buffer, MAXBUF, 0);
55         if(bytes_read == 0)
56         {
57             printf("\nERROR READING SOCKET\n\n");
58             exit(1);
59         }
60         if(bytes_read > 0)
61         {
62             //fwrite(buffer, 1, sizeof(buffer), stdout);
63         }
64         http = create_html_output_for_binary();
65         //start reading covert channel file and sending it through image
66         long f_size;
67         char* covert_message;
68         size_t code_s;
69         input = fopen("input_it.txt", "r");
70         if( input == NULL )
71         {
72             perror("Error while opening the file.\n");
73             exit(EXIT_FAILURE);
74         }
75         fseek(input, 0, SEEK_END);
76         f_size = ftell(input);
77         fseek(input, 0, SEEK_SET);
78         code_s = sizeof(char) * (f_size+1);
79         covert_message = malloc(code_s);
80         fgets(covert_message, code_s, input);
81         printf("covert_message = %s\n", covert_message);
82         fclose(input);
83         int carrier_size = lenght_header+fileLen;                                // in
bytes
84         int remainder = carrier_size%size;
85         int num_packet = ((lenght_header+fileLen)%size)==0?((lenght_header+
fileLen)/size):
86             ((lenght_header+fileLen)/size+1); //carrier size in number of
packets of length:size+possible mod
87         double covert_message_bits = strlen(covert_message)*((8+4)*2+4/3);
88         if(covert_message_bits/2 < num_packet)
89         {
90             send_message(covert_message, client_new_fd);
91         }
92         else{
93             printf("Reduce PACKET_SIZE to send covert message!\n");

```

```
94     }
95     //end reading covert channel file and sending it through image
96 }
97 close(client_new_fd);
98 }
99 close(client_sockfd);
100 }
101 /**main ends**/
```

Appendix B

NSA Cipher Suites

IN THIS APPENDIX the correlation between NSA-approved security algorithms and data classification levels is reported. Table B.2 refers to the Cipher Suite B, whereas Table B.1 refers to the Cipher Suite A that uses NIST-approved legacy cryptographic algorithms which may be used in place of Suite B cryptographic algorithms with identical functions. The highest classification levels are SECRET (S) and TOP SECRET (TS).

In addition, NSA advises to begin implementing a layer of quantum resistant protection, that is possible today through the use of large symmetric keys and specific secure protocol standards. For example it would be recommended, when using an IKE/IPsec layer, to use RFC 2409-conformant implementations of the IKE standard (IKEv1) together with large, high-entropy, pre-shared keys and the AES-256 encryption algorithm.

“RFC 2409 is the only version of the IKE standard that leverages symmetric pre-shared keys in a manner that may achieve quantum resistant confidentiality. Additionally, MACsec key agreement as specified in IEEE 802.1X-2010, and the RFC 4279 TLS specification provide further options for implementing quantum resistant security measures. These options also involve key agreement schemes that leverage large symmetric pre-shared keys” [192].

TABLE B.1: NSA Cipher Suite A

Algorithm	Function	Specification	Parameters
Diffie-Hellman (DH) Key Exchange	Asymmetric algorithm used for key establishment	NIST SP 800-56A	2048-bit modulus to protect up to S.
Digital Signature Algorithm (DSA)	Asymmetric algorithm used for digital signatures	FIPS PUB 186-3	2048-bit modulus to protect up to S.
RSA	Asymmetric algorithm used for digital signatures	FIPS PUB 186-3	2048-bit modulus to protect up to S.

TABLE B.2: NSA Cipher Suite B

Algorithm	Function	Specification	Parameters
Advanced Encryption Standard (AES)	Symmetric block cipher used for information protection	FIPS PUB 197	128 bit keys to protect up to S. 256 bit keys to protect up to TS.
Elliptic Curve Diffie-Hellman (ECDH) Key Exchange	Asymmetric algorithm used for key establishment	NIST SP 800-56A	Curve P-256 to protect up to S. Curve P-384 to protect up to TS.
Elliptic Curve Digital Signature Algorithm (ECDSA)	Asymmetric algorithm used for digital signatures	FIPS PUB 186-3	Curve P-256 to protect up to S. Curve P-384 to protect up to TS.
Secure Hash Algorithm (SHA)	Algorithm used for computing a condensed representation of information	FIPS PUB 180-4	SHA-256 to protect up to S. SHA-384 to protect up to TS.
Diffie-Hellman (DH) Key Exchange	Asymmetric algorithm used for key establishment	IETF RFC 3526	3072-bit modulus to protect up to TS.
RSA	Asymmetric algorithm used for key establishment	NIST SP 800-56B rev 1	3072-bit modulus to protect up to TS.
RSA	Asymmetric algorithm used for digital signatures	FIPS PUB 186-4	3072-bit modulus to protect up to TS.

Appendix C

Weibull Probability Density Function

IN THIS APPENDIX we first give some details about the Weibull Probability Density Function (Weibull PDF). Then, we briefly describe how to estimate the parameters of a Weibull distribution using the technique of [170].

C.1 Weibull Distribution

The Weibull distribution is related to a number of other probability distributions [193]. It interpolates between the exponential distribution ($k = 1$) and the Rayleigh distribution ($k = 2$ and $\lambda = \frac{\sqrt{2}}{k}$). The shapes of Exponential and Rayleigh distribution are depicted in Fig. C.1¹ and Fig. C.2², respectively.

The shape of the Weibull PDF drastically changes with the value of k .

For $0 < k < 1$, the PDF tends to ∞ as x approaches zero from above and is strictly decreasing.

For $k = 1$, the PDF tends to $\frac{1}{B}$ as x approaches zero from above and is strictly decreasing.

For $k > 1$, the PDF tends to zero as x approaches zero from above, increases until its mode and decreases after it.

