

CONTENT ENCAPSULATION TECHNOLOGY FOR DIGITAL LIBRARY SECURITY

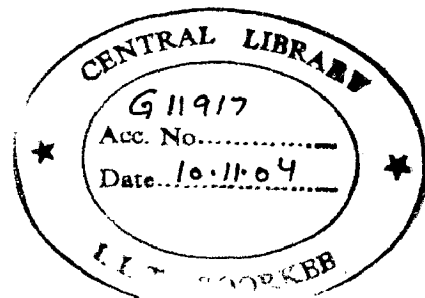
A DISSERTATION

*Submitted in partial fulfillment of the
requirements for the award of the degree*

of
MASTER OF TECHNOLOGY
in
INFORMATION TECHNOLOGY

By

DHRUBA PROKASH PANDEY



**IIT ROORKEE - CDAC NOIDA-
C-56/1, "ANUSANDHAN BHAWAN"
SECTOR-62, NOIDA-201307
JUNE, 2004**

रक्षा वैज्ञानिक सूचना तथा प्रलेखन केन्द्र (डेसिडॉक)
भारत सरकार, रक्षा मंत्रालय
रक्षा अनुसंधान तथा विकास संगठन
मेटकाल्फ हाउस, दिल्ली - 110 054



DESIDOC

**Defence Scientific Information &
Documentation Centre (DESIDOC)**

Govt. of India, Ministry of Defence
Defence Research and
Development Organisation
Metcalfe House, Delhi - 110 054


No: DESIDOC/7100/NSD/2004


Dated: June 22, 2004

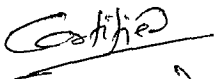
CERTIFICATE

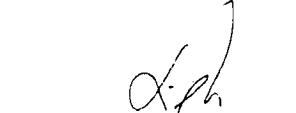
This is to certify that **Mr. Dhruva Prokash Pandey**, a student of M.Tech (IT), IInd year from *Indian Institute of Technology, Roorkee*, has undergone training in this organization from June 02, 2003 to June 21, 2004. During the tenure he has successfully developed a research project titled "**Content Encapsulation Technology for Digital Library Security**" using Visual C++ and Java based on MD5, DSA and AES algorithms and findings of various research papers.

He has worked in all the phases of the project life cycle and shown exemplary sincerity and intelligence during the tenure. He has the inquisitiveness to explore new technologies and works well in a team even under stressful project conditions.


(Ashok Kumar)
Jt. Director


(Sumit Goswami) / 22/6/04
Scientist




(Dr. Mohinder Singh)
Director

Course Coordinator
ER&DCI IIT (E. Campus)
C-56/1, Sector - 02,
NOIDA- 201301

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this dissertation titled "**CONTENT ENCAPSULATION TECHNOLOGY FOR DIGITAL LIBRARY SECURITY**", in partial fulfillment of the requirements for the award of the degree of **Master of Technology in Information Technology**, submitted in **IIT, Roorkee – CDAC, Noida**, is an authentic record of my own work carried out during the period from June 2003 to June 2004 under the guidance of Mr. **Sumit Goswami**, Scientist-C, DESIDOC, DRDO, Delhi.

The matter embodied in this dissertation has not been submitted by me for award of any other degree or diploma.

Date: 21.06.2004

Place: Noida

Dhruba Prokash Pandey
(Dhruba Prokash Pandey)

CERTIFICATE

This is to certify that the above statement made by candidate is correct to the best of my knowledge and belief.

Date: 21.06.2004

Place: Delhi

Supervisor:

Sumit Goswami

(Mr. Sumit Goswami)
Scientist-C, DESIDOC,
DRDO, Delhi.

Co-Supervisor:

V. N. Shukla

(Mr. V. N. Shukla)
Director (Spl. Appl.)
CDAC, Noida


TO WHOMSOEVER IT MAY CONCERN

This is to certify that Mr. Dhruva Prokash Pandey, student of M.Tech. (IT) IITR-CDAC, Noida worked on his dissertation titled “**Content Encapsulation Technology for Digital Library Security**”, under my supervision from June 2003 to June 2004. He was regular and sincere in his work during the said duration.

Date: 21.06.2004

Place: Delhi

Supervisor:



(Mr. Sumit Goswami)
Scientist-C, DESIDOC,
DRDO, Delhi.

ACKNOWLEDGEMENT

The work presented in this report would not have been completed without the guidance and support of many people.

In the first place, I would like to thank **Prof. Prem Vrat**, Director, IIT Roorkee and **Sh. R.K.Verma**, Executive Director, CDAC, Noida.

I would also like to thank **Prof. A.K.Awasthi**, Dean PGS&R, IIT Roorkee. I am also thankful to **Prof. R.P.Agrawal**, Course Coordinator, M-Tech (IT), IIT Roorkee and **Mr. V.N.Shukla**, Course Coordinator, M-Tech (IT), CDAC, Noida.

In particular, I am extremely grateful to **Dr. Mohinder Singh**, Scientist-G, Director, DESIDOC, DRDO, Delhi for providing me the opportunity to undertake this Dissertation work at this organization.

I express my sincere thanks to **Mr. Ashok Kumar**, Scientist-E, DESIDOC, DRDO, Delhi for his valuable suggestions which he gave to us at the time of joining.

I take this opportunity to express my sincere and profound gratitude to my guide **Mr. Sumit Goswami**, Scientist-C, DESIDOC, DRDO, Delhi, for his constructive guidance, active interest and constant encouragement without which this Dissertation work could never have been possible. It was my privilege to work under him.

I would also like to express my indebtedness to **Mr. Munish Kumar**, for his valuable guidance. Without his help and active support this Dissertation report could not have been completed in time.

Dhruba Prokash Pandey
(DHRUBA PROKASH PANDEY)

Enrolment No- 029003

CONTENTS

CANDIDATE'S DECLARATION	(i)
CERTIFICATE	(ii)
ACKNOWLEDGEMENT	(iii)
CONTENTS	(iv)
ABSTRACT	1
1. INTRODUCTION	3
1.1 Objective	3
1.2 Scope	3
1.3 Overview	3
1.4 About Defence Research and Development Organization	4
1.5 Organization of Dissertation	18
2. LITERATURE SURVEY	19
2.1 Digital Library	19
2.2 Preface	19
2.3 Points to Consider Before Creating a Digital Library	23
2.4 Principles of Digital Library	30
2.5 Typical Digital Library Architecture	31
3. CASE OF DIGITAL LIBRARY SYSTEM	35
3.1 Motivation and Background	35
3.2 Safeguarding Digital Library Contents and Users	35
3.3 Structure of Library and Document Protection	36
3.4 Intellectual Property Protection Systems	38
3.5 Digital Watermarking	39
3.6 Super Distribution	54
3.7 Digital Signatures	66
3.8 JAVA	78
3.9 LINUX	78
4. DESIGN AND IMPLEMENTATION	81
4.1 Security for Digital Library	81
4.2 Document Protection	82
4.3 Implementation	90
4.4 Classes and Methods used in the Implementation	119
5. RESULTS AND DISCUSSIONS	125
6. CONCLUSIONS	131
REFERENCES	133

ABSTRACT

This dissertation work gives an account of Content Encapsulation Technology for Digital Library Security, conceived and designed for Defence Research and Development Organization. Digital libraries provide mechanism for the distribution of electronic documents and the terms and conditions of their usage. The tasks of a digital library are not only limited by establishing a secure communication channel, but also extended to the protection of electronic documents even after the transfer. The solutions proposed in this area are based on the use of cryptographically secured containers.

The subject Content Encapsulation Technology for Digital Library Security has been developed as an application software and is designed to achieve secure communication of classified as well as non-classified documents in an encrypted form.

This dissertation work is aimed at protecting documents by wrapping them in a secure cryptographic envelope. The original text message is double encrypted with the help of Hash Algorithms and Digital Signature Techniques. Further, the GUI enables the end-users to encrypt and decrypt the text-messages irrespective of location.

INTRODUCTION

1.1 Objective

The key to the evolution of the library for the information age is not to introduce technology on the basis of technological criteria alone. It is necessary to consider the goals of the library and its users and then assess the appropriateness and utility of a given technology for achieving those goals. So, there is a need to develop a digital library system, which will assure convenient security and data quality with reference to safeguarding digital library contents.

1.2 Scope

There are a number of research publications being published every year by different organizations. Currently, end-users can make a request for any of the publications and can get them manually. But since digitization of libraries is in process, it is necessary that end-users should have access to these publications on their personal computers. Communication channel security issues can be addressed, as there are a number of networking protocols and firewalls installed for securing the channel. What is utmost needed is **Document Protection**, which requires the document to be wrapped in a secure cryptographic envelope coupled with mechanisms for allowing them to be super distributed on variety of external media, such as internet, CD-ROM, satellite, digital cable-television and likewise. We focus on the mechanisms necessary to put security architecture for the Digital Library (DL) in place. This includes protection of the content, feasibility of payment and assertion of copy-and-usage rights.

The scope of this dissertation work will not only be limited to a particular library, but it will prove beneficial to all the libraries. Because unless its usability cannot be realized everywhere, the very existence of such a concept will be crippled.

1.3 Overview

We are focusing on the mechanisms necessary to put security architecture for digital libraries in place. This includes protection of the content, feasibility of payment and assertion of copy-and-usage rights. While current research in secure Web Technology

(like SSH, HTTP or SSL) focuses on the protection of the communication channel, proposals for protecting digital content usually rely on some sort of secure container to realize the functions needed.

The acceptance of the Internet as an infrastructure for electronic commerce, over the last years, triggered the development of several extensions of its underlying protocols in order to integrate or enhance security mechanisms. The new requirements resulting from electronic commerce applications together with a strong commercial interest in secure Internet technology lead to the development of new concepts and the adaptation of existing ideas in order to facilitate secure electronic commerce. All of these mechanisms such as IPSP, SSL, SSH or S/MIME make use of cryptographic operations in order to realize specific security services.

Some previous work which has been done earlier, indicate how a specific mechanism can reduce risks. The mechanism choices are mostly determined when the risk arises in the delivery system. The choice of specific mechanism from a class is sensitive to the value of data at risk and to user preferences.

Several possibilities for communication security systems and their usability for digital library environments have been considered. The common security systems, which attempt to establish a secure communication channel between the communicating parties, are useful to protect less complex, point-to-point transactions, but have several weaknesses for digital libraries. On the other hand, secure container technology seems well suited to digital libraries.

1.4 About Defence Research and Development Organization [1]

The government of Independent India set up the Defence Science Organization in 1948 to advise and assist the Defence Services on scientific problems and to undertake research in areas related to defence. The Defence Research & Development Organization (DRDO) was set up in 1958, by merging the units of Defence Science Organization with the then existing Technical Development Establishments of the three Services. Subsequently, a separate Department of Defence R&D was formed in 1980, to improve administrative efficiency.

The mission of the Department is to attain technological self-reliance in defence systems and weapons. To accomplish this, the Department has the mandate to design, develop and

lead on to production of the state-of-the-art weapon systems, platforms, sensors and allied equipment to meet the requirements of the Armed Forces and to provide support in areas of military sciences to improve combat effectiveness of the troops.

The Department of Defence R&D executes various R&D programmes and projects through a network of 49 laboratories/establishments of the DRDO located all over India and a Centre for Military Airworthiness and Certification (CEMILAC). It also administers the Aeronautical Development Agency (ADA), a society funded by the Department, which is engaged in design and development of the Light Combat Aircraft (LCA). These laboratories and establishments execute programmes and projects in diverse fields of aeronautics, armaments, missiles, combat vehicles, electronics and instrumentation, advanced computing and networking, engineering systems, agriculture and life sciences, advanced materials and composites and Naval R&D. They also conduct specialized training programmes in these areas. The programmes are carried out by a workforce of about 30,000 including more than 6,000 scientists and engineers, supported by a budget of the order of Rs. 30,000 million.

To fulfill its objectives DRDO has a strong partnership with about 40 academic institutions, 15 national S&T agencies, 50 PSUs and 250 private sector enterprises. This has enabled the organization to minimize the effects of sanctions and technology denials, imposed by technologically advanced countries from time to time.

During its first decade, between 1948 and 1957, DRDO was mainly engaged in activities related to clothing, ballistics, operations research, and general stores. During the next decade 1958-68, many products, including small and medium weapon systems, explosives, communication systems and cipher machines were developed. The important achievements of the next decade (1969-79), during which DRDO addressed major hardware systems included field guns, sonar systems, radar and communication equipment and aeronautical systems. Between the years 1980-90, it embarked on programmes of a multi-disciplinary nature for the development of complex and sophisticated weapon systems having latest technology. The contribution of DRDO towards self-reliance in defence systems became evident with the development of flight simulators for *Ajeet* and *Kiran* aircraft, air launched missile target Fluffy, and low-level surveillance radar *Indra*, electronic warfare (EW) systems and sonars.

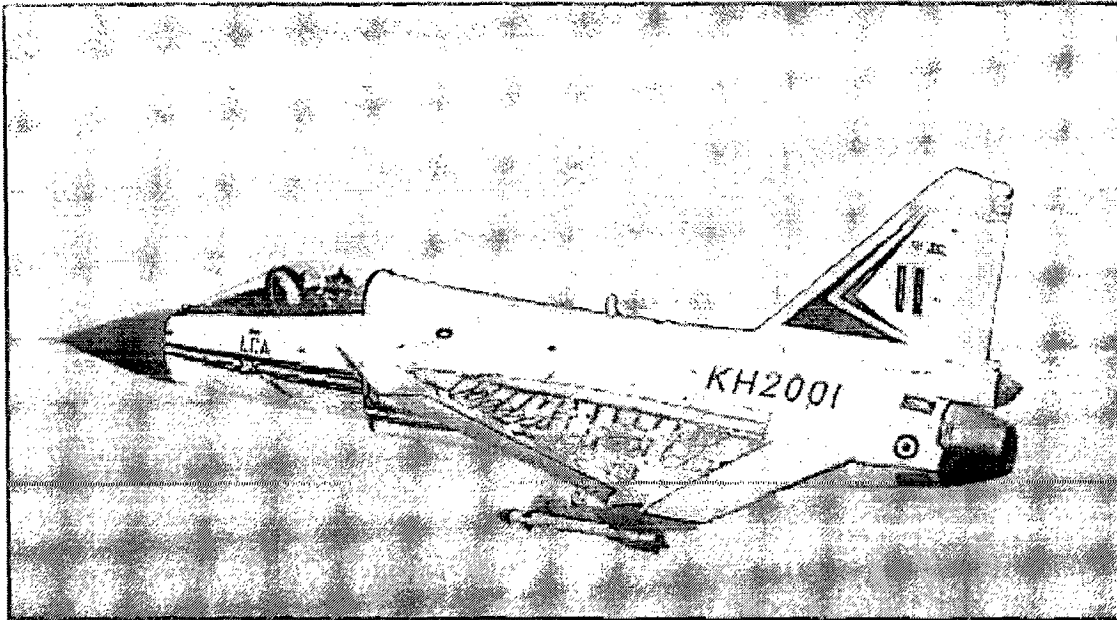


Fig 1.1: Light Combat Aircraft (LCA) in flight

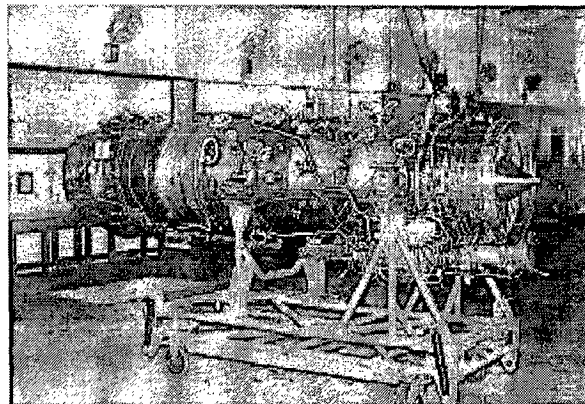


Fig 1.2: Kaveri engine for LCA

During the decade of 1990-2000, certain major programmes undertaken during the previous decade culminated in weapons and systems, like the Ballistic Tank (MBT) *Arjun*, missiles *Prithvi* and *Agni*, pilotless target aircraft *Lakshya*; combat improved T-72 tank *Ajeya*, bridge layer tank on T-72; *Sarvatra* bridging system, artillery combat command and control system, 5.56 mm INSAS rifle; light machine gun and ammunition; the super computer PACE+, sonar systems and Naval mines.

Starting out as an agency which carried out science-based technical improvements to existing systems, DRDO has grown today to a high- technology agency capable of undertaking *ab-initio* design, development and integration, leading to production of world class weapon systems meeting Qualitative Requirements of the Services. DRDO has achieved technological self-reliance in ammunition, armoured systems, surface-to-surface missiles, sonar systems, and Electronic Warfare (EW) systems and advanced computing.

Achievements and Programmes

Aeronautical Systems- DRDO has already delivered pilotless target aircraft *Lakshya*, aircraft arrester barrier, a variety of brake parachutes and balloon barrage system to the Armed Forces. The Light Combat Aircraft (LCA) programme, under execution at Aeronautical Development Agency (ADA), has led to the development of several state-of-the-art aeronautical technologies and the creation of a necessary infrastructure, despite the constraint of sanctions imposed by the advanced countries and the country's industrial base unprepared for the requisite components and advanced materials. The first LCA Technology Demonstrator (TDI) has undergone a number of successful test flights. The remotely piloted vehicle *Nishant*, is at an advanced stage of evaluation. Certain crucial elements, of the modernized avionics of Su-30 MKI aircraft being acquired by the IAF, have been supplied and successfully integrated.

Armaments- DRDO has achieved a high degree of developmental self-reliance in the area of armament and ammunition. More than 300 ammunition items based upon DRDO technology worth Rs. 50,000 million have been manufactured by ordnance factories. These include 5.56 mm calibre rifle and machine gun, anti-tank ammunition, illuminating ammunition, mines and a variety of bombs for the Air Force.

Missile Systems- DRDO has established core competence in the area of surface-to-surface missiles, which has been demonstrated through development of *Prithvi* missile and its variants, demonstration of re-entry and related technologies for *Agni-I* and development of the longer-range version, *Agni-II*. The surface-to-air missiles *Trishul* and *Akash* and anti-tank missile *Nag* are at an advanced stage of flight evaluation. For the first time in the world, the indigenously developed capability to hit a target at 4.18 km in top attack and the fire-and-forget mode has been demonstrated through a flight test of *Nag* missile.

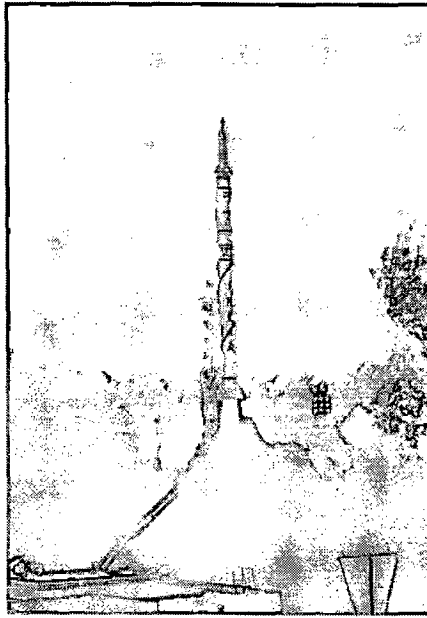


Fig 1.3: Agni Missile

Radar and Communication Systems- In spite of the non-availability of indigenous microelectronic devices and components, DRDO laboratories have successfully developed and delivered a variety of systems falling under this group including INDRA PC radar, equipment for Army Radio Engineered Network (AREN), very low frequency receivers, satellite communication terminals and secure telephones.

Electronic Warfare (EW) Systems- DRDO developed a number of EW systems with considerable success. These include *Ajanta*, *Coin*, *Vikram* and Radar Warning Receiver (RWR) for MiG-23 and MiG-27 aircraft, which have been delivered to the Services. In addition, the self-protection jammer for MiG-27 is ready for delivery. Development of an advanced RWR for MiG-21 aircraft has been completed.

Combat Engineering Systems- DRDO's efforts have led to successful development of a variety of complex multi-disciplinary systems including bridge layer tanks, mat fording vehicles, mine field marking equipment, mortar carrier vehicles, armoured engineering recce vehicle, armoured amphibious dozer, operation theatre complex on wheels, ward container and mobile water purification systems. The R&D expertise in DRDO and the production infrastructure in the country can now be brought together for world-class engineering systems for Defence Services.