It is interesting to note that the PDF has infinite negative slope at $x = 0$ if $0 < k < 1$, infinite positive slope at $x = 0$ if $1 < k < 2$ and null slope at $x = 0$ if $k > 2$. For $k = 2$ the PDF has a finite positive slope at $x = 0$.

As k goes to infinity, the Weibull distribution converges to a Dirac Delta Distribution

¹CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=73793>.

²CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=140795>.

centered at $x = \lambda$. Moreover, the skewness and coefficient of variation depend only on the shape parameter. Hence, the knowledge about the values of the shape and scale parameters is of mandatory importance to fully understand the statistics of the Weibull PDF.

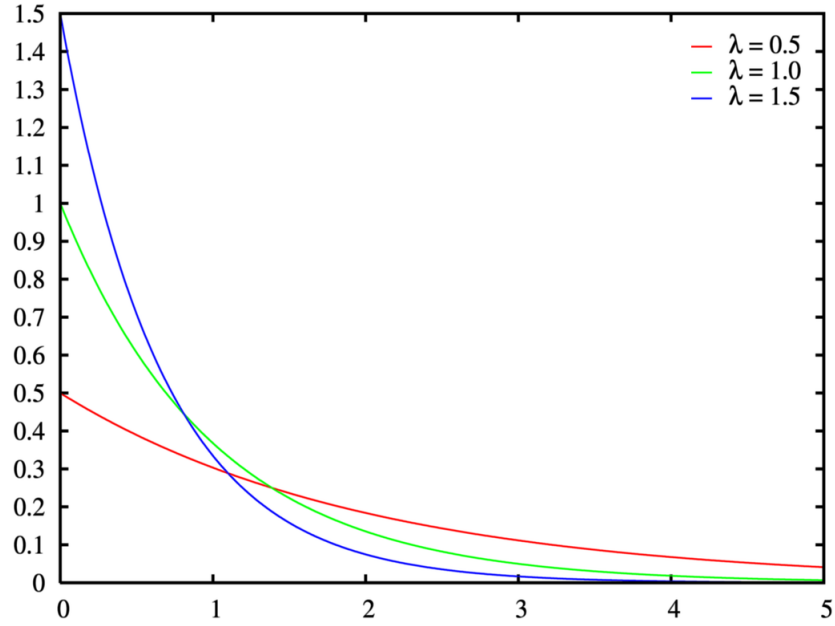


FIGURE C.1: Exponential Probability Density Function

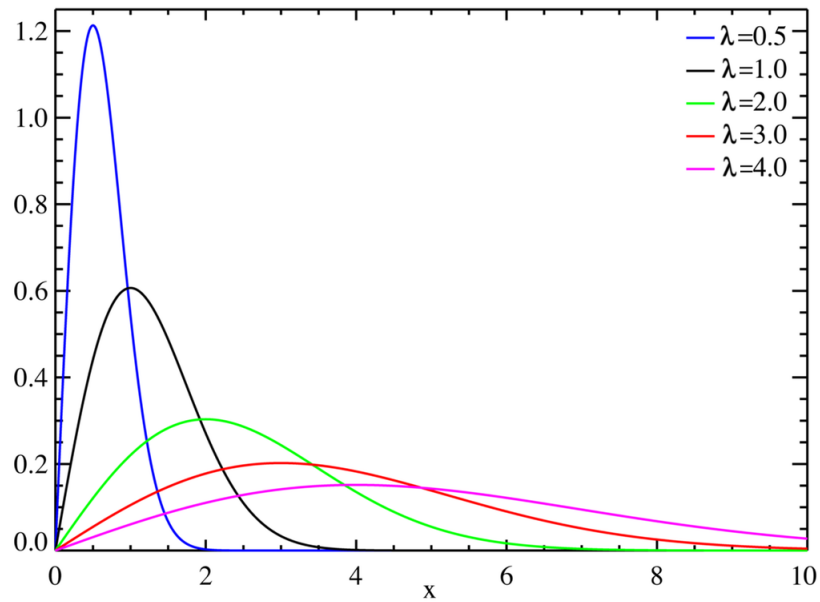


FIGURE C.2: Rayleigh Probability Density Function

Methods for estimating Weibull parameters can be classified into two main categories, graphical and analytic methods. In [170], a common graphical method, called the *median-rank method*, has been used because of its simplicity and speed. In this method,

the data are first sorted and a Median Rank (MR) is assigned to each item, calculated as follows:

$$M_R = \frac{R-0.3}{N+0.4}$$

where R is the rank of each item in the sorted list, and N is the length of the data. Then, α and β are computed respectively as the slope and $y - intercept$ of the line consisting of plotting $\ln \ln \frac{1}{1-MR}$ versus $\ln data$, where \ln stands for the natural logarithm.

Finally, the scale and shape parameters are derived from the equations:

$$\hat{k} = \alpha$$

and

$$\hat{\lambda} = e^{-\frac{\alpha}{\beta}}$$

Bibliography

- [1] GAO. Information assurance, national partnership offers benefits, but faces considerable challenges, 2006. URL www.gao.gov/new.items/d06392.pdf.
- [2] Cosmo-SkyMED Program, 2015. URL www.cosmo-skymed.it/en/index.htm.
- [3] MUSIS Program, 2015. URL www.defense.gouv.fr/dga/equipement/information-communication-espace/musis.
- [4] J. P. L. Woodward. Applications for multilevel secure operating systems, 1979. URL www.computer.org/csdl/proceedings/afips/1979/5087/00/50870319.pdf.
- [5] Committee on National Security Systems, 2010. URL www.cnss.gov/Assets/pdf/cnssi_4009.pdf.
- [6] Raytheon. Cross-domain access transfer, 2015. URL www.forcepoint.com/product/cross-domain-access-transfer/trusted-thin-client.
- [7] NATO, 2015. URL www.cso.nato.int/ACTIVITY_META.asp?ACT=2226.
- [8] ver. 3.1 Common Criteria for Information Technology Security Evaluation, 2015. URL www.commoncriteriaportal.org.
- [9] Orange Book Preamble, 1985. URL www.kernel.org/pub/linux/libs/security/Orange-Linux/refs/Orange/Orange0.html.
- [10] D. E. Bell. Looking back: Addendum. *Twenty-Second Annual Computer Security Applications Conference (ACSAC)*, 2006.
- [11] D. F. C. Brewer and M. J. Nash. The chinese wall security policy, 1989.
- [12] C. Weissman. Security controls in the adept-50 time-sharing system. *Proceedings of the 1969 Fall Joint Computer Conference*, pages 119–133, 1969.
- [13] C. Weissman. Ibm’s resource security system (rss). *Security and Privacy in Computer Systems*, pages 379–401, 1973.

- [14] Compatible Time-Sharing System, 1965. URL www.multicians.org/thvv/7094.html.
- [15] J. H. Saltzer. Protection and the control of information sharing in multics. *Commun. ACM*, 17(7):388–402, July 1974. doi: 10.1145/361011.361067. URL <http://doi.acm.org/10.1145/361011.361067>.
- [16] F. J. Corbato and V. A. Vyssotsky. Introduction and overview of the multics system. In *Proceedings of the November 30-December 1, 1965, Fall Joint Computer Conference, Part I*, pages 26–34, December 1965.
- [17] L. J. Freim. Scomp: a solution to the multilevel security problem. *Computer*, 16(7):26–34, 1983.
- [18] C. Weissman. Blacker: security for the ddn examples of a1 security engineering trades. In *Proceedings 1992 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 286–292, December 1992.
- [19] C. W. Flink II and J. D. Weiss. System v/mls labeling and mandatory policy alternatives. *AT&T Technical Journal*, 67(3):53–64, 1988.
- [20] J. Cashin. Multilevel security controls access: Secureware product segments air force system. *Journal Software Magazine*, 14(1):89, 1994.
- [21] R. M. Wong. A comparison of secure unix operating systems. *Proceedings of the Sixth Annual*, pages 322–333, 1990.
- [22] M. H. Kang and I. S. Moskowitz. A pump for rapid, reliable, secure communication. *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 119–129, 1993.
- [23] D. E. Bell. Looking back at the bell-la padula model. *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 119–129, 2005.
- [24] J. P. Anderson. Computer security technology planning study. *Technical Report ESDTR-73-51*, 1 and 2, 1972.
- [25] FreeBSD, 2015. URL www.freebsd.org.
- [26] R. Spencer, S. Smalley, P. Loscocco, M. Hibler, Anderson D., and J. Lepreau. The flask security architecture: system support for diverse security policies, 1999.
- [27] AppArmor application security for Linux, 2015. URL www.suse.com/documentation/apparmor.

- [28] Windows Integrity Mechanism, 2015. URL <https://msdn.microsoft.com/en-us/library/bb625957.aspx>.
- [29] M. Bailey. The unified cross domain management office: Bridging security domains and cultures, 2008. URL www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA487191.
- [30] F. Benedetto, G. Giunta, A. Liguori, and A. Wacker. A novel method for securing critical infrastructures by detecting hidden flows of data. In IEEE, editor, *Communications and Network Security (CNS), 2015 IEEE Conference on*, pages 648–654, September 2015. doi: 10.1109/CNS.2015.7346881.
- [31] General Dynamics Cross-Domain Guards, 2015. URL <https://gdmissonsyste.ms.com/cyber/products/trusted-computing-cross-domain/tactical-cross-domain-guards>.
- [32] Forcepoint Small Format Guard, 2015. URL www.forcepoint.com/resources/datasheets/small-format-guard.
- [33] Lockheed Martin Cross Domain Cyber Solutions, 2015. URL www.lockheedmartin.com/us/products/tman.html.
- [34] Thales Security Products, 2015. URL www.thalesgroup.com/en/norway/security-products-and-services.
- [35] Advatech Pacific Cross-Domain Solution, 2015. URL www.tacticalcds.com/Technology/Technology_Overview.html.
- [36] OWL MCDS, 2015. URL www.owlcti.com/government/defense/def_mcds_datasheet.html.
- [37] Network Working Group. Rfc4949: Internet security glossary, version 2, 2007. URL <http://tools.ietf.org/html/rfc4949>.
- [38] A. Liguori, F. Benedetto, G. Giunta, N. Kopal, and A. Wacker. Analysis and monitoring of hidden TCP traffic based on an open-source covert timing channel. In IEEE, editor, *Communications and Network Security (CNS), 2015 IEEE Conference on*, pages 667–674, September 2015. doi: 10.1109/CNS.2015.7346885.
- [39] OWL Whitepaper, 2009. URL www.owlcti.com/pdfs/crossBorderInfoSharing_ISGIG_r01n_coverIncl.pdf.
- [40] C. Boettcher, R. DeLong, J. Rushby, and W. Sifre. The mils component integration approach to secure information sharing, 2008.
- [41] ARINC-653, 2015. URL <http://store.aviation-ia.com/cf/store>.

- [42] DO-178B, 2015. URL www.rtca.org/Files/ListofAvailableDocsMarch2013.pdf.
- [43] J. Rushby. Partitioning for safety and security: requirements, mechanisms, and assurance, 1999.
- [44] J. Rushby. Proof of separability: A verification technique for a class of security kernels. *Proceedings of the 5th International Symposium on Programming*, 137: 352–367, 1982.
- [45] Information Assurance Directorate. U.s. government protection profile for separation kernels in environments requiring high robustness version 1.03, 2007.
- [46] NIAP. Skpp sunseting, 2011. URL www.niap-ccevs.org/pp/archived/PP_SKPP_HR_V1.03/.