Main Battle Tank- *Arjun*, and its derivative systems have met stringent requirements of the Army successfully. This tank is contemporary to world-class tanks like M1A2 of the

USA and Leopard 2 of Germany. The bulk production of MBT *Arjun* is now at an advanced stage.

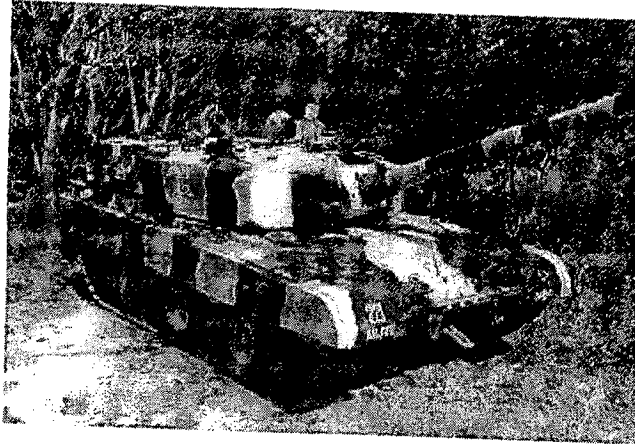


Fig 1.4: Main Battle Tank (Arjun)

Based on the experience gained during the development of MBT *Arjun*, DRDO has successfully integrated a 155 mm SP turret on *Arjun* derivative chassis for development of a 155 mm self-propelled weapon system. It has also modernized the T-72 M1 tank to improve its fire power, mobility and protection.

Underwater Warfare Systems- This has been another area in which a solid foundation for self-reliance has been established by successful development and delivery of a number of sonar systems, including *Simhika*, *Humsa*, *Humvad* and *Panchendriya* and a number of Naval systems including triple tube torpedo launcher and Processor Based Ground Mine and Processor Based Moored Mine. The systems in advanced stages of development include *Mihir* and *Nagan* sonars, advanced experimental torpedo *Shyena* and also wire-guided torpedo.

Other Major Achievements

Advanced Computing and Software Products- DRDO has successfully developed Supercomputer PACE+, consequent to the denial of such a computer by the advanced countries. DRDO's expertise in software has been demonstrated through the development and commissioning of war games *Shatranj* and *Sangram* for the Army; *Sagar* for the Navy and air war game software for the Air Force. A landmark toward self-reliance in microprocessor technology has been achieved through development of ANUCO, a floating-point coprocessor and a 32-bit RISC processor ANUPAMA. Its

processing speed is being further enhanced from 33 MHz to 350 MHz. In addition, a three-dimensional medical imaging system ANAMICA has been developed. Softwares called GITA (Graphical Interactive Three Dimensional Applications, a general purpose CAD software and AUTOLAY, a design for software manufacture is being marketed internationally. DRDO has also set up a Very Large Scale Integrated Circuits (VLSI) design facility, which has been used for developing a number of Application Specific Integrated Circuits (ASICs) like the digital signal processing chip.

Critical Electronic Components- Initiatives to achieve self-reliance in the field of electronic components has been taken by setting up facilities for production of Gallium Arsenide and Silicon devices. Under the programme, CODE, several types of components have been indigenized, like integration components, microwave components, millimeter wave components and other special types of components required for various ongoing DRDO programmes. A facility has been created to lead to fabrication of Gallium Arsenide wafers and Monolithic Microwave Integrated Circuits (MMICs) in 1-18 GHz range. Under a co-operative venture with other S&T Departments and Industry, DRDO has contributed in setting up a silicon foundry, which has the potential of making the country independent of foreign sources in respect of most of the VLSI requirements.

Electronic and Strategic Materials- DRDO has developed several types of strategic materials like Jackal M-1 steel for bullet-proof jackets and bullet-proof vehicles; aluminium alloy for structural applications in the Light Combat Aircraft; single crystal super alloy and directionally solidified super alloy for use in high performance aero-engines; fibre reinforced plastic (FRP) composites for immunity against small arms ammunition and missile fragments on board ships; kevlar/aramid composite material for light weight combat helmet and rare earth based high energy magnets for application in India's space programme. DRDO has undertaken certain initiatives for making the country self-sufficient in a number of strategic materials, like setting up a facility for carbon fibre and prepegs for application in aerospace structures; launching of a national programme for development of smart materials and technology development for high purity alumina substrate and PTFE soft substrate for use in microwave integrated circuits. Technology for Fullerenes and carbon nano tubes which have potential applications in

stealth, smart materials and micro-electronics have been indigenized and facilities for nano tubes at 5 gm/day level has been established.

Metal/Material Processing Technologies- The technology to convert titanium tetrachloride into titanium sponge, which is a closely guarded secret of the few titanium sponge producers in the world, has been developed which will enable India to utilize the world's largest reserves of titanium which the country has. This can be gainfully utilized in defence, aerospace, oil and power sector industries. In addition, innovative processes comprising air induction melting and electro-slag refining have been developed to produce iron aluminide based advanced inter-metallics. Aluminium based particulate metal matrix structural composites for aerospace applications have also also been developed. Technologies and processes such as ion plasma deposition of protective layers and laser processing have been established. Two grades of ultra clean structural and armour steels of High Strength Low Alloy (HSLA) steel variety, copper-boron (CuBux) and armour (ARx) have been designed and developed for structural and armour applications in marine vessels.

Radar and Communication Technologies- To meet the requirements of modern radars, namely longer detection ranges, faster data rates (short reaction times) and ability to accommodate increased target densities, DRDO has indigenously developed the technology area of array design and developed expertise in the development of radiating elements, taking into account the mutual coupling, collimation and beam steering, feeds etc. Aplanar, phased array system has been successfully implemented in *Rajendra* radar. Another achievement is the speech secrecy systems based on state-of-art encryption techniques for telephone secrecy (speech), secrecy over radio and multi-channel (bulk) secrecy over voice and data. The satellite communication terminals, based on state-of-art techniques like spread spectrum multiple access, high-grade secrecy and low bit-rate voice digitization, have been developed. One such terminal in S-band was used during the Orissa cyclone in 2000, for communication with remote villages.

Missile Technologies- During the execution of IGMDP programme, DRDO developed several technologies that have gone into various missile systems. These include: strapdown inertial guidance system, high strength low weight magnesium alloy wings; manoeuvrable trajectory; accurately deliverable high lethality field interchangeable

warheads; multiple target tracking; composite airframe; nitramine based smokeless propellant; ram rocket technology; three beam command guidance system; carbon-carbon technology; and manoeuvrable re-entry guidance and control for long range missions.

Naval Technologies- During the course of the development of indigenous surface, ship and submarine sonars and other sonar systems by DRDO, a number of technologies have been developed. These include multi-channel sonar signal conditioning and data acquisition; sonar signal processing hardware; sonar display systems; sonar simulation and sonar power amplifiers. In the development of underwater acoustic transducers of various types, special acoustic materials like polymers, polymer matrix composites, elastomers and adhesives have been developed along with expertise in engineering aspects like packaging underwater sealings and encapsulations. DRDO is a world leader in development of Impressed Current Cathodic Protection (ICCP) technology to supplement the protection provided by paints to underwater structures against sea water corrosion. Work is in progress on Dual Zone ICCP system. Fire retardant intumescent paint; non-skid and high performance exterior paints and polymer based materials like vibration damping material; ion exchange-cum-indicator resin and polyurethane sealant have been developed. Work is also in progress on fuel cells as an alternative source of power. In the area of underwater weapon propulsion, magnesium-silver battery technology and contra rotating motor with indigenous design and technology have also been developed. Machinery Control Room (MCR) simulators for training engine room crew have been developed. The DRDO-developed hydrophone system was used to detect Gujarat earthquake victims buried under the debris, based on which it was possible to rescue five persons.

National Infrastructure Assets

DRDO has been instrumental in creation of sophisticated and high cost R&D facilities for test, evaluation and other purposes. These may be termed assets, as these fulfill the requirements not only of DRDO but also of other scientific organizations and of the industry. A brief account of such facilities created, is presented.

Range Test Facilities- To meet the requirements of various missiles and other weapon system development programmes, a total of four launch complexes have been established: three at Interim Test Range, Balasore, and one on an island. These launch

complexes suit specific requirements without affecting the natural environment in the test range. The range of instrumentation includes sophisticated radars, electro-optic tracking system, telemetry system, range computer and wide band data acquisition and processing system. With the help of these sophisticated instrumentation, the post-flight data are available within thirty minutes of the flight. In the recent past, the range facility was utilized by Ministry of Defence, Singapore, on a paid basis.

Flight Simulation Facilities- DRDO has created several flight simulation facilities to support design investigations of fighter aircraft performance, handling qualities and capabilities in close combat and mission system performance. Some of the facilities include: research flight simulation facility, pilot-in-loop flight simulation facility, air combat simulator, mission avionics systems simulator, cockpit environment facility and aircraft system maintenance simulator. A virtual reality centre has been set up to address the requirements of virtual prototyping of LCA. The Aeronautical Material Testing Laboratory (AMTL), a national facility, is one of its kind for testing aeronautical material and components. In addition, under Aeronautics Research & Development Board (AR&DB), DRDO helped in setting up of sophisticated test facilities at IISc, Bangalore, IITs, and some universities and at other technological institutions like NAL, to support R&D in aeronautics and applied science. Some of these major facilities are: modified trisonic wind tunnel (NAL), 200 mm hypersonic wind tunnel (IISc), high temperature low cycle fatigue test facility (IIT-B) and full-scale fatigue test facility (NAL).

National Centre for Automotive Testing- DRDO has set up a National Centre for Automotive Testing (NCAT), at Ahmednagar, for testing and evaluation of automotive vehicles, their systems and components for certification for compliance of various national/international standards. Spread over an area of 450 acres, this facility consists of track testing and testing for emission, photometry, EMI and safety and has necessary supporting infrastructure to provide a one-stop solution to the requirements of Indian automotive industry. A variety of test tracks and facilities are spread over an area of 450 acres. The test tracks simulate a variety of ground/road surface conditions, which a vehicle normally encounters during its span of use.

Electronic Warfare Test and Evaluation Facilities- DRDO has created Electronic Systems Evaluation Centre (ELSEC) for ground integration of EW systems and testing of

systems under real life conditions. A Range on Wheels (ROW), comprising six mobile ground stations has been made operational. It is a unique facility for evaluation of airborne EW systems during development, user acceptance and system enhancement phases. The setting up of EW Simulation Testing and Evaluation Station (SITES) and Microwave Components Qualification and Testing Centre (MQTC) is in progress.

Underwater Research Facilities- A premier research facility called High Speed Towing Tank (HSTT) for carrying out studies on experimental hydrodynamics related to model testing of ships, propellers and submerged bodies has been set up. An Underwater Acoustic Research Facility (AURF), a lake facility established by DRDO at Kulamavu in Idukki district of Kerala, carries out calibration and full-scale testing of underwater acoustic transducers, array and other sub-sea equipment like fish finding sonars, echo sounders and underwater communication systems. A dedicated research ship *Sagardhwani* equipped with state-of-the-art laboratories has been developed and is being used for collection of oceanographic data. Materials and Transducers Simulated Test facility (MATS), the only one of its kind in the Asia Pacific region and one of the very few in the world, has been established recently which provides static and dynamic measurements on materials and transducers under different conditions of temperature and pressure, simulating ocean depths. The setting up of an underwater range and a cavitation tunnel facility is in progress.

R&D For Societal Benefits

Floor Reaction Orthosis (FRO)- As a medical spin-off of advanced composite technology used in making missile nose cones, Floor Reaction Orthosis, a walking aid for polio patients with quadriceps muscle weakness, has been developed. This weighs only 300 gm, as against 3 to 3.2 kg for the commonly used variety, is inexpensive and can be worn easily with and without shoes. More than 2,500 such walking aids have been fitted to polio handicapped persons in camps organized for this purpose.

Coronary Stents- Using special grade austenitic stainless steel, developed for LCA and missile programmes, two types of stents have been developed for dilating constricted arteries. Over 115 stents have been fitted in patients so far. The cost of the indigenous stent is Rs. 15,000 as against Rs. 40,000 or more for the imported one. **Cardiac Pacemaker-** An external pacemaker has been designed and developed for intensive care

of patients suffering from degenerative heart diseases. The system has been clinically validated at Nizam Institute of Medical Sciences, Hyderabad. Efforts are being made to convert it into a portable system.

Cardiovascular Catheters- These have been developed to offer a heart patient the option of non-surgical treatment of defect within the heart and the rest of the circulatory system. The cost of the indigenous catheter would be Rs. 1,500 against Rs. 4,500 for the imported one.

Cardiac Stress Test System- A PC-based, low cost indigenous system has been developed to acquire and analyze the ECG of a person doing exercise. The system comprises a standard protocol of graded exercise programme, acquisition, analysis and documentation of ECG and trends in BP, heart rate and ECG, indicating heart abnormalities. The system hardware consists of a 12-channel ECG data acquisition system and is priced at Rs. 350,000 to 400,000 as against Rs. 1.2 to 1.5 million for the imported one. The system is in operation at Air Force Hospital, Delhi, and its technology has been transferred to trade.

Cytoscan- Using the latest pattern recognition and image processing technologies, a computer aided cancer detection device has been developed by DRDO. The system is used for diagnosis and prognosis of several cancers, including cervical and breast cancer. The system has been used for detection of cervical cancer amongst tribal women in Andhra Pradesh under Project *Tulsi*, funded by the Ministry of Social Welfare. The programme will be extended to rural areas of Bihar, Madhya Pradesh, Rajasthan and Uttar Pradesh. More than 20,000 rural women have been scanned so far.

Interaction with Academia:

DRDO has constituted four research boards to nurture and harness talent in academic institutions, universities, R&D centres and industry. The organization provides necessary facilities for promoting basic research and to catalyse cross-fertilization of ideas with R&D agencies in other sectors for expanding and enriching the knowledge base in their respective areas. The boards provide grants-in-aid for collaborative defence-related futuristic front-line research having application in the new world-class systems to be developed by DRDO. The catalytic role played by research boards has helped rapid growth in building capabilities in the area of aeronautical state-of-the-art systems like

light helicopter, and in setting up a centre of excellence in Computational Fluid Dynamics (CFD) at the IISc, Bangalore, which is anticipated to give a boost to the designing of aeronautical systems within the country. Another centre of excellence in aerospace system design and engineering is being set up at IIT, Mumbai. A Centre for Composite Structure Technology is proposed to be set up at the National Aerospace Laboratory, Bangalore. Grants-in-aid by DRDO have led to setting up of a hypermedia digital library in IIT, Kharagpur, to the development of audio-visual training aids for aircrew, to indoctrination in air sickness and positive pressure breathing at the Institute of Aviation Medicine, Bangalore, and to the development of rarefied gas dynamic facility at IIT, Chennai. The Armament Research Board (ARMREB) has approved projects in the fields of high-energy materials, sensors, ballistics and other armament related fields. Under the Naval Research Board (NRB), projects are being pursued in five technology areas. Under Life Sciences Research Board (LSRB) projects have been supported in the areas of biological and bio-medical sciences, psychology, physiology, bio-engineering, specialized high altitude agriculture, food science and technology.

Collaboration with Industry

Eight DRDO laboratories working in the areas of advanced materials, robotics and artificial intelligence, communication systems, life-support systems, corrosion protection, advanced composites and desert technologies have been opened to the industry. Several technologies have been transferred to private industry such as the Scara robot, used for assembly jobs and the articulated robot used for material handling, welding, spray-painting etc. In the field of material science, the technologies transferred include: boropak, a chemical mixture to impart surface hardness and reduce wear and tear of ferrous and some non-ferrous metals; non-spark tools for copper titanium alloy; gigly saw for use by orthopaedic surgeons; rust converter for protection of ferrous metals against corrosion; moisture-resistant corrugated fibre board box as an alternative to timber for packing; and glacier tents for protection in sub-zero temperatures. In a number of areas involving emerging technologies in which industries are not willing to invest setting up defence-specific manufacturing facilities, DRDO has also been collaborating with other departments as well as industry to help transform defence technologies for developing products for the civil sector. As these exercises involve long gestation periods,

technological risk, lack of continuity of orders and lack of economy of scale, DRDO has assisted in setting-up dedicated facilities in such areas. Some of these initiatives are listed as follows:

Heavy Alloy Penetrator Project (HAPP): A fully automated factory for manufacturing high precision components and assembly of Fin Stabilized Armour Piercing Discarding Sabot (FSAPDS) ammunition has been established by DRDO at Trichy and handed over to Ordnance Factory Board. World class FSAPDS ammunition of different calibres are under regular production and have been supplied to the Army.

Bharat Dynamic Limited (BDL): A modern factory has been established for production of *Prithvi* and other missile systems being developed under Integrated Guided Missile Development Programme (IGMDP). Free flow production of *Prithvi* has been established.

Hindustan Aeronautics Limited (HAL): An Aerospace Division has been established to provide special thrust of *Prithvi* and other missile systems being developed under IGMDP.

Non-Ferrous Technology Development Centre (NFTDC): This society has been established as an advanced technology centre with participation of the Department of Mines, DST, DRDO and Industry (BALCO, NALCO, HCL and HZL). Some of the contributions of this centre include millform of copper alloys, silver-based brazing alloys, titanium implants and master alloys for grain refinement of aluminium.

Training Schemes:

DRDO has introduced a number of schemes for training of defence-science personnel in universities and other leading academic institutions. It also has two training institutions namely, Institute of Armament Technology (IAT) at Pune and Institute of Technology Management (ITM) at Mussoorie. These institutes provide specialized training programmes in diverse fields. The ITM has recently started conducting MBA in Technology Management in collaboration with Bhartidasan University. The IAT has been accorded the status of a deemed-to-be university. In addition, a number of laboratories conduct training programmes in disciplines of their core competence like fire-fighting and fire-engineering, war gaming softwares, special clothing and so on. To cite an example, Defence Food Research Laboratory at Mysore conducts training

programmes in food science and technology, modern methods of handling, hygiene, transportation, storage and packaging of food materials; comprehensive course in food microbiology and a ten-month postgraduate programme in food analysis exclusively for Service officers. In recent years, the PG Diploma Course (recognized by the University of Mysore) has trained many civilian candidates who have been absorbed by food industries. Other short term orientation and specialized courses are conducted at the specific request of industries and laboratories, colleges and universities. A DRDO laboratory, the Defence Research & Development Establishment at Gwalior, has been recognized as a centre for training inspectors who are to be appointed by the UN Organization for Prohibition of Chemical Weapons.

1.5 Organization of Dissertation

Chapter 2 gives the Literature Survey, which discusses the literature study of the various concepts involved. Chapter 3 discusses the study of Digital Library System. Chapter 4 describes the Design and Implementation aspects involved in the Dissertation. Chapter 5 shows the Results and Discussion. Chapter 6 discusses the conclusion of the Dissertation.

LITERATURE SURVEY

2.1 Digital Library

The Digital Library System is developed by Defence Scientific Information and Documentation Center, DRDO in the hope of addressing some of the leading issues that this organization is facing in developing digital content and distribute it to its users on wide area network. We are dealing with a plethora of new technologies and issues in the realm of Digital Libraries. The evolution and coalescing of Java applications, digital object standards, internet access, electronic commerce, digital media management models, search engines and library automation systems is causing librarians to rethink many of their traditional goals and modes of operation. Consequently, we felt the need for a comprehensive, state-of-art document that could provide concrete solution in the nascent field of Digital Libraries.

2.2 Preface [2]

A digital library has material stored in a computer system in a form that allows it to be manipulated (for instance, improved retrieval) and delivered (for instance, as a sound file for playing on a computer) in ways that the conventional version of the material cannot be. The digital library can be viewed as an intermediary between information creators and end-users as the library's customers.

Because the material is in digital (or computer readable) form, some new possibilities are opened to the digital library that are not there for the conventional library, even one with the same material. The Digital Library System (DLS) will be an extensive and hospitable IT system enabling the library to meet its strategic and operational objectives in relation to the collection of digital materials and the provision of access to them as follows:

- Managing the storage of all digital materials, which the library acquires.
- Maintaining long-term access.
- Creating and maintaining metadata.
- Enabling reading room users to identify and retrieve the items.
- Interfacing effectively with existing and planned library IT systems.

As an example, the material delivery process can be very different from the removal of a book from a shelf and checking it out. Because the book in the digitized form can be copied to a user's computer for reading, but still remain in the computer stacks, it can immediately be loaned to another user. This implies that holds (reservations) could become a thing of the past for a fully digital library, at the expense of a very much more complex usage tracking system. The processing applies to materials that come originally in non-digital form. For this material to become part of a digital library, the material must be at least digitized, or more usefully, be converted to computer manipulable form.