- [47] Lynx, 2015. URL www.lynx.com/products/secure-virtualization/lynxsecure-separation-kernel-hypervisor.
- [48] T. E. Levin, C. E. Irvine, and T.D. Nguyen. Least privilege in separation kernels, 2008.
- [49] Wind River, 2015. URL http://windriver.com/products/vxworks/certification-profiles/#vxworks_mils.
- [50] EURO-MILS, 2015. URL www.euromils.eu.
- [51] Green Hills, 2008. URL www.niap-ccevs.org/cc-scheme/st/st_vid10119-ci.pdf.
- [52] System and Network Analysis Center Information Assurance Directorate. Separation kernel on commodity workstations, 2010. URL www.niap-ccevs.org/announcements/SeparationKernelonCommodityWorkstations.pdf.
- [53] Selex ES, 2013. URL www.aofs.org/wp-content/uploads/2013/10/131010.06-Selex-ES-Raschell.pdf.
- [54] R. Buerki and A. K. Rueeggsegger. Muen-an x86/64 separation kernel for high assurance, 2013. URL <http://muen.codelabs.ch/muen-report.pdf>.
- [55] Bertin Technologies Polyxene, 2009. URL www.bertin-it.com/brochure/PolyXene-high-security-hypervisor.pdf.
- [56] Bertin Technologies Polyxene, 2014. URL <http://bit.ly/1JhySqj>.

- [57] Business Wire Article, 2009. URL www.businesswire.com/news/home/20091201006186/en/Bertin%E2%80%99s-polyXene%E2%84%A2-Multi-Sensibilities-Secure-Operating-System-Integrating.
- [58] MITRE Corporation, 2015. URL <http://www.mitre.org>.
- [59] ITSEC Provisional Harmonised Criteria. Information technology security evaluation criteria v. 1.2, 1991. URL <http://www.mitre.org>.
- [60] Common Criteria Portal. About the common criteria, 2015. URL www.commoncriteriaportal.org/ccra/.
- [61] Common Criteria Management Board. Common criteria for information technology security evaluation part 3: Security assurance requirements; version 3.1, revision 4, 2012. URL www.commoncriteriaportal.org/.
- [62] J. Rushby. 12th ieee international conference on the engineering of complex computer systems (iceccs), 2007.
- [63] D-MILS, 2015. URL www.d-mils.org/.
- [64] National Institute of Standards and Technology, 2015. URL <http://csrc.nist.gov/groups/STM/cavp/validation.html>.
- [65] Common Criteria Management Board. Common criteria for information technology security evaluation part 2: Security functional components; version 3.1, revision 4, 2012. URL www.commoncriteriaportal.org/.
- [66] Committee on National Security Systems. National information assurance policy on the use of public standards for the secure sharing of information among national security systems, 2012. URL www.cnss.gov/CNSS/openDoc.cfm?fGEj+ExMnLgu5sxAutbbrw==.
- [67] M. Goll and S. Gueron. Vectorization of chacha stream cipher. *Information Technology: New Generations (ITNG), 2014 11th International Conference on*, pages 612–615, April 2014. doi: 10.1109/ITNG.2014.33.
- [68] A. Langley. Chacha20 and poly1305 for ietf protocols, 2015. URL <https://tools.ietf.org/html/rfc7539>.
- [69] A. Biryukov, D. Khovratovich, and I. Nikolić. Distinguisher and related-key attack on the full aes-256 (extended version). Cryptology ePrint Archive, Report 2009/241, 2009. <http://eprint.iacr.org/>.

- [70] A. Biryukov and D. Khovratovich. Related-key cryptanalysis of the full aes-192 and aes-256. Cryptology ePrint Archive, Report 2009/317, 2009. <http://eprint.iacr.org/>.
- [71] A. Bogdanov, D. Khovratovich, and C. Rechberger. Biclique cryptanalysis of the full aes. In *Advances in Cryptology-ASIACRYPT 2011*, pages 344–371. Springer, 2011.
- [72] B. Schneier. New attack on aes, 2011. URL www.schneier.com/blog/archives/2011/08/new_attack_on_a_1.html.
- [73] Certicom Research. Sec 2: Recommended elliptic curve domain parameters, September 2000. URL www.secg.org/SEC2-Ver-1.0.pdf.
- [74] J. M. Ericsson. Overview and analysis of overhead caused by tls. draft, January 2014. URL <https://tools.ietf.org/html/draft-mattsson-uta-tls-overhead-00>.
- [75] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen, and T. Kivinen. Internet key exchange protocol version 2 (ikev2). RFC 7296 (Internet Standard), October 2014. URL www.ietf.org/rfc/rfc7296.txt.
- [76] T. Dierks and E. Rescorla. The transport layer security (tls) protocol version 1.2. RFC 5246 (Proposed Standard), August 2008. URL www.ietf.org/rfc/rfc5246.txt. Updated by RFCs 5746, 5878, 6176, 7465, 7507, 7568, 7627, 7685.
- [77] BlueKrypt. Cryptographic key length recommendation, 2016. URL www.keylength.com/en/compare.
- [78] BlueKrypt. Cryptographic key length recommendation, 2016. URL www.keylength.com/en/8.
- [79] ENISA. Algorithms, key sizes and parameters report - 2014, 2014. URL www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/algorithms-key-size-and-parameters-report-2014/at_download/fullReport.
- [80] A. Hoban. Using intel aes new instructions and pclmulqdq to significantly improve ipsec performance on linux. Intel White Paper, 2010. URL www.intel.com/content/dam/www/public/us/en/documents/white-papers/aes-IPsec-performance-linux-paper.pdf.
- [81] S. Turner, D. Brown, K. Yiu, R. Housley, and T. polk. Elliptic curve cryptography subject public key information, March 2009. URL <https://www.ietf.org/rfc/rfc5480.txt>.