	Original	Digital Form	Digital Information
Physical Form	Physical object (book, video)	Computer file	Computer file
Format	Varied (English text, VHS)	Graphical file (.bmp, .mpg etc)	Structured file (. doc, .mpg), database and index records
Readability	Human or special equipment	Computer graphics program	Computer text, video or database program
Reproduction	Physically duplicate original (photocopy, duplicate)	Copy file and print any number of exact duplicates	Produce original information in different form (reprint book in large italic type, play video with different sound track)
Manipulation	Physically modify (write in margins, cut and slice tape)	Mark electronically and manipulate graphically (add user specific notes, reduce/enlarge)	Edit the original information, produce derivative work, copy and distribute endlessly

Table 2.1: Analog vs. digital information

2.2.1 Core Capabilities of Digital Library Systems- Digital library systems compose a family of automated systems that together provide a comprehensive capability to manage

the digital content of an enterprise. It is useful to divide the capabilities of digital library systems into the following areas:

- Capture or creation of content.
- Indexing and cataloging (metadata).
- Storage.
- Search and query.
- Asset and property rights protection.
- Retrieval and distribution.

Content exists in multiple sizes, formats, and media, each with accompanying technical challenges. Content may be structured or unstructured. It may have exact, precise meaning; or it may be fundamentally ambiguous. Content may directly or indirectly support a business process or function. Such functions and relationships, when reduced to a particular software and hardware implementation, lead to operational digital library systems.

2.2.2 Digital Libraries and Traditional Libraries- Digital library functions, in so far as they purport to organize information, may be compared with traditional library functions. Digitization is technically the conversion of analog to digital formats. A common human artifact, such as a bound book, loses value when simply scanned into bits. In a library context, where organization, access, protection, and preservation are important business functions, digitization technologies are starting points for a complicated set of computational processes that in the first instance reconstruct the cultural, conventional, and intuitive significance, structure, and external relationships that defined the original artifact. Additionally, digitization and other processes may be able to add value and support certain fiduciary responsibilities that resemble functions of traditional libraries. In a similar way, other core capabilities of traditional libraries can be transposed to the digital domain. Cataloging is transposed to the generation of metadata, and is an area where much work needs to be done to develop automated, multidimensional indexing and cataloging procedures. Just as the public card catalog is a gateway to the holdings of a conventional library, search of content and metadata is the gateway to a digital library.

Circulation in a conventional library transposes to network access, retrieval and delivery. Table 2.2 relates digital library capabilities to well-known capabilities of traditional libraries. Although many of these procedures predate digital libraries, digital library design can benefit from the comparisons.

Digital Library Capability	Traditional Library Capability
Capture	Acquisitions and collection development
Catalog and Index	Cataloging rules and bibliographic control
Store	Stacks, inventory management and shelf lists
Search	Public card catalog
Protect	Patron privileges and circulation rules consistent with public law and policy
Retrieve	Loan management and interlibrary loans

Table 2.2: Comparison of Digital and Traditional Library Capabilities

2.2.3 Functionality- The Digital Library System will enable the library to meet its strategic and operational objectives in relation to the collection of digital materials and the provision of access to them as follows:

- It will provide an extensive and hospitable IT solution to enable the library to store, preserve and provide access to digital published output. It will support the full volume projections for digital publications required for the library's collection.
- It will support the collection of digital materials, licensed as necessary, for the remote document supply services - electronic journals and patents - for the increased benefit of research and industry.
- It will provide a system to hold all other digital materials purchased or otherwise acquired for the collection and the library's services.
- It will enable the library to maintain continuing access to digital materials, regardless of origin, for future generations of users.
- It will support greatly increased access to digital materials within the library's reading rooms.

- For wider public access it will support the digitization of significant parts of the library's unique historical collections.

2.2.4 Benefits- The benefits of the DLS to the library will be to provide an integrated digital environment for the acquisition, storage, access and preservation of the library's rapidly growing digital collection. The DLS will provide the secure IT environment to hold all deposit materials and to provide the means of ensuring long term access to the national digital archive. The library will also add purchased and licensed electronic materials to the store for access in the reading rooms. In addition, the library will use the DLS to store the materials digitized from its historical collections for access purposes. Access to these latter materials will be developed in a number of ways, for instance through the library's website.

2.3 Points to Consider Before Creating a Digital Library

2.3.1 Fundamentals- Digital Library: Conventionally there are two possibilities:

- A library that contains material in digitized form.
- A library that contains digital material.

An automated library is not, per se, a digital library as a library consisting entirely of conventional physical material (such as only printed books) may be very highly automated. This automation does not make it digital in the sense we are considering here. However, it is true that a digital library must be automated in some of its essential functions.

Digital material: In this computerized age, information and the medium on which it is recorded can be considered as either digitized or not. There are many other ways of categorizing the material, but computer readability is the important criterion here. Digital can be taken as a synonym for computer readable. This is a serious generality, but it is this aspect of information that is most relevant to a digital library. The creation of digital information from conventional is generally a two-stage process.

The first stage is digitization. This is essentially the conversion of the physical medium into a digital representation of that physical medium. It takes no account of any information content of the original material, in the sense people would generally recognize the term.

The second stage of the computerization process is to have the computer extract information from the digitized image. For text this is done by Optical Character Recognition (OCR) software that recognizes the shapes of the letters of the alphabet and produces a file exactly the same as one produced by a word processor used to type in the same text. At the digitized image stage it is only possible to perform so called graphical manipulation such as stretching, compressing, turning color to black and white, etc. on the image.

Automatic Indexing: This is the extraction of the information for a bibliographic record (the metadata) directly from the original text by a computer program. It is particularly concerned with the extraction of keywords as an indication of the content of the document. Often it is called free text or full text indexing. They do differ, but an important part of their appeal is the automatic extraction of the indexes from the text.

Its advantage is that there is no human intervention. Thus it can be run continuously and cheaply. The extraction (indexing) process is the same for all documents and thus avoids the idiosyncrasies of individual cataloguers. Authority files (or lists) can be used extensively to further ensure consistency.

Its disadvantage is that there is no human intervention. Thus a document is characterized by the frequency of certain words and these may give a very wrong picture of the actual content. Standardized subject headings are difficult to apply, as they must be matched through statistical means. Here again the chances for deviation are quite large.

2.3.2 Policy- Need for a Digital Library: Does the library have a collection of purely digital material? Is it required that this material be delivered directly to users of computers? Do users need to search in non-traditional ways for the material they need? Which material types in the existing library would benefit from being digitized? How? Is there a need for multiple copy distribution at the same time? Is there a need for the material to be modified and returned to the library? Are there no other sources for searching and retrieving the material that you wish to digitize? Is the material unique or confidential? Are the users geographically and temporally widespread?

Generally the more yes answers, the more a digital library is a sensible proposition.

Library's Information is Valuable: If the library is central then, presumably, the information is important. If the library is used only occasionally, but intensively (as in the

reference section of a library attached to a research laboratory), then it is still valuable. The best benefit may be obtained through improved search tools for internal and external information rather than digitizing everything in sight. If the library is used infrequently or is an archive then the cost of digitizing may still be justified, but on different grounds. The library may have been bypassed as an information dissemination center and digitizing its collection(s) may be a way to re-position the library into a more central role. Information is Changing- If information is changing, then the ongoing costs of re-entering the information into the system will need to be considered. Also important is the policy for handling different versions of documents within the digital library.

An alternative scenario is that the focus of the library (and presumably the organization) is changing. The material may be acquired in digital form or the costs may be justified by one-time processing as the new material is acquired.

Library or organization wants an In-House Digital Library: This is a methodology question. It may be possible to outsource all, or most, of the functions of a digital library. Alternatively, it may be possible to buy access to external search and delivery services that cover most of the library's requirements in the digital area. This can save on initial digitization and on-going administration costs.

However, organizations in general, have the subject expertise of the in-house library staff and there may still be security and service availability issues to consider. Application Service Providers (ASPs) do nothing but provide the library with access to its particular application requirements in an environment that is staffed with experts in the particular application area. Although this is not a common business model in the Library Automation area at present, it is a growing trend and this is a factor that can change many of the economic numbers if a suitable partner ASP can be found.

Coexistence of a Digital Library and a Conventional One: This is really the all or nothing question. Is it intended to replace the existing library with an all-digital material version, or is it intended to supplement what the existing library does with new services? The question here impinges on the perceived role of the library in the organization, its efficiency, how the organization is going to handle internal (and external) communications in the future, and if there are different functions and facilities provided by the physical library and the digital one. Bearing on this are the issues of branch (or

local campus) libraries, specialist knowledge, specialist collections, centralization vs. decentralization, and the other spin-off functions of the library, both physical and digital. Serendipity is the word here, especially in the supply of tangential material, the answering of strange questions, the place to work quietly, the superior analysis tools available on-line, and the benefit to the organization of informal communications.

2.3.3 Audience- Whether demand for new services and/or material exists or not: Is the digital library proposal coming from the library users or is it generated by the library staff or the computer (MIS) department? This is really a question addressing the issue of how the organization decides on the introduction of new services. Is it market driven or does it follow a planned introduction of services or technology?

If management is proposing the digital library, it is important to determine that it will prove beneficial for the organization and its potential users. If the potential user demand extends outside the organization it becomes a marketing and business case to determine which, if any, digital library services and information sources are justified.

Has the market been sized: How much use will be made of the library and its services? Is the (potential) user population large enough to achieve the organization's goals, whether they are cost recovery, better information flow, or even corporate publicity?

There are no absolute numbers for the user population as it depends on the services, their cost and the desired return. However, an estimate of user numbers must be made so the global benefit can be discussed.

Method of use of Digital Library: Will users be offered different (new) services that are not currently available? Will some of the services replace those of the conventional library? What changes in the way users work will be introduced by the advent of the digital library? What changes in the rest of the organization will be needed to accommodate the new method of operation? What will be needed to make best use of the digital library? Is there likely to be an improvement in conventional library services as a result of converting some services and information to digital format?

As an example, the delivery of material directly to the user's desktop can have a profound impact on work patterns. Both the method of working and the nature of the information held by the digital library are candidates for modification. This is generally an iterative process to achieve the best result.

Procedure for accessing Digital Library: Since digital libraries are held within computers it is important to realize the possibilities for access that are offered and are denied. Access can be permitted from the user's desk wherever that may be.

If users do not all have computers then public access must be provided. Is this best done within the physical confines of the library or should/could it be distributed across different buildings? If these library access stations are used then they will need equipment and maintenance, but will be physically close to the users.

Another access model is to assume all access will be from desktop computers. These do not need to be fully functional PC's. Network computers or clustered workstations can be tied to a local server and provide inexpensive access to users who do not have full PC's. This model has the advantage of cheap deployment, easy desktop access and access to a wide variety of applications, possibly held within the digital library itself.

2.3.4 Alternatives- Do nothing: Instead of creating a digital library, it may be just as effective to continue with a conventional library or to upgrade the library or information service in some other way. This may be a service consideration or it may, eventually, be a cost-based one.

If the library is providing a good service and the users are well adjusted to, and happy with, what is provided, then there may be no case for digitizing even part of the collection. One of the problems here is assessing if best use is being made of the collections and if the users are being most effectively served. To do this it is necessary to take a larger perspective than just that of the library. Often users do not know what alternatives could be made available to them and thus they don't know what they are missing. They may be satisfied, but not realize more is possible. The same is true for library staff who may well be doing a sterling job with the resources they have, not realizing that alternatives are available.

Out-Source: Even if it is decided that a digital library is needed, it may be that the best way to achieve it and run it is to give that task to another organization. The process of retrospectively digitizing the material from the chosen collections may be highly specialized and need expensive equipment for best results. Since this will be done once, contracting for the service may be the most cost-effective way.

Once the material has been digitized it is worth considering the costs of making it available, particularly if the service is to be provided externally and will be paid for.

Provide a Gateway: If the library is considering primarily providing access to already digitized material and/or material acquired from other parties, it may be sensible to consider a gateway operation. The gateway may provide its own indexing and search services and it may combine original resources from a number of different providers. Gateways are becoming quite common in the library world.

The major difference between outsourcing and running a gateway is that the outsourcing is entirely of your information and is an operation being run for you by a third party. A gateway is where your operation is linking to independent third party sources.

Both outsourcing and a gateway service can be provided through the organization's computer addresses, with the organization's logo and house style for the screens. If designed properly, the eventual users would be unable to tell if the digital library were in-house, outsourced, or a gateway operation.

2.3.5 Costs- Start-Up Costs: The variability in start up costs depends mostly on the material to be included in the digital library. Volumes are an obvious factor. The type of material and the degree of digitization and the completeness (resolution) of the digitization also affect the cost. Care and attention to fragile material adds to the cost.

All the above apply to local material to be digitized. If the material is to be acquired in digital form then it has an obvious cost. Once the material has been digitized, it has to be loaded into a suitable library application. Disruption inside and outside the library adds a cost to the whole exercise. If the service is to be made widely available (particularly outside the organization) then it must be advertised and promoted in some way.

If the service is to be made publicly available then there may well be registration and licensing costs involved as well as trademark and name protection. If some services will come from external organizations, then there is an obvious (though not necessarily easily quantified) cost attached to them.

Ongoing Costs: Addition of new material to the library incurs the same processing (or purchasing) costs as when the retrospective conversion was done. If there is a regular flow of material then it will be possible to negotiate a reduced per unit rate with a conversion specialist. For externally networked services there may be

telecommunications charges, particularly if the service is transferring large amounts of data across a third party network (such as the Internet or a public carrier network).

Every so often the physical capacity of the computers either to store material or to handle the number of users will be exceeded and they will need to be upgraded. The organization's IT policy and infrastructure will change and the application and data must be migrated. Regular backups of the data must be made, checked, and archived. Staff will change and new staff needs to be trained. Users will need regular training.

2.3.6 Copyright/IPR- Owner of the Material: The material may belong to the organization or it may belong to an affiliate or subsidiary of the organization. It may belong to a third party. It may be in the public domain. It may belong to a foreign organization subject to different copyright laws. It may belong to an individual. Or, of course, it may belong to a mixture of the above.

Having a copy of the material does not necessarily constitute ownership in terms of copyright laws. There is only one copyright owner however many copies are made. This is true for computer copies (digitized or otherwise) as well as physical copies.

Re-Use and Dissemination: Many countries allow for material to be copied for research purposes by individuals. However, making copies for re-sale or re-distribution is usually a matter for a commercial contract between the copyright owner and the organization wishing to re-distribute.

It is usual that the owners require that payment is made if all or part of the whole of the material is disseminated. It may even be that the owners require that they are the source for the eventual distribution of the whole material object. This is the case for many publishers where they allow their journal articles to be catalogued and indexed in retrieval systems, but the publisher must do the delivery of the full text of an article.

Charging: The digital library may not wish to charge for material, but may have no choice so far as copyright fees are concerned. If these fees are payable to the copyright holder then the organization will have to pay them. It may decide to absorb these fees itself as a service to its users. This is often the case where the library is exclusively used internally to the organization.

If the library intends to charge for the information it disseminates then the terms of its license with the copyright owner may change and the price the owner charges may be much higher than for free material.

2.3.7 Watermarks and Other Protections- The user's computer is inherently capable of copying any digital material on it. One protection against this practice is to introduce watermarks into the library's digital material. A watermark will not prevent copying, but it will mean that the owner of the copied material can be recognized. Modern systems can do this even if only part of the material (say a part of a picture) is copied. It is also not possible to overwrite one watermark with another without a special key.

The same protection can be provided to all digital material. There are various methods of watermarking, however, none of them are yet foolproof and they are often incompatible. Some require user-side software for full protection and this means they are limited to where this software can be mandated for access to the library.

Material may be disseminated in degraded form or it may be only partially disseminated. In these cases either further verification of the user is required before the full copy of the material is delivered, or the material is delivered by some other means usually as the physical delivery of a CD of the digital material.

2.4 Principles of Digital Library [3]

The purpose of a digital library is to provide coherent organization and convenient access to typically large amounts of digital information. The following principles provide working definitions of a digital library from both a conceptual and a practical standpoint:

- A digital library is an integrated set of services for capturing, cataloging, storing, searching, protecting, and retrieving information.
- Digital library services bring order where data floods and information mismanagement have caused much critical information to be incoherent, unavailable, or lost.
- Digital library architecture emphasizes organization, acquisition, preservation, and utilization of information.
- Digital library systems are realizations of architecture in a specific hardware, networking, and software situation.

2.4.1 Core Capabilities of Digital Library Systems- Digital library systems compose a family of automated systems that together provide a comprehensive capability to manage the digital content of an enterprise. It is useful to divide the capabilities of digital library systems into the following areas:

- Capture or creation of content.
- Indexing and cataloging (metadata).
- Storage.
- Search and query.
- Asset and property rights protection.
- Retrieval and distribution.

Content exists in multiple sizes, formats, and media, each with accompanying technical challenges. Content may be structured or unstructured. It may have exact, precise meaning; or it may be fundamentally ambiguous. Content may directly or indirectly support a business process or function.

Digital library architecture shows how capabilities are realized and related, and does this at several levels. Digital library architectures show how business processes or functions are enhanced; they show how technology components fit together and how, in detail, components interoperate with each other. Such functions and relationships, when reduced to a particular software and hardware implementation, lead to operational digital library systems.

2.5 Typical Digital Library Architecture

The following notional architecture for digital libraries is worth mentioning:

- A digital library approach to information management depends fundamentally upon a distinction between data and metadata. Metadata provide external classifying and organizing relations for data that may be unstructured, complex, or very large.
- Middleware services such as search, asset protection, and retrieval processes depend on metadata. Since metadata refers to data, which may be stored in

separate hierarchical storage subsystems, integrity of reference must be maintained between metadata and data.

2.5.1 Operational Architecture- Operational architecture is an information management system represented in terms of the business processes it supports. The example shown in Figure 2.3 is an enterprise that conducts training by utilizing an extensive computer-based simulation system. The operational (business) processes, most obvious in the example, depend on the timely and well-organized capture of training information as it happens, and both contemporaneous and retrospective search and retrieval of information from a training event. Although the information is generated in several different enterprise domains (eight in the example), effective utilization of information often depends on cross-domain searches and retrievals.

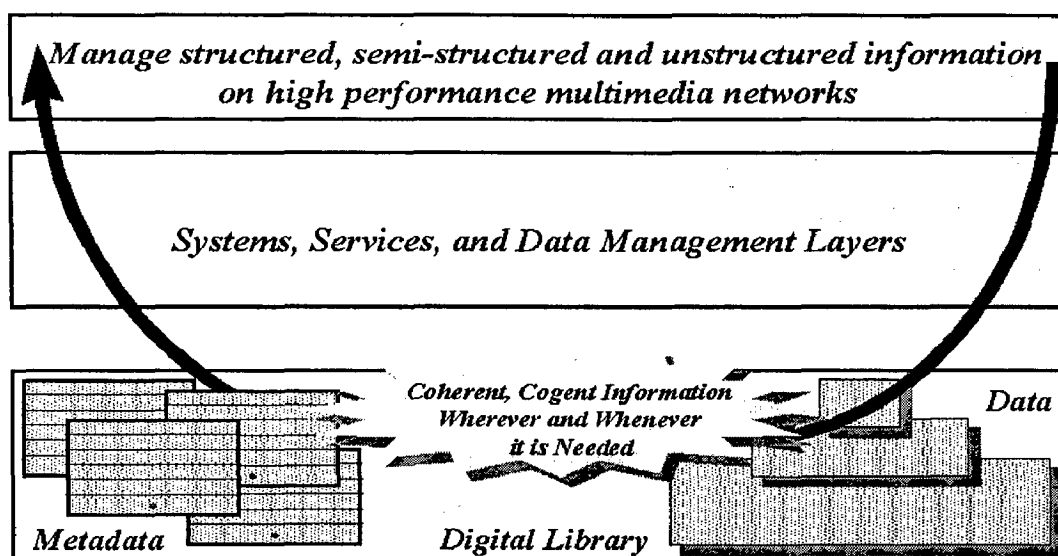


Fig 2.3: Operational Architecture.

2.5.2 Technical Architecture- A technical architecture breaks down operational (business) processes into functional components and capabilities (Figure 2.4).

Hardware and software implementations are still not resolved. The utilization of digital library materials depends on the existence of metadata to give an efficient and accurate view of content. Metadata must be created as content is added to the digital library. Metadata and data must be bound together logically, and there must be a robust

underlying technology to manage the logical connection through time, across platforms, and over geographical separations, all on a networked, distributed system.

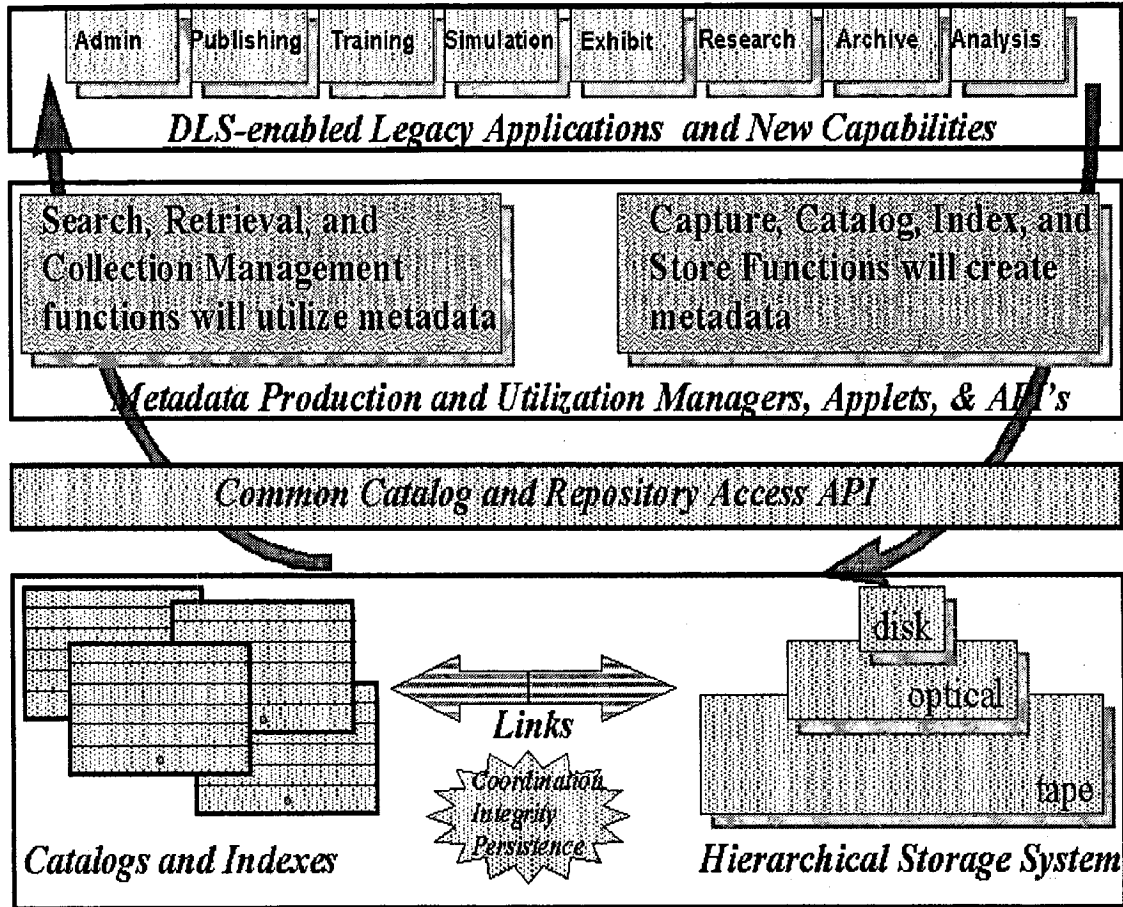


Fig 2.4: Technical architecture

CASE OF DIGITAL LIBRARY SYSTEM

3.1 Motivation and Background

All content is going digital. Content is documents, images, music, video, database records, multimedia. As a result of the World Wide Web / Internet frenzy of the last couple of years, consumers have come to expect information to be readily (if not instantly) available and inexpensive or even (virtually) free.

But Copyright holders, that is authors, artists, and their publishers and agents don't want their information to be free. They want to collect royalties or licensing fees when their original works are used or performed.

3.2 Safeguarding Digital Library Contents and Users: Assuring Convenient Security and Data Quality [4]

Digital library (DL) services must protect copyright holders, owners, users, and themselves against deliberate and inadvertent misuses of contents. Unobtrusive programs can provide sufficient protection; yet spare users irksome choices between distracting inconveniences and imprudent risks. Which safeguards are important depends on the kind of information resource and on the circumstances of its use.

DL protection extends well-known computing security practices. Novel elements include means of representing contractual obligations associated with information ownership, means for reducing loss outside library protection perimeters, and means for doing these and other tasks without adding to people's administrative burdens.

Safeguarding the economic, quality, and confidentiality interests of copyright holders and end users is an essential DL service. DL applications span widely disparate content types, values, origins, and longevities, and widely disparate user purposes and operating environments.

We can't expect 100% protection of anything. Safeguards are nearly always imperfect. Thus a realistic objective is to make misappropriation economically unattractive or, alternatively, to maximize benefits to the property owner, as suggested by Fig 3.1

Revenue is nil with either no security or too much security; from the mean value theorem, given a positive revenue point, there is also maximal revenue.

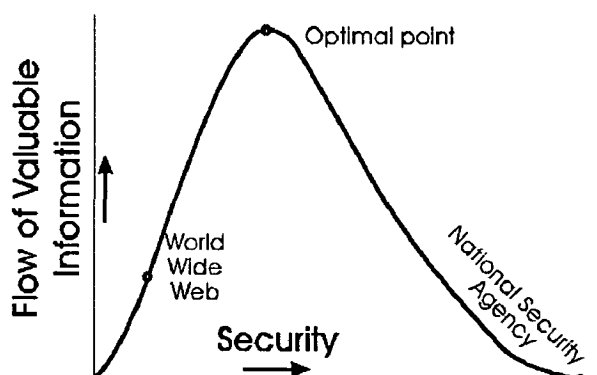


Fig 3.1: Balance between too little and too much protection

3.3 Structure of Library and Document Protection

3.3.1 Envelopes for distributing documents- Each package, called a Cryptolope, can include information so that a user can determine not only the price of any protected piece, but also its value. It also includes a bill of materials (BOM), digital signatures and cryptographic checksums of document portions, clear text elements sufficient to convey the value of encrypted document portions, and so on. The only obligatory portion is the BOM.

This packaging enables super distribution, information delivery before it is needed, possibly by inexpensive channels (e.g., network services at off hours). A Cryptolope can safeguard information it does not carry as surely as it can safeguard embedded, encrypted document parts.

Bill of Materials	
Clear Text 'teaser' (HTML)	
Encrypted Fingerprinting and Watermarking Instructions	
Encrypted Document Part	Key Record
Encrypted Document Part	Key Record
Encrypted Document Part	Key Record
Terms and Conditions	
Integrity Protection and Signatures	

Fig 3.2: Information Packing, called a Cryptolope

When a user decides to accept the terms and conditions of document use, a fast exchange of encryption, authentication, and financial data occurs between his workstation, the library service, and an associated clearance centers. The system decrypts the information granted and makes whatever accounting records are called for.

3.3.2 Protection Technologies- Secure Container Technology: Containers are software components that can contain a variety of different media objects; when accessed by a user, the container activates appropriate processes such as decryption, viewing, etc. The container is not an inert object such as a data file that can be opened and manipulated by a wide variety of applications; it incorporates code as well as data and only allows itself to be read or altered under specific conditions.

We discuss a snapshot of one of the currently available container technologies: InterTrust's DigiBoxes.

InterTrust's DigiBoxes Technology: It provides a way of securely encapsulating content to be managed and protected by the InterTrust rights management system. Software developers can write applications such as servers, browsers and plug-ins, which implement the DigiBox model by including code from software libraries provided by InterTrust. Data relevant to rights management, such as price and license information, are embedded in the container along with the content data-images and text. Security is provided by encryption throughout: it is used to prevent unauthorized access to the content of DigiBox containers, to protect the rights management components from tampering, and to ensure the privacy of users with respect to usage data.

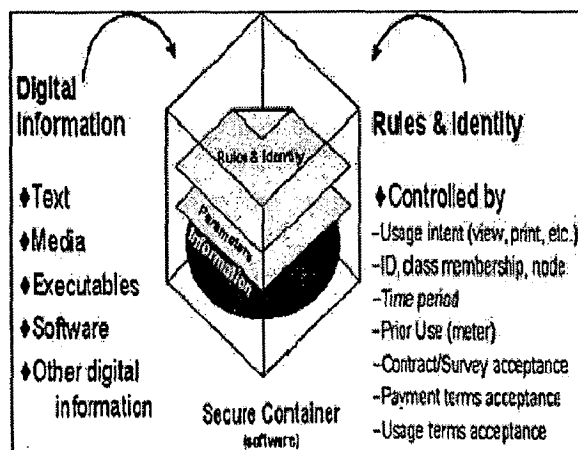


Fig 3.3: Digibox Secure Container

3.4 Intellectual Property Protection Systems [5]

3.4.1 Introduction- Adequate protection of digital copies of multimedia content, both audio and video, is a prerequisite to the distribution of these contents over networks. Until recently digital audio and video content has been protected by its size: it is difficult to distribute and store without compression. Modern compression algorithms allow substantial bit rate reduction while maintaining high fidelity reproduction. If distribution of these algorithms is controlled, clear text uncompressed content is still protected by its size. However, once the compression algorithms are generally available clear text content becomes extremely vulnerable to piracy. The implications of this vulnerability and the use of compression and watermarking in the control of piracy are discussed here.

Such large quantities of digital audio and video content data are difficult to distribute and store. Modern compression algorithms provide high-fidelity reconstruction while allowing substantial size reductions. Watermarking techniques can contribute to a system strategy that protects intellectual property.

3.4.2 A Systemic View of IP Protection- The design of secure systems should be based upon an analysis of the application risks and threats. As Fig 3.4 illustrates, such analysis will identify some of the risks of a particular domain.

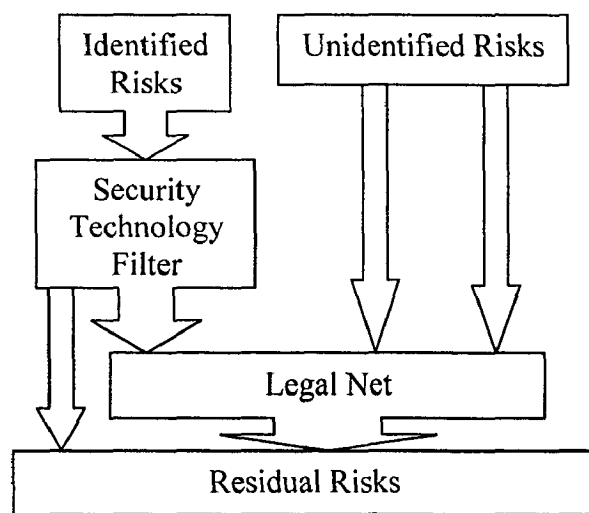


Fig 3.4: Identification of Risks

System security should not interfere with legitimate use. We want to design the system so that even if an attacker does break the system he cannot then use the same system to distribute that IP for his gain. Such a system should consist of following components:

1. A compression engine for managing music or video. This mechanism should discourage multiple compression/decompression cycles.
2. A mechanism for protecting the integrity of the content and for enforcing rights-to-use rules.
3. A flexible mechanism for licensing content and for granting various rights to consumers with appropriate credentials.
4. A secure client for accessing, rendering, playing or viewing content in a manner consistent with system policy and with the credentials or licenses associated with that content.
5. A mechanism for labeling the content to be distributed in a persistent manner.

Component 2 involves the use of cryptographic containers. The content is encrypted and perhaps digitally signed. The encryption keys are distributed via other channels using cryptographic protocols. A flexible licensing mechanism (Component 3) manages these keys and governs their use. Client security (Component 4) is what distinguishes the IP protection problem from the protected communications channel problem. That is, content must be protected in the client, not just in the channel. Protection mechanisms include tamper resistant software and hardware.

3.5 Digital Watermarking [5, 6, 7]

Digital Watermarking technology allows us to embed in audio, images, video and printed documents a digital code that is imperceptible during normal use but readable by computers and software. The science of creating these imperceptible codes is known as digital watermarking.

3.5.1 Brief Technical Appreciation – Three types of attack are identified on the intellectual property contained in software and three corresponding technical defenses are suggested. A defense against reverse engineering is obfuscation, a process that renders software unintelligible but still functional. A defense against software piracy is watermarking, a process that makes it possible to determine the origin of software. A defense against tampering is tamper proofing, so that unauthorized modifications to software (for example, to remove a watermark) will result in nonfunctional code.

Electronic Copyright Management Systems (ECMS) deal with the problem of electronic management of copyright. In particular, the use of digital watermarking techniques satisfying the requirements of ECMS is important.

The progress over the last few years of digital technologies, the rapid development of the Internet and of other communications means, have caused an ever-increasing need for protecting Intellectual Property Rights (IPRs). Traditional laws on IPR protection do not seem suitable to solve all the problems raised by this technological revolution. We focus on the main requirements that ECMS should satisfy, and the technologies that will be most probably employed, with a particular attention to digital watermarking.

A mechanism is needed for binding content identification. Digital watermarking is such a mechanism. In this case, when decisions regarding access to or use of the content are made, the mark must be retrieved in real-time and used as input in the decision making process. No one marking algorithm is best suited for these two functions, both because of complexity issues and because different functions and different marking algorithms are resistant to different attacks.

3.5.2 System Attacks- We list several general classes of attacks against information embedded in multimedia content. The use of a watermarking algorithm for a particular application needs to be sanity-checked against this list to determine whether or not the watermark serves any useful purpose. One attack is forgery of identity. Whether the watermark is a point-of-sale watermark (a fingerprint) or pressing-plant watermark, if the input to the marking process is fraudulent, then the watermark doesn't protect the IP. Works distributed in versions distinguished only by different watermarks are susceptible to collusion attacks. The simplest collusion attack is the bitwise XOR attack. The attacker compares the differences between two representations of the same work, and jams differing bits to 0s or 1s. The jam pattern can be either random or – if the attacker has knowledge of the marking algorithm – one that creates a counterfeit of a legitimate mark. When a work is to be marked in multiple versions each with its own markings, the marking algorithm must be designed so that in the presence of tampering one of three conditions holds. The work should be impossible to decompress, the quality of the decompressed output should be significantly degraded, or the mark should nonetheless be recovered from the bits, which were not changed by the attacker.

Another collusion attack specific to frame-based compression algorithms can be effective against marks that extend in time through the work. In these algorithms, a bit stream is composed of frames, each representing a segment of the original signal.

3.5.3 Two Possible Approaches- In the design of effective ECMSs, two different approaches can be distinguished to solve the problem of IPR protection: the first approach is aimed at developing systems able to prevent copyright violations (as for example the IBM proposal named Cryptolope), the second is on the contrary aimed at developing systems able to track copyright violations. Both categories of systems require that multimedia works are properly managed before their distribution; in particular, the works can be:

(i) Wrapped in an encrypted system and integrated with an application (the reader) allowing to use the work only in a controlled manner; e.g. the images can just be displayed, but they can not be printed, or the audio files can be played but can not be stored in the hard disk of the user. The content cannot be accessed without the proper application. The main disadvantages of this approach are that a standard for the embedding applications is difficult to be established; moreover when the works are enjoyed (for example, they are visualized in a PC screen, or they are played), it is still possible to capture and copy them without constraint.

(ii) Watermarked through digital watermarking techniques whose aim is to tightly and robustly embed IPR related information into purchased creations (the hidden data can be the name of the copyright owner, or the unique code identifying the work); it is thus possible to check the legal status of the content exchanged through the network.

3.5.4 General requirements for Watermarking Algorithms – Though, in general, the requirements to be fulfilled are common to most practical applications. In the following, such general requirements are listed and briefly discussed:

Security: As for cryptography, it is well known that the effectiveness of an algorithm can not be based on the assumption that possible attackers do not know how the code mark has been embedded into the multimedia document. Though some of them are claimed to be exceptionally resistant, by knowing how the watermark encoder and decoder work, it is usually easy to make the watermark unreadable.

Invisibility: Though some applications require the watermark to be visible, others only focus on invisible watermarking (in general the term imperceptible should be used). So far researchers have tried to hide the watermark in such a way that data quality is not degraded and attackers are prevented from finding and deleting it. However, this requirement conflicts with other requirements such as tamper resistance and robustness, specially against lossy compression algorithms.

Number of bits, which can be hidden: Depending on the application at hand, the watermarking algorithm should allow a predefined number of bits to be hidden. In the case of images, the possibility of embedding into the image at least 300-400 bits should be granted. In any case, the number of bits which can be hidden into data is not unlimited, very often is fairly small, due to the fact that the higher the number of embedded bits, the more perceivable the watermark will be.

Low error probability: Even in the absence of attacks or signal distortions, the probability of failing watermark detection (false-negative error probability) and detecting a watermark when, in fact, one does not exist (false-positive error probability), must be very small to allow the use of watermarking systems as unambiguous evidence of ownership on a legal basis.

Robustness: The use of sounds, images and video signals in digital form commonly involves many types of distortions, such as lossy compression, or, in the image case, filtering, resizing, contrast enhancement, cropping, and rotation and so on. Watermarking tools must grant that the embedded information is not removed by these accidental modifications by neither collusion nor forgery attacks.

Watermark invertibility/reversibility: Though robustness is commonly indicated as the major requirement to be satisfied, great attention should be given to watermark invertibility as well. The most natural meaning of invertibility is the one defining a watermark to be invertible if authorized users can remove it from the document.

3.5.5 Desirable Characteristics of Watermark Algorithms- The following requirements are typically expected of watermarks:

- **Imperceptibility-** A watermarked signal should (usually) not be distinguishable from the original signal.

- Information capacity- The mark bit rate must be compatible with the rate limits imposed by the system.
- Robustness- The mark must be recoverable, not only in the complete work, but also in truncated, filtered, dilated, and otherwise processed clips, in a concatenation of unrelated content, and in the presence of noise.
- Low complexity- Marking schemes intended for use with real-time applications should be low complexity.
- Survive multiple encode-decode generations- A watermark should survive tandem encoding-decoding.
- Tamper resistant or tamper evident- It should be possible to recognize that a mark has been modified. It should not be possible to modify a mark in such a way as to create a different valid mark.
- Difficult to create or extract legitimate watermark without proper credentials- In the context of the watermarking engine alone, a proper credential is knowledge of the algorithm used to insert the mark. An ideal would be a public key analogue to watermarking: hard to insert mark, easy to retrieve, hard to counterfeit.

For copyright identification every copy of the content can be marked identically, so the watermark can be inserted once prior to distribution. Not only must the watermark be short enough to be recovered in a truncated version, some means must be provided to synchronize the detection process so that the watermark can be located in the processed bit stream. Finally, any attempt to obscure the mark, including re-encoding the content, should lead to perceptible distortion.

Transaction identification requires a distinct mark for each transaction. That is, the algorithm must be low complexity. One strategy is to insert the watermark in the compressed domain, in which case mark insertion should increase the data rate very little. Watermarking algorithms designed for fingerprinting must be robust to collusion attacks.

3.5.6 General Mechanisms- Watermarks for compressed content fall into three categories: clear text or original (PCM in the case of audio or video) marking, compressed bit stream marking which does not alter the bit stream semantics, and marking integrated with the compression algorithm in which the semantics of the bit

stream are altered. These are described below along with their advantages and limitations:

1. Clear text PCM: Clear text watermarks are marks inserted in the original text or during decompression into output (e.g. while writing a decompressed song to CD). Clear text marking embeds a data stream imperceptibly in a signal. The model for many clear text-marking algorithms is one in which a signal is injected into a noisy communication channel, where the audio/video signal is the interfering noise. Because the channel is so noisy, and the mark signal must be imperceptible, the maximum bit rates that are achieved for audio are generally less than 100bps. There are two major concerns with clear text marking. Because such algorithms (usually) compute a perceptual model, they tend to be too complex for point-of-sale applications. Second, these algorithms are susceptible to advances in the perceptual compression algorithms. Retrieval mechanisms for clear text watermarks fall into two classes: reference necessary and reference unnecessary. In either case the mechanism for mark recovery is generally of high complexity and is often proprietary.

2. Bit stream Watermarking (semantic-non-altering): Bit stream marking algorithms manipulate the compressed digital bit stream without changing the semantics of the audio or video stream. Bit stream marking, being low-complexity, can be used to carry transaction information. Because the mark signal is unrelated to the media signal, the bit rate these techniques can support can be as high as the channel rate. However these marks cannot survive D/A conversion and are generally not very robust against attack; e.g. they are susceptible to collusion attacks. This type of mark can easily be extracted by clients and is thus appropriate for getting access to content.