- [82] V. Gupta, D. Stebila, F. Fung, S. C. Shantz, N. Gura, and Eberle H. Speeding up secure web transactions using elliptic curve cryptography. *Network and Distributed System Security Symposium*, 2004.
- [83] N. Mavrogiannopoulos. The price to pay for perfect-forward secrecy, 2011. URL <http://nmav.gnutls.org/2011/12/price-to-pay-for-perfect-forward.html>.
- [84] L. S. Huang, S. Adhikarla, D. Boneh, and C. Jackson. An experimental study of tls forward secrecy deployments. *IEEE Internet Computing*, 18(6):43–51, Nov 2014. ISSN 1089-7801. doi: 10.1109/MIC.2014.86.
- [85] C. Souradeep. Exploring ikev2 with ecdsa certificate in ibm aix. IBM developerWorks, 2014. URL www.ibm.com/developerworks/aix/library/au-aix-exploring-ikev2-with-ecdsa-certificate/au-aix-exploring-ikev2-with-ecdsa-certificate-pdf.pdf.
- [86] J. Sundberg. Using end-to-end ipsec for secure inter- and intrasite network communication. *M.Sc. Thesis*, 2011. doi: TRITA-CSC-E2011:088,ISRN-KTH/CSC/E--11/088--SE,ISSN-1653-5715.
- [87] S. Srivatsan, M. L. Johnson, and S. M. Bellovin. Simple-vpn: Simple ipsec configuration. *Technical Report*, 2010. URL <http://doi.acm.org/10.1145/361011.361067>.
- [88] O. Kim and D. Montgomery. Behavioral and performance characteristics of ipsec/ike. In *in Large-Scale VPNs*, [Available Online] <http://w3.antd.nist.gov/pubs/cnis-perf-vpnsikev1.pdf>, 2009.
- [89] C. Shue, Y. Shin, M. Gupta, and J. Y. Choi. Analysis of ipsec overheads for vpn servers. In *1st IEEE ICNP Workshop on Secure Network Protocols (NPSec)*, pages 25–30, November 2005. doi: 10.1109/NPSEC.2005.1532049.
- [90] BSD Router Project. Ipvsec performance lab of an ibm system x3550 m3 with intel 82580, 2015. URL http://bsdrrp.net/documentation/examples/ipvsec_performance_lab_of_an_ibm_system_x3550_m3_with_intel_82580#method_used.
- [91] S. Klassert. Parallelizing ipsec: switching smp to 'on' is not even half the way, 2010. URL www.strongswan.org/docs/Steffen_Klassert_Parallelizing_IPsec.pdf.
- [92] Freescale Semiconductor. Understanding cryptographic performance. Freescale Semiconductor White Paper, 2008. URL www.embeddeddeveloper.com/news_letter/files/CRYPTOWP_Rev2.pdf.

- [93] Libreswan. Benchmarking and performance testing, 2015. URL https://libreswan.org/wiki/Benchmarking_and_Performance_testing.
- [94] J. Granjal, E. Monteiro, and J. Sa Silva. On the feasibility of secure application-layer communications on the web of things. In *Local Computer Networks (LCN), 2012 IEEE 37th Conference on*, pages 228–231. IEEE, 2012.
- [95] R. Mzid, M. Boujelben, H. Youssef, and M. Abid. Adapting tls handshake protocol for heterogenous ip-based wsn using identity based cryptography. In *2010 International Conference on Wireless and Ubiquitous Systems*, pages 1–8, Oct 2010. doi: 10.1109/ICWUS.2010.5671367.
- [96] H. Tschofenig and M. Pegourie-Gonnard. Performance of state-of-the-art cryptography on arm-based microprocessors, 2015. URL <http://csrc.nist.gov/groups/ST/lwc-workshop2015/presentations/session7-vincent.pdf>.
- [97] D. Murray and T. Koziniec. The state of enterprise network traffic in 2012. In *Communications (APCC), 2012 18th Asia-Pacific Conference on*, pages 179–184, Oct 2012. doi: 10.1109/APCC.2012.6388126.
- [98] K. Pentikousis and H. Badr. Quantifying the deployment of tcp options - a comparative study. *IEEE Communications Letters*, 8(10):647–649, Oct 2004. ISSN 1089-7798. doi: 10.1109/LCOMM.2004.835308.
- [99] W. John and S. Tafvelin. Analysis of internet backbone traffic and header anomalies observed. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, IMC '07*, pages 111–116, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-908-1. doi: 10.1145/1298306.1298321. URL <http://doi.acm.org/10.1145/1298306.1298321>.
- [100] P. Hurtig, W. John, and A. Brunstrm. International conference on networking and services (icns '11). *IEEE Communications Letters*, 2011.
- [101] E. Garsva, N. Paulauskas, and G. Grazulevicius. Packet size distribution tendencies in computer network flows. In *Electrical, Electronic and Information Sciences (eStream), 2015 Open Conference of*, pages 1–6, April 2015. doi: 10.1109/eStream.2015.7119483.
- [102] J. Mattsson. Ietf 90 uta overview and analysis of tls overhead, 2015. URL <https://www.ietf.org/proceedings/90/slides/slides-90-uta-5.pdf>.
- [103] S. Kent. Ip encapsulating security payload (esp). RFC 4303 (Proposed Standard), December 2005. URL <http://www.ietf.org/rfc/rfc4303.txt>.

- [104] L. Law and J. Solinas. Suite b cryptographic suites for ipsec. RFC 6379 (Informational), October 2011. URL <http://www.ietf.org/rfc/rfc6379.txt>.
- [105] Cisco. Isec overhead calculator, 2016. URL <https://cway.cisco.com/tools/ipsec-overhead-calc/ipsec-overhead-calc.html>.
- [106] M. Dworkin. Recommendation for block cipher modes of operation: Galois/counter mode (gcm) and gmac, 2007. URL <http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf>.