3. Bit stream Marking Integrated with Compression Algorithm (semantic altering): Integrating the marking algorithm with the compression algorithm provide improvements in hiding data imperceptibly in content thereby motivating further improvements in perceptual compression algorithms. Since the perceptual model is available from the workings of the compression algorithm, the complexity associated with marking can be minimized. Integrated marking algorithms alter the semantics of the audio or video bit stream, thereby increasing resistance to collusion attacks.

3.5.7 Integrating the Watermarking Algorithm with Compression- A system that combines bit stream and integrated watermarking can be configured to support the three marking functions mentioned above. It does not include but is compatible with use of a front-end clear text-marking algorithm as well. It is assumed that the original clear text is not available except possibly to auditors seeking to recover the watermark. The decompressed and marked content will generally be available to everyone.

This method relies on the fact that quantization, which takes place in the encoder, is a lossy process. By combining mark insertion with quantization it is ensured that the attacker cannot modify the mark without introducing perceptible artifacts. The fact that marking data is present is indicated by characteristics of the bit stream data. The marking technique involves the perceptual modeling, rate control, quantization, and noiseless coding blocks of a generic perceptual coder. In MPEG, AAC spectral lines are grouped into 49 scale factor bands (SFB), each band containing between 4 and 32 lines. Associated with each band is a single scale factor, which sets the quantizer step-size, and a single Huffman table (AAC employs 11 non-trivial Huffman tables). The coefficient for each spectral line is represented by an integer (i.e. quantized) value. In MPEG video, a block consists of 64 coefficients, and each set (termed a macro block) of 6 blocks has an associated quantization step-size Q_p . The same Huffman table is used for the coefficients for all Q_p values. As with audio, each coefficient is represented by an integer after quantization.

Let $A = \{f_i, H_i, \{q_{ij}\}\}$ be the set of triples of scale factors f_i , Huffman tables H_i , and quantized coefficients $\{q_{ij}\}$. (Only one Huffman table is used in video.) It is assumed that some set of scale factor bands into which mark data will be inserted have been selected. The marking set will generally be dynamic. Let M be the set of indices associated with the set of SFB chosen for marking. A set of multipliers $\{x_i: i \in M\}$, with all x_i close to unity is chosen. The triple $\{f_i, H_i, \{q_{ij}\}: i \in M\}$ is modified as follows. Let $\{v_{ij}\}$ be the set of spectral coefficients prior to quantization, and Q_i be the quantizer for SFB i.e. $\square_i \{q_{ij}\} = Q_i[\{v_{ij}\}]$.

Then $\{f_i, H_i, \{q_{ij}\}\} \square \{f_i', H_i', \{q_{ij}'\}\}$, where

$$f_i' = f_i/x_i$$

$$q_{ij}' = Q_i'[x_i \times v_{ij}]$$

$$H_i' = H_i \text{ or the next larger codebook}$$

$$x_i \square 1$$

Because the modification to the spectral coefficients occurs before quantization, the changes to the reconstructed coefficients will be below perceptual threshold. If this change were introduced after quantization, the change in some quantized values would be greater than the perceptual noise floor. Equivalently, an attacker who modifies the quantized values to eradicate or modify the mark will be introducing energy changes that exceed the noise floor. Because the changes in step-sizes will be small, because not all coefficients will change, and because the attacker will not have access to the uncompressed clear text source material, the attacker will generally not be able to identify those SFB which are used for marking. Further, the increase in bit rate associated with marking should be small, and so must be monitored. A feedback mechanism can be used to prevent modification of scale factors that would increase the bit rate significantly.

Watermark bits can be inserted in a variety of ways. Generally watermark sequences are inserted a few bits per frame. The data to be carried by the stream is typically mapped into a marking sequence prior to embedding, where the characteristics of the mapping function depend on the type of attack expected. Indeed, since there may be a wide range of attacks, the data may be redundantly mapped in different ways in the hope that at least one mapping will survive all attacks.

The marking sequence can be recovered by comparison to a reference or through the use of synchronization codes. If synchronization codes are used, the watermark can be recovered in the compressed domain through a lightweight recovery process. It can therefore be used for gating access to content. Although the attacker can use the gating mechanism to probe for the watermark sites and perhaps damage the synchronization codes, damage to the codes will generally produce perceptible artifacts.

Audio Results: To evaluate audio watermarking algorithm, AT&T's implementation of AAC is used. Watermark synchronization is indicated by the sequence comprising the LSB of the first 44 decoded scale factors in a long block. When the value of the LSB of a scale factor does not match the corresponding bit in the synchronization code then the scale factor is decremented and the spectral coefficients adjusted accordingly, resulting in perceptually irrelevant over coding of the associated spectral data. The following table shows the cost of carrying watermark data inserted into every frame of an AAC bit stream for a stereo signal sampled at 44.1 kHz and coded at 96 kbps.

	Increase in bits (per marked frame)	Increase in rate
Synchronization	5.2	0.25 %
Sync + 32 bits	9.0	0.44%

Table 3.5: Increase in Audio Bit Rate

Video Results: The baseline system for video compression uses a rudimentary perceptual model. A variance-based activity measure is used to select the quantization step-size for each macro block as in step 3 of the MPEG-2 TM5 rate control. I frames are generated every half second; all other frames are P frames. Watermark data into both I and P frames are inserted and results taken from an average over two different 10 second sequences is presented. The first 44-macro blocks of a frame are used for synchronization. The next several macro blocks (100 or 600 in the Table, of 1320) of a frame carry mark bits. For each macro block, when the LSB of the step-size Q_p does not match, Q_p is decremented. However, a dead-zone is applied to the original Q_p to ensure that zero coefficients remain zero.

	Increase in bits (per marked frame)	Increase in rate
Synchronization	124	0.005 %
Sync + 100 bits	138	0.006%

Table 3.6: Increase in Video Bit Rate

Formatting Watermark Data: For transaction watermarking, the bits representing differently marked versions of the same content should have bit streams which are either nearly the same or as different as possible. A simple method for formatting watermark data that is relatively resistant to XOR collusion attacks is developed. Although this technique is used for formatting watermark data for a semantic non-altering scheme, it is more generally applicable. The set identifying data (e.g. transaction identities), one datum of which is to be formatted, can be put into a linear sequence. For example, each transaction that occurs on 31st April 2004 can be marked uniquely so that Transaction 1, Transaction 2, and so on can be identified. Instead of representing the Nth transaction by the ordinal N, it is represented by 2N-1. It is assumed that no further formatting of the mark data has been performed. When an attacker bitwise XORs two copies of the same content, the resulting sequence will indicate both the first transaction and the second. If the attacker sets or clears the bits identified by the XOR operation, then the resulting mark is identical to one of the original marks. If the projected bits are randomized, then the mark is invalid. This exponential sequence is inadequate by itself as a hiding mechanism. What needs to be protected from the attacker is the location of the transitions. This can be accomplished by permuting the bits of the sequence, possibly after XORing them with a mask. The bits of the watermark sequence can also be interleaved with other data. Finally, the watermark sequence can be redundantly inserted. These manipulations hide the transition in the watermark sequence, so that the result of an XOR of two bit streams (which differ only in the sequences with which they are marked) appears as a random jumble of 1s and 0s.

3.5.8 Example of watermarking technique in the tuscany&gifu art virtual gallery-

Aiming at demonstrating the feasibility of watermark-based copyright protection, an Art Virtual Gallery containing watermarked images representing works of art of Tuscany (Italy) and Gifu (Japan) artists has been created by the LCI (Laboratorio Comunicazioni Immagini) of the DET (Dipartimento Elettronica e Telecomunicazioni) of the University of Florence (Italy) and the ManART of Toki-shi (Japan). This Gallery has been named Tuscany&Gifu Art Virtual Gallery because all the pictures exposed in it represent ceramics, paintings, sculptures and so on, realized by some artists coming from the regions of Tuscany (Italy) and Gifu (Japan). A watermarking technique developed by the

LCI-DET has been used to embed in each image a specific mark; this mark is composed by the Gallery Name (ManART&LCI) and a code number identifying the image itself (ID). The insertion of the watermark does not result in image quality degradation and the watermarked image is perfectly identical to the original one, obviously under a human-eye observation. After choosing the image to see and clicking on the representative icon, an Art Gallery visitor will receive a watermarked image as proposed, for instance, in Fig 3.5 where a sculpture on stone by Pirzio titled 'Uomini di sempre' is depicted:



Fig 3.5: An image of the Tuscany&Gifu Art Virtual Gallery.

The user is allowed to download the images in his computer, and to verify the presence on every image of the respective mark, by connecting to the watermark detection service. A watermark detector is also provided, as illustrated in Fig 3.6, which has been created to allow each visitor of the Gallery to practically verify how a watermark detection phase can take place. In this application there are three blank fields to fill in with the following data:

- The Gallery name which is the same for each pictures: ManART&LCI;
- The ID number which is the identifier code of the work of art;
- The path of the image file in JPEG format to be checked.

After writing these codes, to start the investigation of the presence, in the chosen image, of the supplied watermark, the Submit button is to be clicked and in few seconds the positive/negative answer of the requested research will be obtained. To further test the powerful characteristics of the used algorithm from a robustness point of view, some usual processing, as contrast enhancement, linear or non-linear filtering, etc., or geometric transformations, as cropping, rotation, etc., can be applied to an image before going to the detection step; if a heavy quality degradation has not been introduced by means of these attacks, the correct retrieval of the watermark can be verified again.

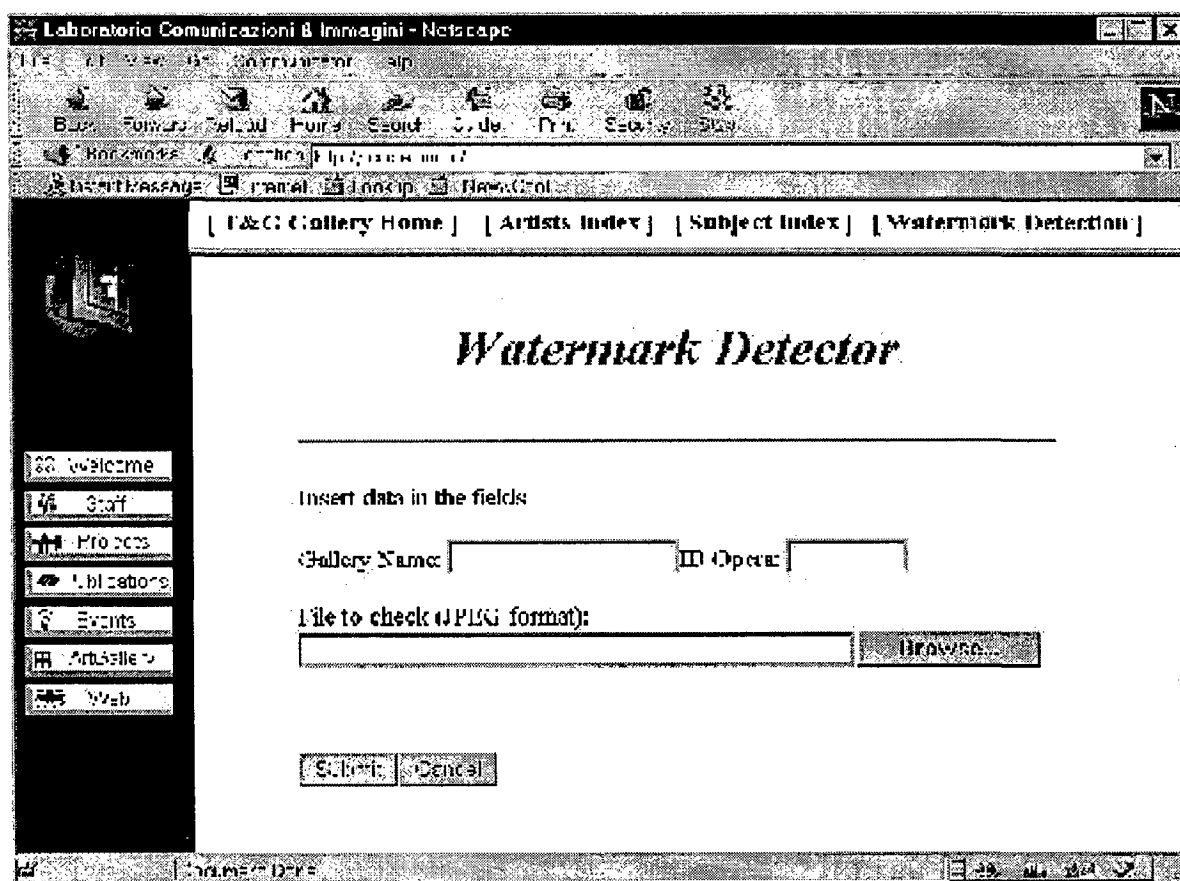
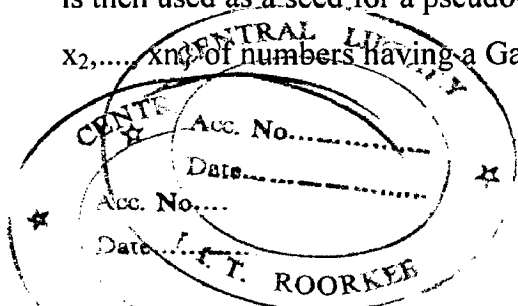


Fig 3.6: The watermark detector

3.5.9 The watermarking algorithm- The images are marked according to the scheme briefly described below. The algorithm encodes the two strings the mark consists of, based on a set of conversion tables in such a way to generate a set of 4 integers. This set is then used as a seed for a pseudo-random generator which produces a sequence $X = \{x_1, x_2, \dots, x_n\}$ of numbers having a Gaussian distribution, with zero mean and unity variance.



The obtained sequence is the watermark, that is used to modify the magnitude of a selected set $V = \{v_1, v_2, \dots, v_n\}$ of full-frame DFT coefficients of the image. Watermark embedding is accomplished according to the following rule:

$$v'_i = v_i + av_i x_i \quad (1)$$

where v'_i is the magnitude of modified DFT coefficient, and a is the watermark energy. After watermark embedding, the modified DFT coefficients are reinserted in their position, and an inverse DFT is applied, obtaining a temporary watermarked image. To enhance watermark robustness, without compromising its invisibility, the characteristics of the Human Visual System are then exploited by blending the original image and the temporary watermarked one based on a spatial masking image. In watermark detection, the image to be tested is DFT-transformed again, and the same set of DFT coefficients used in the marking stage is selected. As a measure of the presence of the mark, the correlation between the extracted DFT coefficients and the mark itself is considered. The correlation is simply compared to a predefined threshold, so that if the correlation is larger than the threshold, it is decided that the given watermark is embedded in the image, otherwise it is assumed that the watermark is not present. Together with the watermark synchronization pattern is embedded during the coding phase, to allow to determine if a geometric transformation, as rotation, scaling etc., occurred; in this case the inverse operation can be applied to obtain again the initial conditions of the watermarking step. The retrieval of the inserted watermark is granted even when some attacks, such as geometric distortions, cropping, and so on, and some processing tools, such as filtering, JPEG compression and so on, have been applied to the image, with the help of this strategy along with the intrinsic robustness of the watermarking algorithm, applied in the gallery.

3.5.10 Limits of current watermarking techniques – Digital watermarking is a recent research field, therefore its intrinsic limits are not well understood yet. On the other hand, more insight into the technical possibility of satisfying the requirements imposed by practical applications is needed, if the practical possibility of using watermarking for copyright protection is to be evaluated. In the following, some of the most common limits shared by digital watermarking schemes are described:

- Even if the evaluation of the maximum number of information bits that can be hidden within a piece of data of a given size plays an important role, such an issue has not been satisfactorily addressed yet. Existing works addressing this problem model the watermarking process as a communication task.
- A blind watermarking algorithm, which is really robust, does not exist yet. In the image case, robustness is still an open issue. More specifically, resistance to geometric manipulations such as cropping is recognized as a very difficult goal to achieve in a computationally efficient way.
- Owners can erase the mark: virtually all the blind watermarking schemes proposed so far are reversible according to the definition previously given. In other words, by knowing the exact content of the watermark, and the algorithms used to embed and retrieve it, it is always possible to make it unreadable without any significant degradation of the data.

3.5.11 Digital Watermarking Applications- Here we discuss digital watermarking applications and an overview of the digital watermarking capabilities:

Asset Management: Digital watermarks can be used as a persistent media asset tag, acting as keys into a digital asset management system.

Authentication- Authentication identifies if content has been altered or falsified. For example, digital watermarks can verify authenticity and identify counterfeiting as a second layer of security for encrypted content or for content in the open.

Broadcast Monitoring: Broadcast monitoring enables content owners and distributors to track broadcast dissemination of their content. Broadcast monitoring is currently applicable to radio and TV, and will be applicable to multicast streams.

Copy Prevention: Copy prevention and play control uses the digital watermark to identify whether content can be copied. It guards against unauthorized duplication by detecting the copy control watermark information.

Copyright Communication: Digital watermarks enable copyright holders to communicate their ownership and offer links to copyright and purchase information, thereby helping to protect their content from unauthorized use, enabling infringement detection.

E-Commerce/Linking: e-Commerce links content to related information, usually on the Internet. The value is that digital watermarking enables the user to purchase or access information about the content, related contents, or items within the content.

Forensic Tracking: Forensic tracking locates the source of content, especially illegitimate content. The key advantage of digital watermarking is that it enables tracking of content to where it leaves an authorized distribution path.

Filtering/Classification: Classification / filtering enable content to be identified, classified, and used appropriately. This enables users to selectively filter potentially inappropriate content, such as consumers determining what content is appropriate for their children.

Rights Management: Digital watermarking enables digital rights management (DRM) systems to connect content outside the DRM back to the DRM, such as linking the content to usage rules, billing information, and other critical metadata. Digital watermarking allows content to pass through the analog domain.

3.5.12 Comparison with other Security Technologies-

1-D and 2-D Bar- Codes	Bar codes allow data to be stored on cards. 1-D conforms to AAMVA standard. 2-D conforms to AAMVA and PDF47 standards.
Altered (or Modified) Fonts	Slight modifications of text characters are obvious only to those who are trained to look for them.
Biometrics	Biometrics are used for both 1-to-1 authentication and 1-to-many verification of the new applicant's unique identity. This reduces the opportunity for obtaining valid identity documents under false pretext.
Fine-Line Printing	A pattern of fine lines, similar to those found on currency, can be placed on documents and photo backgrounds. This feature can thwart attempts at photocopying.
Ghost Image	A faint photo image covers printed data, making it virtually impossible to alter. Requires no special equipment for verification.
Signature Area	Signature is captured electronically and printed digitally.
Split Fountain	Use of a color degrade that can't be color copied.

Table 3.7: Comparison of Digital Watermarking with other Security Technologies

3.6 Super Distribution [8, 9, 10, 11, 12]

3.6.1 Overview- Super Distribution is the exploitation of cheap (broadcast) distribution channels such as CD-ROM, Internet, and Digital Cable and Satellite modems to essentially broadcast copyrighted content or proprietary information to all potential users of the information. Super Distribution is an approach to distributing software in which software is made available freely and without restriction but is protected from modifications and modes of usage not authorized by its vendor. By eliminating the need of software vendors to protect their products against piracy through copy protection and similar measures, super distribution promotes unrestricted distribution of software. The Super Distribution architecture provides three principal functions: administrative arrangements for collecting accounting information on software usage and fees for software usage; an accounting process that records and accumulates usage charges, payments, and the allocation of usage charges among different software vendors; and a defense mechanism, utilizing digitally protected modules, that protects the system against interference with its proper operation. Super Distribution software is distributed over public channels in encrypted form. In order to participate in Super Distribution a computer must be equipped with an S-box, a digitally protected module containing microprocessors, RAM, ROM, and a real-time clock. The S-box preserves secret information such as a deciphering key and manages the proprietary aspects of the super distribution system. A Software Usage Monitor ensures the integrity of the system and keeps track of accounting information. The S box can be realized as a digitally protected module in the form of a three-dimensional integrated circuit.

Super Distribution relies neither on law nor ethics to achieve the protections; instead it is achieved through a combination of electronic devices, software, and administrative arrangements whose global design we call the 'Super Distribution Architecture'.

Super Distribution of software has the following novel combination of desirable properties:

- Software products are freely distributed without restriction. The user of a software product pays for that product, not for possessing it.
- The vendor of a software product can set the terms and conditions of its use and the schedule of fees, if any, for its use.

- Software products can be executed by any user having the proper equipment, provided only that the user adheres to the conditions of use set by the vendor and pays the fees charged by the vendor.
- The proper operation of the super distribution system, including the enforcement of the conditions set by the vendors, is ensured by tamper-resistant electronic devices as digitally protected modules.

From a different viewpoint, the needs of users and the needs of vendors have until now been in irreconcilable conflict because the protective measures needed by vendors have been viewed by users as an intolerable burden. The Super Distribution architecture provides a solution to that conflict that serves the interests of both parties. Wide distribution benefits vendors because it increases usage of their products at little added cost and thus brings them more income. It benefits users because it makes more software available and the lower unit costs lead to lower prices. It also creates the possibilities of additional value-added services to be provided by the software industry. Moreover, users themselves become distributors of programs that they like, since with super distribution there is absolutely nothing wrong with giving a copy of a program to a friend or colleague.

It might seem at first that publicly distributed software such as freeware and shareware already solves the problem addressed by Super Distribution. But the likelihood of the authors being paid is too small for public domain software to play a leading role in the software industry. Super Distribution software is much like public domain software for which physical measures are used to ensure that the software producer is fairly compensated and that the software is protected against modification.

3.6.2 Super Distribution Architecture- The super distribution architecture provides three principal functions:

- Administrative arrangements for collecting accounting information on software usage and fees for software usage.
- An accounting process that records and accumulates usage charges, payments, and the allocation of usage charges among different software vendors.
- A defense mechanism, utilizing digitally protected modules that protects the system against interference with its proper operation.

A computer must be equipped with a device known as an S-box (Super Distribution Box) in order to participate in Super Distribution. An S-box can be installed on nearly any computer, although it must be specialized to the computer's CPU type. It is also possible to integrate the S-box directly into the design of a computer. A computer equipped with an S-box is called an S-computer. Programs designed for use with Super Distribution are known as S-programs.

In order to make it acceptable to users, software vendors, and hardware manufacturers, the super distribution architecture has been designed to satisfy the following requirements:

- The presence of the S-box must not prevent the host computer from executing software not designed for the S-box. The presence of the S-box must be invisible while such software is active.
- The modifications needed to install an S-box in an existing computer must be simple and inexpensive.
- The initial investment required to make S-programs generally available must be small.
- The execution speed of an S-program must not suffer a noticeable performance penalty in comparison with an ordinary program.
- The S-box and its supporting protocols must be unobtrusive both to users and to programmers.
- The S-box must be compatible with multi-programming environments, since we anticipate that such environments will become very common in personal computers.

The design of Super Distribution architecture can be decomposed into four tasks:

- Design of the Super Distribution network.
- Logical design of the S-box and its interfaces.
- Design of the supporting software and file structures, including appropriate cryptographic protocols and techniques.
- Design of the protection for the S-box.

S-programs are stored and transmitted in encrypted form. So the transmission paths need not be protected in any way. Each S-computer that is, an end-user computer supporting

Super Distribution is equipped with an S-box. The S-box contains a metering program, the Software Usage Monitor (SUM), which enforces the terms set by each software vendor for executing that vendor's products and keeps track of how much the user owes to each vendor. The S-box generates a payment file that contains this information. The fees charged for software usage are measured in units called S-credits. Payment files are encrypted and transmitted to the collection agency.

The collection agencies receive the payment files from users, process those files, and transmit payments to vendors, both the authors of the S-programs and the manufacturers of the S-boxes. Each collection agency has an S-box in its computer. The payment files are decrypted and processed under control of this S-box so as to ensure the integrity of payments to the vendors of S-programs and of the supporting hardware as well.

The clearing house keeps track of funds transfers engendered by Super Distribution. The clearing house can be either a new organization or an existing one, e.g. a credit card company that is prepared to provide the necessary services. The advantage of creating a new organization is that it provides additional degrees of freedom in the systems design. Using an existing organization as the clearing house gives that organization the opportunity to enlarge its market.

Identification numbers (ID's) are essential to this arrangement. Each user, each software vendor, each S-box manufacturer, and each collection agency has a unique ID. In addition, users can also have ID's, either as individuals or as organizations. The ID of a user is usually in a different name space than the ID of that user's machine. User ID's provide a convenient mechanism for establishing credit, since they are associated with the individual or organization who is actually paying for the software. On the other hand, a software distribution system based purely on S-box ID's can provide a high degree of privacy for users.

The super distribution scheme does not provide any absolute guarantee that users will pay what they owe, or even that they will return the necessary payment files to the collection agents. However, the S-box, which contains a real-time clock, will suspend its services (other than transmitting payment files) if the conditions, which are specified by the vendor or the system, are not met. Transmission can be either over a telecommunication link or with a memory card. In the event that a user transmits the payment files but does

not remit the corresponding payment, a collection agent can apply appropriate sanctions. To join the system, a user need only insert an S-box, most likely in the form of a coprocessor chip, into his or her own personal computer. A collection agency can join the system in the same way. The collection agency needs a slightly different form of S-box, one that contains the software for decoding and processing the payment files. Processing at and between the collection agencies and the clearing house can be conveniently handled by a Credit Authorization Terminal (CAT).

3.6.3 Design of the S-Box- The function of the S-box is to execute the Software Usage Monitor in a secure way. The S-box also contains the cryptographic keys that ensure the integrity of the system. These keys are kept secret by storing them within digitally protected modules. An S-box can be added to an existing computer in any of several ways:

1. Attaching it to an I/O port.
2. Connecting it to the bus.
3. Placing it between the CPU and the bus.
4. Placing it on the same chip as the CPU or inside the CPU.

Methods (1) and (2) require no modification to the computer. This method has the disadvantage that it introduces significant overhead in a multi-programming environment. Method (4), though more difficult to implement initially, offers the best performance and the lowest cost.

This connection is best realized by making the S-box a coprocessor and using an existing coprocessor socket. The S-box provides a limited form of protection against viruses, in that programs that are specifically designed to work with the S-box cannot be modified at the user's computer and so cannot be disabled or other wise taken over by a virus. However, the S-box does not protect against programs, distributed through super distribution or otherwise, whose effect is on parts of the system unrelated to the S-box. The workings of the S-box depend on the existence of cryptographic keys within the S-box.

The Software Usage Monitor: the Software Usage Monitor is executed in the S-box. It performs the following functions:

- It monitors the execution of an S-program in order to make sure that it is only executed in the manner intended by its vendor.
- It encrypts and decrypts S-programs and other necessary information.
- It maintains an account of the charges associated with the execution of an S-program.

An S-program can issue instructions to the Software Usage Monitor. These instructions are realized as extended instructions of the CPU.

3.6.4 Design of the Supporting Software and File Structures- Structure of an S-Program- The pricing information for an S-program is contained in its tariff section. When execution of the S-program is initiated, the coprocessor verifies the information in the tariff section and copies the pricing information into the accounting buffer of the S-box. A verification routine is embedded in the body of the S-program. Execution is aborted if the verification routine detects any inconsistency in the tariff section.

The tariff section includes the following:

- Software ID: an identifier unique to the S-program.
- Access control key: A key used by the authentication routine in the S-program body. This key is checked by the coprocessor.
- Encrypting key: A key used to encrypt the machine instructions in the body of the S-program. An encrypted block of instructions is transferred to the coprocessor, where it is decrypted using this key and then executed.
- Charging interval: the length of execution time for which a single S-credit is charged to the user.
- Service pricing schedule: the number of S-credits charged to the user each time a particular software service, e. g. saving a file, has been rendered and successfully completed.
- The amount of memory required within the coprocessor for executing the S-program.

Execution of an S-Program: Execution of an S-program is monitored by the program and the coprocessor working together. The coprocessor halts execution whenever it detects a

protocol violation. For proper protection, the S-software must be structured so that analysis of it by an intruder is not possible. Some methods of achieving this are:

- The coprocessor receives a number from the S-software and compares it with an encrypted value, returning the result of the comparison.
- The coprocessor receives data from the S-software and returns a jump address.
- The coprocessor verifies that a particular message is sent by the S-software within a prescribed period.
- The coprocessor receives a block of encrypted MC68020 instructions and executes them, placing the results of the execution in the registers and/or the memory of the MC68020.

These methods can be used individually or in combination. The sequence of events that takes place as an item of S-software is executed is as follows:

- Initialization: The pricing information is sent to the coprocessor by the initialization routine when execution of the S-software commences.
- Execution: The check routine in the software body is executed, and execution of the software is aborted if the prescribed protocol is not satisfied.
- Termination: The coprocessor is informed when the S-software has completed execution. This can be implemented using general-purpose coprocessor instructions.

Supporting Files: The payment file keeps track of S-credits that are owed as the result of software usage. Each record in a payment file ordinarily contains three fields: a payer, a receiver, and an amount. The collection of payment files is an essential function in a Super Distribution system in order to make sure that revenues are properly distributed to the providers of services.

The privilege file records a user's privileges with respect to software usage. For example, if a user has purchased a program then the user has the privilege of using it thereafter without paying any additional fees. Some forms of Super Distribution can be realized without having a privilege file.

Super Distribution is equipped with an S-box. The S-box contains a metering program, the Software Usage Monitor (SUM), which enforces the terms set by each software vendor for executing that vendor's products and keeps track of how much the user owes to each vendor. The S-box generates a payment file that contains this information. The fees charged for software usage are measured in units called S-credits. Payment files are encrypted and transmitted to the collection agency.

The collection agencies receive the payment files from users, process those files, and transmit payments to vendors, both the authors of the S-programs and the manufacturers of the S-boxes. Each collection agency has an S-box in its computer. The payment files are decrypted and processed under control of this S-box so as to ensure the integrity of payments to the vendors of S-programs and of the supporting hardware as well.

The clearing house keeps track of funds transfers engendered by Super Distribution. The clearing house can be either a new organization or an existing one, e.g. a credit card company that is prepared to provide the necessary services. The advantage of creating a new organization is that it provides additional degrees of freedom in the systems design. Using an existing organization as the clearing house gives that organization the opportunity to enlarge its market.

Identification numbers (ID's) are essential to this arrangement. Each user, each software vendor, each S-box manufacturer, and each collection agency has a unique ID. In addition, users can also have ID's, either as individuals or as organizations. The ID of a user is usually in a different name space than the ID of that user's machine. User ID's provide a convenient mechanism for establishing credit, since they are associated with the individual or organization who is actually paying for the software. On the other hand, a software distribution system based purely on S-box ID's can provide a high degree of privacy for users.

The super distribution scheme does not provide any absolute guarantee that users will pay what they owe, or even that they will return the necessary payment files to the collection agents. However, the S-box, which contains a real-time clock, will suspend its services (other than transmitting payment files) if the conditions, which are specified by the vendor or the system, are not met. Transmission can be either over a telecommunication link or with a memory card. In the event that a user transmits the payment files but does

not remit the corresponding payment, a collection agent can apply appropriate sanctions. To join the system, a user need only insert an S-box, most likely in the form of a coprocessor chip, into his or her own personal computer. A collection agency can join the system in the same way. The collection agency needs a slightly different form of S-box, one that contains the software for decoding and processing the payment files. Processing at and between the collection agencies and the clearing house can be conveniently handled by a Credit Authorization Terminal (CAT).

3.6.3 Design of the S-Box- The function of the S-box is to execute the Software Usage Monitor in a secure way. The S-box also contains the cryptographic keys that ensure the integrity of the system. These keys are kept secret by storing them within digitally protected modules. An S-box can be added to an existing computer in any of several ways:

1. Attaching it to an I/O port.
2. Connecting it to the bus.
3. Placing it between the CPU and the bus.
4. Placing it on the same chip as the CPU or inside the CPU.

Methods (1) and (2) require no modification to the computer. This method has the disadvantage that it introduces significant overhead in a multi-programming environment. Method (4), though more difficult to implement initially, offers the best performance and the lowest cost.

This connection is best realized by making the S-box a coprocessor and using an existing coprocessor socket. The S-box provides a limited form of protection against viruses, in that programs that are specifically designed to work with the S-box cannot be modified at the user's computer and so cannot be disabled or other wise taken over by a virus. However, the S-box does not protect against programs, distributed through super distribution or otherwise, whose effect is on parts of the system unrelated to the S-box. The workings of the S-box depend on the existence of cryptographic keys within the S-box.

The Software Usage Monitor: the Software Usage Monitor is executed in the S-box. It performs the following functions:

- It monitors the execution of an S-program in order to make sure that it is only executed in the manner intended by its vendor.
- It encrypts and decrypts S-programs and other necessary information.
- It maintains an account of the charges associated with the execution of an S-program.

An S-program can issue instructions to the Software Usage Monitor. These instructions are realized as extended instructions of the CPU.

3.6.4 Design of the Supporting Software and File Structures- Structure of an S-Program- The pricing information for an S-program is contained in its tariff section. When execution of the S-program is initiated, the coprocessor verifies the information in the tariff section and copies the pricing information into the accounting buffer of the S-box. A verification routine is embedded in the body of the S-program. Execution is aborted if the verification routine detects any inconsistency in the tariff section.

The tariff section includes the following:

- Software ID: an identifier unique to the S-program.
- Access control key: A key used by the authentication routine in the S-program body. This key is checked by the coprocessor.
- Encrypting key: A key used to encrypt the machine instructions in the body of the S-program. An encrypted block of instructions is transferred to the coprocessor, where it is decrypted using this key and then executed.
- Charging interval: the length of execution time for which a single S-credit is charged to the user.
- Service pricing schedule: the number of S-credits charged to the user each time a particular software service, e. g. saving a file, has been rendered and successfully completed.
- The amount of memory required within the coprocessor for executing the S-program.

Execution of an S-Program: Execution of an S-program is monitored by the program and the coprocessor working together. The coprocessor halts execution whenever it detects a

protocol violation. For proper protection, the S-software must be structured so that analysis of it by an intruder is not possible. Some methods of achieving this are:

- The coprocessor receives a number from the S-software and compares it with an encrypted value, returning the result of the comparison.
- The coprocessor receives data from the S-software and returns a jump address.
- The coprocessor verifies that a particular message is sent by the S-software within a prescribed period.
- The coprocessor receives a block of encrypted MC68020 instructions and executes them, placing the results of the execution in the registers and/or the memory of the MC68020.

These methods can be used individually or in combination. The sequence of events that takes place as an item of S-software is executed is as follows:

- Initialization: The pricing information is sent to the coprocessor by the initialization routine when execution of the S-software commences.
- Execution: The check routine in the software body is executed, and execution of the software is aborted if the prescribed protocol is not satisfied.
- Termination: The coprocessor is informed when the S-software has completed execution. This can be implemented using general-purpose coprocessor instructions.

Supporting Files: The payment file keeps track of S-credits that are owed as the result of software usage. Each record in a payment file ordinarily contains three fields: a payer, a receiver, and an amount. The collection of payment files is an essential function in a Super Distribution system in order to make sure that revenues are properly distributed to the providers of services.

The privilege file records a user's privileges with respect to software usage. For example, if a user has purchased a program then the user has the privilege of using it thereafter without paying any additional fees. Some forms of Super Distribution can be realized without having a privilege file.

The privilege file grows with time, since information is added to it but is rarely if ever deleted. Moreover it must be preserved indefinitely. It is ordinarily the obligation of a software vendor to provide backup for the portion of the privilege file pertaining to that vendor's software.

If a Super Distribution system provides for trial usage of software either free or at a discount, it must keep records of that usage in a trial usage file. Otherwise a user could use software indefinitely on a trial basis. a. These records need to be backed up, although backup is less critical than it is for the privilege file.

The benefit file keeps track of special benefits granted to the user by various vendors. The benefit file differs from the privilege file in that the information in the benefit file has no direct correspondence to the information in the payment file and is unaffected by what software is used or how much it is used.

The account file is an aggregation of the payment file, the privilege file, the trial usage file, and the benefit file.

A Super Distribution system needs to provide secure backup for its accounting files and for the privilege file in particular. It need not make any special provision for backing up software since a user can back it up by any convenient means or get another copy without cost if the original copy is lost.

The backup requirements for the privilege file depend on whether or not privileges can decrease with time. If not, it suffices to ensure that the process of restoring the privilege file from its backup cannot generate additional privileges. This requirement can be met by associating an ID and a randomly generated validation number with each privilege file and encrypting the file before backing it up. The restoration process checks to make sure that the decrypted validation number is correct and that the backup file is indeed a proper one.

3.6.5 Digitally Protected Modules- Design of the Digitally Protected Module: A protected module (PM) is a container for information that protects that information from attacks intended to read or write it in an unauthorized manner. Protected modules have also been described in the literature as tamper resistant modules or protected processors. Increasingly effective levels of protection are possible:

- A protective method may rely on the ignorance or lack of resources of the attacker. For example, software can be stored in ROM in a form where access is difficult and does not follow the usual conventions of the computer to which it is attached.
- A protective method may rely on keeping a permanent record of attacks rather than on withstanding those attacks. If the proprietor of a protected module can retrieve this record, then attackers will be dissuaded by the knowledge that their activities will be discovered.
- A protective method may be designed to defend a protected module even when the proprietor of the module has no access to it, either physically or logically. Since no physical device can be expected to resist destruction indefinitely when the attacker has sufficient time and resources, a device using methods of this kind must be able to detect attacks and erase its contents before the attacker can retrieve them. Then an attacker is simply left with a broken module and has gained nothing.
- A protective method may be designed to defend a protected module even from attacks by its manufacturer. Since in some cases a protected module may contain information that the module's manufacturer is not authorized to access, this case is also of interest.

3.6.6 Operation of the Digitally Protected Module (DPM)- The protective mechanism consists of a random access memory (RAM) containing a great number of closely spaced microscopic one-bit detectors. The DPM contains one or more layers of these detectors, packed tightly so as not to leave enough room for penetration between them. Layers of detectors can be staggered so as to increase the effective density. The detectors are individually addressable, and each one can be read or written. A possible attack is indicated when any detector is found to have lost its normal operating functions.

The testing method is simple: write a random value into the detector and read it back. Then write the complement of that value into the detector and read it back. The element is considered to be working if and only if the retrieved values agree with the written

values. If the values do not agree, a possible attack is indicated. We call this event an exception.

If an exception occurs, the microprocessor examines the faulty detector several more times. Repeating the test provides protection against random transient failures such as might be caused by alpha rays. The probability of such a failure and the consequent change of state are proportional to the time between writing and reading. If a detector is read immediately after it is written, the probability becomes very small. In some environments the probability of transient errors may be low enough so that transient errors can be disregarded altogether.

If an exception is judged not to be a transient error, then the DPM can test other detectors in the physical vicinity of the erroneous one. The number of faulty elements can then be compared to a threshold. If that threshold is exceeded, we assume that the DPM has been attacked. The threshold test reduces the chance of a false alarm.

Techniques that apply to RAM used in other applications can also be used for the DPM. For instance, the microprocessor can be provided with a list, resembling the list of bad tracks on a disk that indicates elements known to be faulty. Similarly, it is possible to read and write blocks of several bits in parallel instead of operating on a single bit with each test. However, some economies are possible for the DPM that are not possible in general. For instance it is not necessary to refresh the memory because it can hold its state for at least ten milliseconds without being refreshed. That interval is much greater than the interval between writing and reading a bit. Such economies can help to reduce the cost of manufacturing the DPM.

The design of any type of protected module must address an inherent conflict. In order to make the device more likely to detect an attack it must be made more sensitive; but the more sensitive it is, the greater the likelihood of a false alarm. The more information that can be gotten about a presumed attack, the easier it is to resolve this conflict.

We next consider various kinds of potential attacks on a digitally protected module and the protection against them:

- An attacker might attempt to separate the detecting layer and the protected region. Such an attack can be detected by monitoring the continuity of the inter layer wires.

- An attacker might aim at the detection mechanism itself rather than at the information being protected. This strategy can be foiled by placing the detection mechanism in the protected region.
- An attacker might attempt to manipulate the wires that connect the circuitry to the outside world. Such attacks can be prevented by a structure where the wires pass through the detecting layer. It is possible to reuse a digitally protected module that has erased its cryptographic keys in response to an apparent attack, but caution is necessary in order to guard against Trojan horses or other attempts at subversion of the logical protection.

The protective capability of any protected module cannot be regarded as permanent. Once a module has been installed the hardware implementation of its defensive technology is fixed, even though the technology of attack advances continually. With time it becomes easier and easier to find ways of bypassing or otherwise neutralizing whatever defenses the module utilizes. Thus the useful life of a protected module is determined by the progress of the technology of attacks.

3.6.7 Realization of Digitally Protected Modules- Three-dimensional large-scale integrated circuits, possibly manufactured using chip-bonding methods, provide an excellent method of realizing digitally protected modules.