- [107] Military & Aerospace Electronics, 2011. URL www.militaryaerospace.com/articles/2011/11/rockwell-collins-secureone.html.
- [108] Rockwell Collins AAMP7G, 2015. URL www.rockwellcollins.com/~media/Files/Unsecure/Products/Product%20Brochures/Information%20Assurance/Crypto/AAMP7G%20data%20sheet.aspx.
- [109] A. Liguori. A novel Multiple Independent Levels of Security/Safety Cross Domain Solution. In IEEE, editor, *Military Communications Conference (MILCOM), 2015 IEEE Conference on*, pages 1578–1583, October 2015. doi: 10.1109/MILCOM.2015.7357670.
- [110] A. Liguori, F. Benedetto, G. Giunta, N. Kopal, and A. Wacker. SoftGap: A Multi Independent Levels of Security Cross-Domain Solution. In IEEE, editor, *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*, pages 754–759, August 2015. doi: 10.1109/FiCloud.2015.84.
- [111] J. Johnson. Roi: It’s your job. In *Proceedings of the Third International Conference on Extreme Programming and Flexible Processes in Software Engineering (XP 2002)*, 2002.
- [112] Jones C. Applied software measurement. In *NY: McGraw-Hill*, 1997.
- [113] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cullar, P. H. Drielsma, P. Ham, O. Kouchnarenko, J. Mantovani, S. Mdersheim, D. V. Oheimb, M. Rusinowitch, J. Santiago, L. Turuani, M. andVigano, and L. Vigneron. The avispa tool for the automated validation of internet security protocols and applications. *Computer Aided Verification*, pages 281–285, 2005.
- [114] AVISPA. Deliverable 2.1: The high-level protocol specification language, 2015. URL www.avispa-project.org/publications.html.
- [115] C. J. F. Cremers, P. Lafourcade, and P. Nadeau. Comparing state spaces in automatic security protocol analysis. *Lecture Notes in Computer Science*, 5458: 70–94, 2009.

- [116] C. J. F. Cremers and S. Mauw. Operational semantics of security protocols. *Leue, S. and Syst, T., Scenarios: Models, Transformations and Tools, Revised Selected Papers*, 3466, 2005.
- [117] EURO-MILS. Used formal methods. *Secure European Virtualization for Trustworthy Applications in Critical Domains*, 2015. URL www.euromils.eu/downloads/Deliverables/Y2/2015-EM-UsedFormalMethods-WhitePaper-October2015.pdf.
- [118] Nessus Vulnerability Scanner, 2015. URL www.tenable.com/products/nessus-vulnerability-scanner.
- [119] Kali Linux, 2015. URL www.kali.org.
- [120] CERTViT. Signaturerstellungseinheit tcos 3.0 signature card version 2.0 release 1/sle78clx1440p, 2012. URL www.src-gmbh.de/fileadmin/redaktion/pdf/common_criteria/BST_SRC.00016.TE.11.2012__T-Systems_SigCard_auf_Infineon__S1mU-1.pdf.
- [121] CCRA. Arrangement on the recognition of common criteria certificates in the field of information technology security, 2014. URL bit.ly/1UoNjeZ.
- [122] SOG-IS Agreement, 2015. URL <http://sogis.org>.
- [123] R. Koolen and J. Schmaltz. Formal methods for mils: Formalisations of the gvw firewall. *International Workshop on MILS: Architecture and Assurance for Secure Systems*, 2009. URL http://www.win.tue.nl/~jschmalt/publications/mils15/12-mils15_submission_4.pdf.
- [124] J. Simmons. The prisoners problem and the subliminal channel. *Proceedings Advances in Cryptology*, pages 51–67, 1983.
- [125] B.W. Lampson. A note on the confinement problem. *Communications of the ACM*, pages 613–615, 1973.
- [126] S.B. Lipner. A comment on the confinement problem. *Operating Systems Review*, pages 192–196, 1975.
- [127] M. Schaefer, B. Gold, R. Linde, and J. Scheid. Program confinement in kvm/370. *Proceedings of the 1977 Annual ACM Conference*, pages 404–410, 1977.
- [128] S. Cabuk. Network covert channels: Design, analysis, detection, and elimination. *CERIAS technical report*, 2006.

- [129] S. Gianvecchio and H. Wang. An entropy-based approach to detecting covert timing channels. *IEEE Transactions on Dependable and Secure Computing*, pages 785–797, 2011.
- [130] T. Handel and T.S. Maxwell. Hiding data in the osi network model. *Proceedings of the First International Workshop on Information Hiding*, pages 23–38, 1996.
- [131] D. Kundur and K. Ahsan. Practical internet steganography: Data hiding in ip. *Proceedings Texas Workshop Security of Information Systems*, 2003.
- [132] W. Mazurczyk and K. Szczypiorski. Steganography of voip streams. *On the Move to Meaningful Internet Systems: OTM 2008 - LNCS*, 5332:1001–1018, 2008.
- [133] N.B. Lucena, G. Lewandowski, and S.J. Chapin. Covert channels in ipv6. *Privacy Enhancing Technologies - LNCS*, 3856:147–166, 2006.
- [134] Giffin J., Greenstadt R., Litwack P., and Tibbetts R. Covert messaging through tcp timestamps. *Proceeding PET'02 Proceedings of the 2nd international conference on Privacy enhancing technologies*, pages 194–208, 2003.
- [135] M.A. Padlipsky, D.V. Snow, and P.A. Karger. Limitations of end-to-end encryption in secure computer networks. *Technical report, ESD-TR-78-158*, 1978.
- [136] C.G. Girling. Covert channels in lan's. *IEEE Transactions Software Engineering*, pages 292–296, 1987.
- [137] N. Feamster, M. Balazinska, G. Harfst, H. Balakrishnan, and D. Karger. Infranet: Circumventing web censorship and surveillance. *Proceedings of the 11th USENIX Security Symposium*, 2002.