Layer 1 consists of a protected region where the information to be protected is stored. The memory for the information can be implemented using technology at the level of 1 Mbit DRAMs. The protected region either has its own power supply or uses an external power supply. It is possible that there is more than one power supply. The protected region also contains circuitry whose function is to erase the information if an attack is detected. The erasure is accomplished by cutting off the power to Layer 1.

An attack might involve making holes in the device, etching it, or subjecting it to electro-optical reading. Layer 2 is made of devices and wirings designed to detect these kinds of attacks.

The wires between the two layers serve two purposes: to provide signal paths between the circuitry in the layers and to provide signal paths from the protected layer to the outside world. These wires are connected to small pads on the inner surface of each layer. They

cover the entire perimeter of the DPM. Any disruption of their signals will have the same effect as a disturbance to the detectors, so they provide an additional level of protection to the device.

The DPM can be encapsulated using an appropriate resin, preferably one that is not transparent. By mixing a substance such as alumina with the resin we can increase the resistance of the device to attacks using mechanical or chemical methods, or even attacks using lasers or plasmas. Simpler measures may, however, suffice for some environments. The RAM in Layer 1 depends on a continuous supply of power to maintain its state. If the protective logic of the DPM concludes that an attack has taken place, it grounds the power supply and erases the secret information in the protected region. It is pointless for an attacker to disrupt the power supply because the effect of doing that will be to erase the protected information.

3.6.8 Detection of Attacks- The DPM has several different memories, and these memories have distinct purposes. The memory in Layer 1, the protected region, stores the information that is to be protected. The memory in Layer 2 is devoted to detecting attacks.

The order in which the detectors are tested is arbitrary, but the total test duration must be kept short. The detectors can be tested individually or in groups. Group testing helps to keep the testing interval short because the detectors in a group can be tested in parallel. There are at least three levels of protection that can be provided for the information in the protected region:

- The lowest level of protection is provided by using the same physical structure in every DPM. The protected information is represented by the presence or absence of electrical charges at particular physical locations in the structure.
- The second level of protection is provided by storing the protected information in a way that is different for each individual DPM. One way of doing this is as follows: each module contains a random number, which is generated independently for each module. The information is encoded in such a way that it can be decoded by using the random number. Full cryptographic protection is not necessary for this encoding; even such a simple method as exclusive ordering the stored information with the random number is sufficient.

change even one bit of the message without altering the signature. It is therefore clear that for parties with conflicting interests, a digital signature should offer more protection against fraud than today's analog signature.

Digital signature schemes are usually classified into one of the two categories: true signatures and arbitrated signatures. In a true signature scheme, signed messages produced by the sender S are directly transmitted to the receiver R, who verifies their validity and authenticity. A dispute arises if S denies R's claim that a signed message in R's possession was actually sent by S. In such a case, a judge may be called in to decide who is at fault. In an arbitrated signature scheme, on the other hand, all signed messages are transmitted from S to R via an arbitrator A who serves as a witness. The presence of A reduces the occurrence of disputes.

Digital signatures enable the recipient of information to verify the authenticity of the information's origin, and also verify that the information is intact. Thus, public key digital signatures provide authentication and data integrity. A digital signature also provides non-repudiation, which means that it prevents the sender from claiming that he or she did not actually send the information.

Security is always a concern with digital signature technology. A digital signature is considered superior to a handwritten signature in that it attests to the contents of a message as well as to the identity of the signer. As long as a secure hash function is used, there is almost no chance of taking someone's signature from one document and attaching it to another, or of altering a signed message in any way. The slightest change in a signed document will cause the digital signature verification process to fail. Thus, public key authentication allows people to check the integrity of signed documents. If signature verification fails, however, it will generally be difficult to determine whether there was an attempted forgery or simply a transmission error.

Public-Key Cryptography is used in conjunction with digital signatures in ensuring security. The primary advantage of public-key cryptography over private-key cryptography is increased security and convenience. Private keys would never need to be transmitted or revealed to anyone.

3.7.3 Digital Signature Technology Terms- One of the inhibitors to the increasing use of electronic commercial transactions has been the concern for the risks of forgery over

cover the entire perimeter of the DPM. Any disruption of their signals will have the same effect as a disturbance to the detectors, so they provide an additional level of protection to the device.

The DPM can be encapsulated using an appropriate resin, preferably one that is not transparent. By mixing a substance such as alumina with the resin we can increase the resistance of the device to attacks using mechanical or chemical methods, or even attacks using lasers or plasmas. Simpler measures may, however, suffice for some environments. The RAM in Layer 1 depends on a continuous supply of power to maintain its state. If the protective logic of the DPM concludes that an attack has taken place, it grounds the power supply and erases the secret information in the protected region. It is pointless for an attacker to disrupt the power supply because the effect of doing that will be to erase the protected information.

3.6.8 Detection of Attacks- The DPM has several different memories, and these memories have distinct purposes. The memory in Layer 1, the protected region, stores the information that is to be protected. The memory in Layer 2 is devoted to detecting attacks.

The order in which the detectors are tested is arbitrary, but the total test duration must be kept short. The detectors can be tested individually or in groups. Group testing helps to keep the testing interval short because the detectors in a group can be tested in parallel. There are at least three levels of protection that can be provided for the information in the protected region:

- The lowest level of protection is provided by using the same physical structure in every DPM. The protected information is represented by the presence or absence of electrical charges at particular physical locations in the structure.
- The second level of protection is provided by storing the protected information in a way that is different for each individual DPM. One way of doing this is as follows: each module contains a random number, which is generated independently for each module. The information is encoded in such a way that it can be decoded by using the random number. Full cryptographic protection is not necessary for this encoding; even such a simple method as exclusive ordering the stored information with the random number is sufficient.

- The third level of protection is provided in a similar way as the second, but the random number and the decoding function change with time and are not always stored in the same location.

The choice among these levels should be made according to the defensive strength that is required and the cost that is acceptable. In some applications a less effective method of protection may suffice. The protected information is kept on a mask ROM, and the chip containing the ROM is encapsulated in resin.

The Protected Region: Layer 1 of the DPM is the protected region. Layer 2 of the DPM covers the protected region. The purpose of the detectors is to insure that nobody can physically invade the protected region without destroying one of the detectors. However, if it is known that physical invasion will not result in the theft of the protected information, the invasion can be tolerated.

At very low temperatures, a RAM may hold an electrical charge for some length of time even without power. Therefore the protected region can contain a monitoring circuit that will erase the secret information should the temperature drop below a specified threshold. The DPM requires a continual supply of electrical power, whether it is constructed using RAM or EPROM. Once the DPM has been loaded with its protected information power is needed for the memory that contains the secret information, the addressing logic, and the testing circuitry.

3.7 Digital Signatures [13, 14]

3.7.1 Overview- Security is, perhaps, the most important and necessary assurance requirement to consider when deploying a system. Because computing systems perform widespread business and mission-critical functionality each day, the cost associated with a breach of security is also increasing. Furthermore, because security problems are human induced problems, the range of possible problems is as limitless as the human mind. Security, thus, becomes a crucial problem to solve.

Our aim is to provide such an application, which aims at providing a secure way of sending messages. This is achieved through the use of Digital Signatures. It is a secure form of transacting. Contracts, letters, images, texts etc. may be digitally signed and sent electronically in seconds. This application has been made intelligent enough to send the

messages without the danger of being breached by any intruder. This application uses cryptographic technique. We generate a key-pair consisting of public key and private key. The private key is used to sign the message generating a signature and the public key is used to verify the signature generated. In between, we create a hash code of the message and send it along with the signature to make sure that the message is not modified in the process of communication. A digital signature is basically a value that is computed from a sequence of bytes using a secret key. It indicates that the person who holds the secret key has verified that the contents of the message are correct and authentic.

3.7.2 Introduction- Recent advances in telecommunications and computer networking have brought us into the era of electronic mail. By providing rapid and economic channels for the dissemination of data, electronic mail systems are certain to significantly reduce our reliance on paper as the major medium for transactions. It is also clear that with the widespread implementation and use of such systems, senders and receivers of sensitive or valuable information will require secure means for validating and authenticating the electronic messages they exchange. Validation and authentication refer to the methods of certifying the contents of a message and its originator, respectively. Both functions can usually be achieved through the use of a digital signature, which is appended to every message. A digital signature differs from an analog signature (a line drawn with a pen on paper) in two important ways:

- No matter how complicated an analog signature is, a forger intent on committing fraud will eventually be able to duplicate it. A digital signature, on the other hand, should by definition be inimitable.
- A person's analog signature is constant; it is same on all documents signed by that person. By contrast, digital signatures must be different for every message.

From this it follows that a digital signature is a message-dependent quantity that can be computed only by the sender of the message on the basis of some private information. It allows authentication of messages by guaranteeing that no one can forge the sender's signature and the sender cannot deny a message he sent. In addition, validation is also possible because the receiver can verify that no one tampered with the message while it was on its way to him and because the sender is sure that the receiver will not be able to

change even one bit of the message without altering the signature. It is therefore clear that for parties with conflicting interests, a digital signature should offer more protection against fraud than today's analog signature.

Digital signature schemes are usually classified into one of the two categories: true signatures and arbitrated signatures. In a true signature scheme, signed messages produced by the sender S are directly transmitted to the receiver R, who verifies their validity and authenticity. A dispute arises if S denies R's claim that a signed message in R's possession was actually sent by S. In such a case, a judge may be called in to decide who is at fault. In an arbitrated signature scheme, on the other hand, all signed messages are transmitted from S to R via an arbitrator A who serves as a witness. The presence of A reduces the occurrence of disputes.

Digital signatures enable the recipient of information to verify the authenticity of the information's origin, and also verify that the information is intact. Thus, public key digital signatures provide authentication and data integrity. A digital signature also provides non-repudiation, which means that it prevents the sender from claiming that he or she did not actually send the information.

Security is always a concern with digital signature technology. A digital signature is considered superior to a handwritten signature in that it attests to the contents of a message as well as to the identity of the signer. As long as a secure hash function is used, there is almost no chance of taking someone's signature from one document and attaching it to another, or of altering a signed message in any way. The slightest change in a signed document will cause the digital signature verification process to fail. Thus, public key authentication allows people to check the integrity of signed documents. If signature verification fails, however, it will generally be difficult to determine whether there was an attempted forgery or simply a transmission error.

Public-Key Cryptography is used in conjunction with digital signatures in ensuring security. The primary advantage of public-key cryptography over private-key cryptography is increased security and convenience. Private keys would never need to be transmitted or revealed to anyone.

3.7.3 Digital Signature Technology Terms- One of the inhibitors to the increasing use of electronic commercial transactions has been the concern for the risks of forgery over

unsecured networks. This focus has brought about the need for a reliable, cost-effective way to replace a handwritten signature with a digital signature. Like a handwritten signature, a digital signature can be used to identify and authenticate the originator of the information. A digital signature can also be used to verify that information has not been altered after it is signed. The following are terms used with digital signature technology:

Electronic signature: an electronic signature scheme can range from the simplest Personal Identification Number (PIN) to the actual burning-in of a bitmap of the user's signature on a document. Electronic signatures have no way of verifying whether a document has been altered since it was signed. In other words, electronic signature technology does not provide any kind of signer or document authentication. Since electronic signatures are not secure, they have limited legal acceptance.

Digital signature: a digital signature is legally more acceptable than an electronic signature, since it offers both signer and document authentication. Signer authentication is the capability to identify the person who digitally signed the document. The implementation of this technology should be such that any unauthorized person will find it difficult to forge the digital signature. Document authentication ensures that the document or transaction (or the signature) cannot be easily altered.

Digital signatures are created and verified by the use of two different keys. One key is used to create the signature, and the other key is used for verifying the signature. The key used for creating the signature is called the private key, and the key used for verifying is called the public key.

Cryptography: is known as keeping communications private. It also deals with the terms encryption, decryption and authentication. Encryption is the transformation of information into an unreadable form. Its purpose is to ensure privacy by keeping the information hidden from anyone for whom it is not intended, even those who can see the encrypted data. Decryption is the reverse of encryption; it is the transformation of encrypted data back into some intelligible form.

Encryption and decryption require the use of some secret information, usually referred to as a key. Depending on the encryption mechanism used, the same key might be used for both encryption and decryption, while for other mechanisms, the keys used for encryption and decryption might be different.

A digital signature binds a document to the possessor of a particular key, while a digital time-stamp binds a document to its creation at a particular time. There are two types of cryptography, secret-key and public-key.

Secret-Key Cryptography: the traditional type is based on the sender and receiver of a message knowing and using the same secret key: the sender uses the secret key to encrypt the message, and the receiver uses the same secret key to decrypt the message. This method is known as secret-key or symmetric cryptography. The main problem with this method is getting the sender and receiver to agree on the secret key without anyone else finding out. If they are in different physical locations, they must trust a courier, or a phone system, or some other transmission medium to prevent the disclosure of the secret key being communicated. Anyone who overhears or intercepts the key in transit, can later read, modify, and forge all messages encrypted or authenticated using that key.

Public-Key Cryptography: was introduced in 1976 in order to solve the key management problem with the secret-key method. This term deals with each person having a pair of keys, one called the public key and the other called the private key. Each person's public key is published while the private key is kept secret. The need for the sender and receiver to share secret information is eliminated. Communications involve only the public keys, and no private key is ever transmitted or shared. This allows communication over channels that are not completely secure. The only requirement for this is that the public keys are associated with their users in a trusted (authenticated) manner (for instance, in a trusted directory). Anyone can send a confidential message by just using public information, but the message can only be decrypted with a private key, which is the sole possession of the intended recipient. Furthermore, public-key cryptography can be used not only for privacy (encryption), but also for authentication (digital signatures). If we are using a secret-key or private-key cryptography or possibly both, then someone and/or entity must maintain the keys. This introduces the term Key Management.

Key Management: deals with the generation, transmission, storage, and destruction of keys. Any type of cryptography system must deal with key management. The use of secret-key cryptography often has difficulty providing secure key management, especially in open systems with a large number of users. In addition to managing the

keys, sometimes the identities of the keys must be authenticated and a Certificate Authority is necessary.

Certificate Authority: the certificate authority is an individual, organization, or agency, public or private that acts as a notary to authenticate the identity of users of a public-key encryption. The Certificate Authority issues, manages, and revokes digital certificates and vouches for the identities of the end users for whom they are issued.

A Certificate Authority is used to associate a pair of keys with a person (or a company). Further, the Certificate Authority is responsible for publishing the public keys in a directory. The Certificate Authority is also responsible for maintenance functions associated with the keys, such as, revoking keys that have been compromised, and also renewing expired keys

In today's world of paper documents, some documents are required to be notarized. In the future, Certificate Authorities will serve a role similar to that of a notary for electronic transactions. Parties to electronic transactions need a common method of accreditation of Certification.

Digital Certificate Authorities: certification authority addresses the uncertainty of an individual or an enterprise identity, particularly when there is no previous relationship between that entity and others in an open network environment

A digital certificate (or digital ID as it is sometimes called) is an electronic ID file, operating like a driver's license or passport that the Certificate Authority attests that bearers of the certificates are actually who or what they claim to be. The digital certificate acts like an electronic envelope in which the public key travels. This electronic ID file verifies the connection between the public key and the owner. The digital certificate is issued by a Certificate Authority and signed with that Certificate Authority's private key, authenticating the public key. The digital certificate typically includes:

- Public key and owner's name.
- Certifying Authority issuing the key.
- Serial number.
- Digital signature of Certificate Authority, signed using the Certificate Authority's private key.
- Other optional identifying information.

The digital certificates cannot be altered, with the Certificate Authority making it tamper-proof through the strength of the authentication from their digital signature.

Digital Certificate versus Digital Signature: the digital certificate is sometimes confused with a digital signature. The digital certificate authenticates a public key and identifies the owner of that key. The digital certificate answers the question who is authorized to use the key.

The digital signature provides a means to electronically replace a handwritten signature. The digital signature is used to authenticate a specific document or transaction, and the entity (person or company) that created/sent the transaction.

3.7.4 Working of Digital Signature Technology- Digital signatures are created and verified by cryptography. Digital signatures use public-key cryptography, one key for creating a digital signature and another key for verifying a digital signature. These two keys (which form a key pair) are collectively termed as asymmetric cryptosystem.

The complementary keys of an asymmetric cryptosystem for digital signatures are termed the private key and the public key. The private key, known only to the sender and used to create the digital signature, and the public key, which is ordinarily more widely known and is used by a relying party to authenticate the digital signature. Although many people may know the public key of a given sender (signer) and use it to verify that sender's signature, they cannot discover that sender's private key and use it to forge digital signatures.

The sender accomplishes the process of creating a digital signature. The receiver of the digital signature performs the verification of the digital signature. The following example: writing and sending a check illustrates how digital signature technology works.

Digital Signature Creation:

Sign- To begin the process, a check must be created. In order to create a digital signature with the check, a process known as hash function must occur. A hash function is a mathematical algorithm, which creates a digital representation or fingerprint in the form of a hash result or message digest. The hash function generally consists of a standard length that is usually much smaller than the message but nevertheless substantially unique to it. Hash functions ensure that there has been no modification to the check (message) since it was digitally signed.

The next step is to encrypt the check and signature. The sender's digital signature software transforms the hash result into a digital signature using the sender's private key. The resulting digital signature is thus unique to both the message and the private key used to create it.

Typically, a digital signature is appended to its message and stored or transmitted with its message. However, it may also be sent or stored as a separate data element, so long as it maintains a reliable association with its message. Since a digital signature is unique to its message, it is useless if wholly disassociated from its message.

Seal- Since public-key algorithms can be slow to transmit, the next step is to encrypt this information. The check is encrypted with a fast symmetric key (uniquely generated for this occasion) and then, the symmetric key is encrypted with the receiver's public key. Now only the private key of the receiver can recover the symmetric key, and thus decrypt the check. A digital version of the envelope has been created.

Deliver- At this point, the digital envelope is electronically sent to the receiver and the verification process begins.

Digital Signature Verification:

Accept- The encrypted digital envelope arrives at the destination.

Open- The receiver of the check decrypts the one-time symmetric key by using the receiver's private key. Then the check is decrypted using the one-time symmetric key. Once this has been completed, the verification process begins.

Verify- Verification of a digital signature is accomplished by computing a new hash result of the original message. Then, using the sender's public key and the new hash result, the verifier checks:

- Whether the digital signature was created using the corresponding private key; and
- Whether the newly computed hash result matches the original hash result.

The software will confirm the digital signature as verified:

- The sender's private key was used to digitally sign the message and
- The message was unaltered.

If the verification cannot be made, the software will identify that verification has failed.

3.7.5 Purposes of Digital Signatures- The processes of creating a digital signature and verifying it accomplishes the essential effects that a handwritten signature does today for many legal purposes:

- **Signer authentication:** If a public and private key is associated with an identified signer, the digital signature attributes the message to the signer. The digital signature cannot be forged, unless the signer loses control of the private key, such as by divulging it or losing the media or device in which it is contained.
- **Message authentication:** The digital signature also identifies the signed message, typically with far greater certainty and precision than paper signatures. Verification reveals any tampering, since the comparison of the hash results (one made at signing and the other made at verifying) shows whether the message is the same as when signed.
- **Non-Repudiation:** Creating a digital signature requires the signer to use the signer's private key. This act can alert the signer to the fact that they are consummating a transaction with legal consequences. This can decrease the chances of litigation later on.
- **Integrity:** The processes of creating and verifying a digital signature provide a high level of assurance that the digital signature is genuinely the signer's. Compared to paper methods, such as checking signature cards, methods that are tedious and labor-intensive, digital signatures yield a high degree of assurance without adding greatly to the resources required for processing.

3.7.6 Applications of Digital Signatures- The function of digital signatures is the authentication and validation of electronic messages exchanged over the channels of telecommunication network. Typical applications requiring such signatures include business transactions and military orders as well as a variety of cases involving contracts or agreements between people and institutions. For example, an electronic funds transfer system using digital signatures consist of a special card for use in an EFT terminal, such as an automated teller machine, being issued by organization Y. The card is accepted by organization Z, which manages the terminals. Every transaction request by a cardholder is routed from the terminal to Z and subsequently to Y for approval. The request is honored by Z only if Y's response is positive and signed.