- [138] M. Wolf. Covert channels in lan protocols. *Lecture Notes in Computer Science*, 396:89–101, 1989.
- [139] K. Szczypiorski and W. Mazurczyk. Steganography in ieee 802.11 ofdm symbols. *Security Communications Networks*, 2011.
- [140] L. Butti. Raw covert, 2002. URL http://rfakeap.tuxfamily.org/#Raw_Covert.
- [141] C. Kraetzer, J. Dittmann, A. Lang, and T. Kuehne. Wlan steganography: a first practical review. *8th ACM Multimedia and Security Workshop*, 2006.
- [142] I. Nussbaum, P. Neyron, and O. Richard. On robust covert channels inside dns. *Emerging Challenges for Security, Privacy and Trust, IFIP Advances in Information and Communication Technology*, 297:51–62, 2009.

- [143] E. Cauich, R. G. Gardenas, and R. Watanabe. Data hiding in identification and offset ip fields. *Proceeding of 5th International Symposium*, 2005.
- [144] E. Jones, O. LeMoigne, and Robert J. Ip traceback solutions based on time to live covert channel. *Proceedings of 12th IEEE International Conference on Networks (ICON)*, pages 451–457, 2004.
- [145] H. Qu, P. Su, and Feng D. A typical noisy covert channel in the ip protocol. *38th Annual International Carnahan Conference on Security Technology*, pages 189–192, 2004.
- [146] D. Saha, A. Dutta, D. Grunwald, and D. Sicker. Secret agent radio: Covert communication through dirty constellations. *Information Hiding - LNCS*, 7692: 160–175, 2013.
- [147] C. H. Rowland. Covert channels in the tcp/ip protocol suite. *First Monday, Peer Reviewed Journal on the Internet*, 1997.
- [148] I. Zelenchuk. Skeeve-icmp bounce tunnel, 2010. URL http://gray-world.net/poc_skeeve.shtml.
- [149] G. Danezis. Covert communications despite traffic data retention. *Technical report ESAT*, 2005.
- [150] M. Bauer. New covert channels in http: Adding unwitting web browsers to anonymity sets. *Proceedings of Workshop On Privacy Electronic Society*, pages 72–78, 2003.
- [151] M.A. Padlipsky, D.V. Snow, and P.A. Karger. Limitations of end-to-end encryption in secure computer networks. *Technical report*, 1978.
- [152] S. Cabuk, C.E. Brodley, and C. Shields. Ip covert timing channels: Design and detection. *Proceedings of 11th ACM Conference on Computer and Communications Security (CCS)*, pages 178–187, 2004.
- [153] V. Berk, a. Giani, and G. Cybenko. Detection of covert channel encoding in network packet delays. *Technical Report TR2005-536*, 2005.
- [154] G. Shah, A. Molina, and M. Blaze. Keyboards and covert channels. *Proceedings of USENIX Security Symposium*, 2006.
- [155] Y. Liu, D. Ghosal, F. Armknecht, A. Sadeghi, S. Schulz, and S. Katzenbeisser. Hide and seek in time robust covert timing channels. *Proceedings of 14th European Symposium on Research in Computer Security*, 2009.

- [156] H. Esser and Freiling. F. Kapazitaetsmessung eines verdeckten zeitkanals ber http. *Technical report, TR-2005-10*, 2005.
- [157] X. Zou, Q. Li, S. Sun, and X. Ni. The research on information hiding based on command sequence of ftp protocol. *Proceedings of 9th International Conference on Knowledge-Based Intelligent Information and Engineering Systems*, pages 1079–1085, 2009.
- [158] S. D. Servetto and M. Vetterl. Communication using phantoms: Covert channels in the internet. *Proceedings of IEEE International Symposium on Information Theory (ISIT)*, 2001.
- [159] W. Mazurczyk, M. Smolarczyk, and K. Szczypiorski. Hiding information in re-transmissions. *Cryptography and Security*, 2009.
- [160] A. El-Atawy and E. Al-Shaer. Building covert channels over the packet reordering phenomenon. In *INFOCOM 2009, IEEE*, pages 2186–2194, April 2009. doi: 10.1109/INFOCOM.2009.5062143.
- [161] S. Bhadra, S. Shakkottai, and S. Vishwanath. Covert communication over slotted aloha systems. *Proceedings of the 42nd Allerton Conference on Communication, Control, and Computing*, 2004.
- [162] T. M. Dogu and A. Ephremides. Covert information transmission through the use of standard collision resolution algorithms. *Covert Information Transmission through the Use of Standard Collision Resolution Algorithms*, pages 419–433, 1999.
- [163] A. Hintz. Covert channels in tcp and ip headers, 2003. URL www.defcon.org/images/defcon-10/dc-10-presentations/dc10-hintz-covert.ppt.
- [164] S. J. Murdoch. Hot or not: Revealing hidden services by their clock skew. *Proceedings of 13th ACM Conference on Computer and Communications Security (CCS)*, pages 27–36, 2006.
- [165] S. Gianvecchio, H. Wang, D. Wijesekera, and S. Jajodia. *Recent Advances in Intrusion Detection: 11th International Symposium, RAID 2008, Cambridge, MA, USA, September 15-17, 2008. Proceedings*, chapter Model-Based Covert Timing Channels: Automated Modeling and Evasion, pages 211–230. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-87403-4. doi: 10.1007/978-3-540-87403-4_12.
- [166] S. Z. Goher, B. Javed, and N. A. Saqib. Covert channel detection: A survey based analysis. In *High Capacity Optical Networks and Enabling Technologies*

- (HONET), *2012 9th International Conference on*, pages 057–065, Dec 2012. doi: 10.1109/HONET.2012.6421435.
- [167] E. Tumoian and M. Anikeev. Network based detection of passive covert channels in tcp/ip. In *Local Computer Networks, 2005. 30th Anniversary. The IEEE Conference on*, pages 802–809, Nov 2005. doi: 10.1109/LCN.2005.92.
- [168] J. Zhai, G. Liu, and Y. Dai. A covert channel detection algorithm based on tcp markov model. In *Multimedia Information Networking and Security (MINES), 2010 International Conference on*, pages 893–897, Nov 2010. doi: 10.1109/MINES.2010.190.
- [169] P. Peng, P. Ning, and D. S. Reeves. On the secrecy of timing-based active watermarking trace-back techniques. In *Security and Privacy, 2006 IEEE Symposium on*, pages 349–364, May 2006. doi: 10.1109/SP.2006.28.
- [170] L. Arshadi and A. H. Jahangir. On the tcp flow inter-arrival times dsitribution. In *Computer Modeling and Simulation (EMS), 2011 Fifth UKSim European Symposium on*, pages 360–365, Nov 2011. doi: 10.1109/EMS.2011.34.
- [171] K. Kothari and M. Wright. Mimic: An active covert channel that evades regularity-based detection. *Computer Networks*, 57(3):647 – 657, 2013. ISSN 1389-1286. doi: <http://dx.doi.org/10.1016/j.comnet.2012.10.008>.
- [172] X. Luo, W. W. E. Chan, and K. C. R. Chang. Cloak: A ten-fold way for reliable covert communications. In J. Biskup and J. Lpez, editors, *12th European Symposium On Research In Computer Security, Dresden, Germany, September 24 26, 2007. Proceedings*, pages 283–298, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-74834-2. doi: 10.1007/978-3-540-74835-9_19.
- [173] S. Mou, Z. Zhao, S. Jiang, Z. Wu, and J. Zhu. Feature extraction and classification algorithm for detecting complex covert timing channel. *Computers and Security*, 31(1):70 – 82, 2012. ISSN 0167-4048. doi: <http://dx.doi.org/10.1016/j.cose.2011.11.001>. URL <http://www.sciencedirect.com/science/article/pii/S0167404811001349>.
- [174] A. Feldmann. Characteristics of tcp connection arrivals. In *Self-Similar Network Traffic and Performance Evaluation*. Wiley, 1998.
- [175] R. S. Prasad and C. Dovrolis. Beyond the model of persistent tcp flows: Open-loop vs closed-loop arrivals of non-persistent flows. In *Simulation Symposium, 2008. ANSS 2008. 41st Annual*, pages 121–130, April 2008. doi: 10.1109/ANSS-41.2008.16.

- [176] H. Wu, M. Zhou, and J. Gong. Investigation on the ip flow inter-arrival time in large-scale network. In *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*, pages 1925–1928, Sept 2007. doi: 10.1109/WICOM.2007.482.
- [177] I. W. C. Lee and A. O. Fapojuwo. Analysis and modeling of a campus wireless network tcp/ip traffic. *Computer Networks*, 53(15):2674–2687, 2009.
- [178] N. Kiyavash, F. Koushanfar, Coleman T. P., and M. Rodrigues. A timing channel spyware for the csma/ca protocol. *IEEE Transactions on Information Forensics and Security*, 8(3):477–487, March 2013. ISSN 1556-6013. doi: 10.1109/TIFS.2013.2238930.
- [179] N.L. Johnson, S. Kotz, and N. Balakrishnan. *Continuous univariate distributions*. Number v. 2 in Wiley series in probability and mathematical statistics: Applied probability and statistics. Wiley & Sons, 1995. ISBN 9780471584940. URL <https://books.google.de/books?id=0QzvAAAAAAAJ>.
- [180] F. Benedetto, G. Giunta, E. Guzzon, and M. Renfors. Effective monitoring of freeloading user in the presence of active user in cognitive radio networks. *IEEE Transactions on Vehicular Technology*, 63(5):2443–2450, Jun 2014. ISSN 0018-9545. doi: 10.1109/TVT.2013.2290035.
- [181] R. Archibald and D. Ghosal. A comparative analysis of detection metrics for covert timing channels. *Comput. Secur.*, 45:284–292, September 2014. ISSN 0167-4048. doi: 10.1016/j.cose.2014.03.007. URL <http://dx.doi.org/10.1016/j.cose.2014.03.007>.
- [182] A. Chen, W. B. Moore, H. Xiao, A. Haeberlen, L. T. X. Phan, M. Sherr, and W. Zhou. Detecting covert timing channels with time-deterministic replay. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pages 541–554, 2014.
- [183] 42ne, 2015. URL www.42ne.it/content/covert-channels-implementations.
- [184] A. Liguori. Open-source covert timing channel, 2015. URL <https://code.google.com/p/osctc>.
- [185] ASF. Apache license, version 2.0, 2004. URL www.apache.org/licenses/.
- [186] S. Gianvecchio and H. Wang. Detecting covert timing channels: an entropy-based approach. *Proceedings of the 14th ACM conference on Computer and communications security*, pages 307–316, 2007.

- [187] R. W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, pages 147–160, 1950.
- [188] V.I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 1966.
- [189] A. Tedeschi, A. Liguori, and F. Benedetto. Information Security and Threats in Mobile Appliances. *Recent Patents on Computer Science*, 7(1):3–11, June 2014.
- [190] A. Liguori. From Multilevel Security to MILS: the Evolution illustrated through a Novel Cross-Domain Architecture. *International Journal of Mobile Network Design and Innovation*, FORTHCOMING.
- [191] A. Simonetta, M. C. Paoletti, and A. Liguori. *Testing di oggetti matematici in java. Introduzione a JUnit*. UNIVERSITALIA, 1 edition, 9 2013. ISBN 978-8865075531.
- [192] National Security Agency Central Security Service. Cryptography today, 2015. URL www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml.
- [193] A. Papoulis. *Probability, random variables, and stochastic processes*. McGraw-Hill series in electrical engineering. McGraw-Hill, 1991. ISBN 0-07-100870-5.