Another interesting instance, involves two countries that have agreed to stop all underground testing of nuclear weapons. To verify that the agreement is being respected, each country places on other country's territory seismic observatories whose output is sufficient to determine whether a test has taken place. All observatories in one country use a public-key scheme with two procedures P and $1/P$ to sign their messages M . The first of these two procedures is secret and physically protected inside the observatory; the second is publicly known. To guarantee that P is unknown even to the country owning the observatory, the two procedures are generated by a third and neutral body, such as the United Nations. Periodically, each observatory outputs a pair $(M, P(M))$, which it transmits to its owner. In that way, a country can verify that every message it receives is authentic and valid. In addition, it has access to all the information leaving its territory and thus can make sure that the other country's observatories are not used for spying. Disputes are easily settled, since neither country can change the output of an observatory to conceal the occurrence of a nuclear test or, alternatively, to falsely accuse the other country of conducting one.

Digital signatures can also be used in environments where the interests of the parties involved are not necessarily conflicting. Indeed, in a number of useful applications of the concept disputes do not arise. Two such applications pertaining to software protection include:

- A central source distributes software updates to the individual nodes of a network. By signing its messages and requiring each node to check the signature, the central source is guaranteed that no node will execute wrong software by mistake or accept an update received from a malicious source.
- It concerns operating system security. Here, it is required to protect the operating system against programs attempting to perform tasks without proper authorization.

One way around this problem is to require each program to bear the signature of the original programmer and those of various supervisory authorities. The hardware would then refuse to run a program that is not properly signed.

3.7.7 A Critical Review- Digital Signature technology is still considered to be evolving and immature. To reduce costs and increase productivity, many government agencies are

transforming paper-based systems into automated electronic systems. This trend has created the need for a reliable, cost-effective way to reduce a handwritten signature with a digital signature.

As mentioned earlier, Digital Signature technology is still considered to be evolving and immature. Hence, there is ample scope for further improvement. In the True Signature scheme, message is directly passed from the sender to the receiver. One may also use the Arbitrated Signature scheme in which message is passed from the sender to the receiver via an arbiter. The advantage of this scheme is that if the receiver do not accepts that he has received a message, even if he has received one, the sender can settle the dispute with the help of the arbiter, to whom he has sent the message first. To make message passing more secure, we can further encrypt the hash code generated from Message Digest Algorithm (MD5) using either secret-key cryptography or public-key cryptography. The first scheme can be implemented using Digital Encryption Standard (DES), DESede, Blowfish or Rivest Ciphers. The later scheme can be implemented via Rivest, Shamir, Adleman (RSA) and ElGamal algorithm. One of the most powerful and secure features of Digital Signature is that it generates different signatures each time the sender sends a message. It is almost impossible to realize the fact that this scheme generates different signature even if the sender sends the same message more than one time. As such, it is very difficult to forge the signature even if an intruder verifies the signature.

Yet it is a sad realization that every thing has its limitation. The Digital Signature scheme is no exception. The signature is liable to be verified by an intruder if he gains access to the digital envelope consisting of the sender's public key and the signature. However, it is a different proposition altogether that even if he verifies the signature, he will not be able to make changes in it. A major limitation is that the sender's private key must be kept secret; otherwise it will be relatively easy to decrypt the message. Yet another limitation is that the sender may claim that he has sent a message, even if he has not sent any message. This will require the need of a trusted third party to resolve the dispute. A final limitation is that this scheme requires that the sender must send all the information (hash code, message and signature) very carefully because if even there is a change in a single bit, it will lead to an entirely different signature being created and the receiver will assign the message as unauthentic.

Signature Generation

Signature Verification

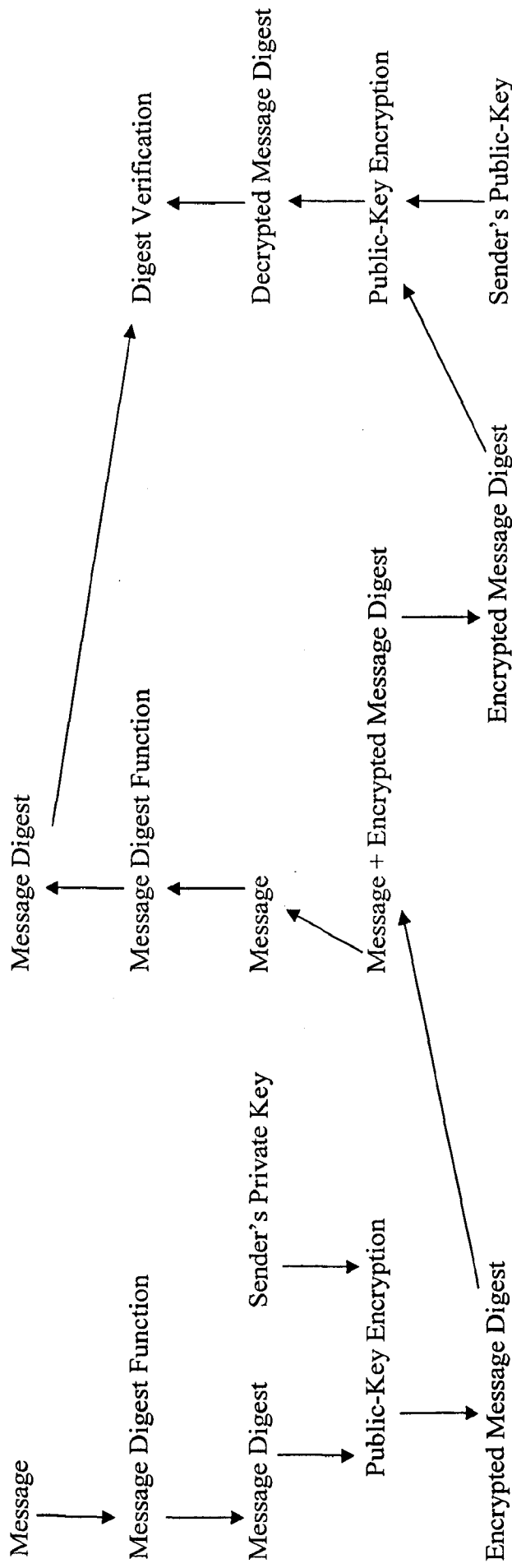


Fig 3.8: Working of Digital Signatures

3.8 Java [15, 16]

Java was conceived by James Gosling, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan at Sun Microsystems, Inc. in 1991. The environmental change that prompted Java was the need for platform independent programs destined for distribution on the internet. Java enhances and refines the object-oriented paradigm used by C++.

5.7.1 OOP in Java-

Java uses the three mechanisms of object-oriented model: **encapsulation, inheritance and polymorphism**. Class Encapsulation links data with the code that manipulates it. Encapsulation provides another important attribute: access control. Classes and packages are means of encapsulating and containing the name space and scope of variables and methods. Packages act as the container for classes and other subordinate packages. Classes act as containers for data and code. All the classes used are included in the package File tracking. Inheritance is the process by which one object acquires the properties of another object. This supports the concept of hierarchical classification. Polymorphism is a feature that allows one interface to be used for a general class of action. The specific action is determined by the exact nature of the situation.

Java can be used to create two types of programs: applications and applets. An application is a program that runs on your computer, under the operating system of that computer. An applet is an application designed to be transmitted over the internet and executed by a Java-Compatible Web Browser. An applet is actually a tiny Java program dynamically loaded across the network, just like an image, sound file, or video chip. The importance difference is an intelligent program, not just an animation, or media file. In other words applet is a program that can react to user input and dynamically change, not just run the same animation or sound over and over.

3.9 Linux [17]

Linus Trovalds of the University of Helsinki in Finland originally created Linux. Linux is based on a small PC-based implementation of UNIX called minix. Near the end of 1991, Linux was first made public.

In November of that same year, version 0.1 was released. A month later, in December version 0.11 was released. Linus made the source code freely available and encouraged others to develop it further. They did. Linux continues to be developed

today by a worldwide team, led by Linus, over the Internet. The current stable version of Linux is version 2.0. Linux uses no code from AT&T or any other proprietary source. Much of the software is developed by the Free Software Foundation's GNU project. Linux, therefore, is very inexpensive; as a matter of fact, it is free.

3.9.1 Advantages of Linux- The advantages of Linux are as follows:

- **Full Multitasking:** Multiple tasks can be accomplished, and multiple devices can be accessed at the same time.
- **Virtual Memory:** Linux uses a portion of your hard drive as virtual memory, which increases the efficiency of your system by keeping active processes in RAM and placing less frequently used portions of memory on disk.
- **The X Window System:** The X Window System is a graphics system for UNIX machines. This powerful interface supports many applications and is the standard interface for the industry.
- **Built-in Networking Support:** By connecting your system with an Ethernet card or over a modem to another system, you can access the Internet.
- **Shared Libraries:** Each application, instead of keeping its own copy of software, shares a common library of subroutines it can call at run time. This saves a lot of hard drive space on your system.
- **GNU Software Support:** Linux can run a wide range of free software available through the GNU project. This software includes everything from application development to system administration and even games.

3.9.2 Reasons to use Red Hat Linux- The main reasons to use Linux are:

- It is based on Linux 2.0x.
- Red Hat Package Manager is included.
- Disk Druid, new version of Red Hat, is Red Hat's disk management utility. In particular Disk Druid enables you to add and delete partitions through a GUI interface.
- Red Hat leads the industry in providing the most up to date security features.
- Red Hat depends on the open development model Linus started with.

DESIGN AND IMPLEMENTATION

4.1 Security for the Digital Library [18, 19, 20, 21]

4.1.1 Introduction- Just like the physical material, the digital material of the library is valuable. Access to it must be guarded and its well-being must be ensured. Security for the digitized material must be provided in the form of restricted access to the computers also that hold the material. Access security at least, must be allowed for in the digital library. The library may be freely accessible to all, but certain sections must be protected. Specialized software now exists (such as EduLib's STOPit system) for use on library workstations, which allows the functionality of those machines to be linked to individual users via their library card or some form of key.

4.1.2 Secure Channels of Communication- Various methods exist to ensure the security of material delivered over the Internet. This covers both the prevention of unauthorized access if the material goes astray, and the insurance that the material does not go astray in the first place.

The standard file delivery mechanisms such as via HTTP or FTP protocols have built-in mechanisms to ensure that all packets of your material are delivered and reconstructed correctly.

These mechanisms combined with the robustness and security of the underlying TCP/IP delivery infrastructure should ensure that the files reach their intended recipient in pristine order.

However, files can be snooped on in transit and copies made. This is where the encryption and secure delivery mechanisms come in. Even if a copy of your material falls in the wrong hands, they should not be able to read it. Signed delivery and digital certificates, which authenticate the parties at both ends of a transaction, are mechanisms we can put in place. More recent mechanisms such as Microsoft's Hailstorm/Wallet and the Liberty Alliance where a third party vouches for both the parties can be used, especially for paid for transactions.

Recent encryption techniques developed for the music industry claim that they are encrypted in such a way that files cannot be unencrypted without the key, and cannot even be copied. We can refer to RIAA (Recording Industry Association of America)

web site for the latest in protecting files in transit, as they are publishing this development.

4.2 Document Protection

We focus on the mechanisms necessary to put security architecture for digital libraries in place. This includes protection of the content, feasibility of payment and assertion of copy-and-usage rights. While current research in secure Web Technology (like SSH, HTTP or SSL) focuses on the protection of the communication channel, proposals for protecting digital content usually rely on some sort of secure container to realize the functions mentioned above. IBM Cryptolope Technology can be considered as an example for secure containers, but we will discuss the concept underlying both approaches.

4.2.1 Introduction- The acceptance of the Internet as an infrastructure for electronic commerce, over the last years, triggered the development of several extensions of its underlying protocols in order to integrate or enhance security mechanisms. The shortcomings of the Internet Protocols in this area are well known. The new requirements resulting from electronic commerce applications together with a strong commercial interest in secure Internet technology lead to the development of new concepts and the adaptation of existing ideas in order to facilitate secure electronic commerce. All of these mechanisms such as IPSP, SSL, SSH or S/MIME make use of cryptographic operations in order to realize specific security services. In different ways, these protocols attempt to secure the communication channel between two parties communicating over the Internet.

Digital libraries as intermediaries in the distribution of electronic goods and the associated usage rights need additional mechanism in order to fulfill their role as custodians of electronic documents and the terms and conditions of their usage. The tasks of a digital library are not limited by establishing a secure communication channel, but extended to the protection of electronic documents even after the transfer. Most of the solutions proposed in this area are based on the use of cryptographically secured containers.

4.2.2 Description- In the first section, we explain the security requirements for digital library. In the second section, we give a brief introduction in cryptographic operations, since they are the basis of all currently available Internet security solutions. In the third section, we focus on security architectures for a digital library.

In the fourth section, we consider a case study of IBM Cryptolope Technology as an example for demonstrating the properties of cryptographically secured containers. We conclude with a comparison of the two approaches showing how the latter is more appropriate to the digital library's requirements.

Section-I

Security Requirements for the Digital Library: The Digital Library has to face a wider variety of threats than its conventional counterpart.

At the beginning of each transaction, both the publisher and the reader will want to make sure that their respective partner really is the one he claims to be, i.e. they have to authenticate each other. Likewise, both publisher and the reader will require that the content is authentic, i.e. has been really published by the given publisher, and that it is intact, which means that nobody has added to or deleted from the package. To be secure from the eavesdropper, the content never should be transmitted and stored in readable format.

These authenticity and integrity requirements are not only applicable to the content, but also to contract offer which may accompany the content and which states the terms and conditions under which a reader may use the content. The publisher may want to prove that the reader has accepted the terms, and the reader may want to have a signed copy of what he is entitled to do.

Once a reader accepts the contract offer, both parties have to adhere to the terms and conditions. This may include the payment and the compliance to the copyright from the publisher's perspective, and the right to use the information from the reader's perspective.

Digital Library content items can be very large. In this case, it is often useful to decouple the distribution of information and its licensing by distributing encrypted bulk data and controlling the release of content through the key management. Then the distribution can take place over a cheap broadcast channel, and access to the content can be controlled via a separate non-broadcast channel. This separation channel is basically a key exchange between a user's personal computer and a dedicated royalty/license clearing center. This facilitates the distribution of the actual content data over a variety of media like the Internet, digital cable TV, satellite broadcast or CD-ROM publishing. This concept called 'Super Distribution', gives the publisher a very flexible way to use the most appropriate distribution method.

Section-II

Internet Security Mechanisms: Most of the current security systems are based on cryptographic algorithms. We will briefly explain the characteristics of some cryptographic algorithms and their use in secure electronic commerce applications. We will then show how these mechanisms are used to provide secured communication channels over the Internet.

Building Blocks of Security Solutions: Cryptographic algorithms can generally be divided into two categories. In the case of symmetric cryptographic algorithms, the same key is used for encryption and decryption. Commonly found examples for this kind of algorithms are DES, SEAL or RC4.

Asymmetric cryptographic algorithms (or public-key algorithms) are characterized by the fact that the key used for encryption is different than the key used for decryption. Public-Key-operations are performed with key pairs: every message, which is encrypted with one of these keys, can only be decrypted using the other one. If one of these keys is kept secret and the other one is published, asymmetric cryptographic algorithms serve two purposes:

- Data, which is encrypted with the recipient's public key, can only be decrypted with the corresponding private key.
- Data, which is encrypted with the sender's private key, can be decrypted by everybody who is in possession of a copy of the corresponding public key. This property serves as the foundation for digital signatures: the sender signs with the private key and the signature can be verified with the corresponding public key.

The most prominent example for an asymmetric cryptographic algorithm being used for both encryption and digital signatures is RSA; another one, which can only be used for digital signatures, is the Digital Signature Algorithm (DSA). Since these asymmetric algorithms are computationally very demanding, they are never used for the encryption of large amounts of data. Today's implementation typically makes use of the following two concepts:

- Large amounts of data are symmetrically encrypted with a random key. This so called session key is asymmetrically encrypted using the recipient's public key and can thus be safely transmitted. In connection-oriented protocols, this phase is referred to as key exchange.

- In the case of digital signatures, one-way hash function are used to condense the data to be signed to a fixed length hash value, which is then encrypted with the sender's private key. Examples for one-way hash functions are MD5 or SHA. If the hash function is not only based on the data but also on a secret key, it is called a Message Authentication Code (MAC).

The uses of asymmetric algorithms require the existence of trusted public keys. While this relationship of trust could be established for every potential recipient individually, this process is typically delegated to a trusted third party, which verifies the identity of the participating entities and certifies the resulting binding of a name to a public key with its signature. These signed public keys are referred to as public key certificates or 'digital ids'; the signing entity is called a certificate authority (CA). The existing X.509 standard was adapted for exchanging certificates over the internet.

An application of cryptographic algorithms, which is of particular interest, is the area of tamper resistant software. Every purely software based intellectual protection scheme has one or more vital components which has to be protected from tampering.

Electronic Commerce has not only triggered advances in the area of cryptography; a different and relatively new area of research is digital watermarking. Digital watermarks are barely perceptible transformations of digital data (often a digital multimedia object like image, audio or video data), which can be extracted computationally. The use of digital watermarks opens different scenarios:

- Ownership watermarks can be used in order to convey ownership information. In this scenario they identify the recipient of digital documents and facilitate the detection of copyright violations.
- Originator watermarks are applied by the content owners in order to automatically trace slight alterations of their intellectual property.
- Captioning is an application of digital watermarking which refers to the integration of the metadata into the digital documents. This allows for example to embed usage restrictions such as 'copy once' to be encoded into digital movies.

Another area of research is the development of flexible licensing mechanisms with so called rights management languages (RML). An RML is used to specify the credentials required by user to access a digital document and determines the resulting usage rights.

instructions which specify when and how identifying information is to be added to the documents. Digital signatures and certificates included in the cryptolope serve the purpose to authenticate the contents and optionally the users.

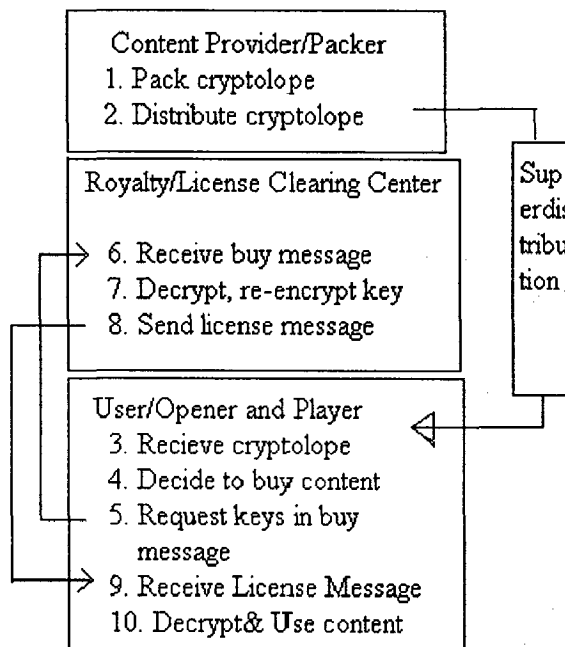


Fig 4.2: Cryptolope Processing

A cryptolope is created by the publisher of the content and can be distributed on arbitrary channels. Its security is inherently guaranteed because everybody can check the checksums and signatures, so nobody can tamper with a cryptolope and nobody can use the content without purchasing the PEKs.

The purchasing transaction requires a clearing center, which acts on behalf of the publisher. A client who decides to buy some content is directed by the cryptolope instructions to an appropriate clearing center. The buy request message contains the encrypted PEK and public key certificate. The clearing house knows the master key (which could be its own private key or a shared secret symmetric key), decrypts the PEK and re-encrypts it using the client's public key. A cryptolope-based solution is well suited to meet the digital library's requirements:

- Entity authentication is needed just between the client and clearing house: the publisher need not to have a special relationship with each other.
- Every cryptolope and every message is digitally signed and includes the certificate of the signer, so it can be checked easily. The signature process is

explicitly driven by the end user, so the signature can be considered as an act of free will.

- Checksums and signatures of the contents parts allow to check the authenticity and integrity of the content.
- Each encrypted part is confidential and can only be decrypted by an owner of the key, i.e. the publisher who created this key and the client who buys the keys from a clearing center. A clearing center is able to decrypt and sell the key, but generally does not decrypt the content. The information is in clear text only at the publisher's and the client's side.
- As cryptolope processing requires dedicated opener and viewer software running on the client, code-signing techniques can be applied to make software on the client side hard to tamper with.

Clients do not need to get a cryptolope directly and online from a publisher, but can copy a cryptolope from the nearest cache and purchase the unlocking key from any authorized clearing house. In order to realize site licensing, a clearing house can be authorized by the publisher to unlock cryptolopes for customers of its own domain for free after the institutional subscription fee has been paid.

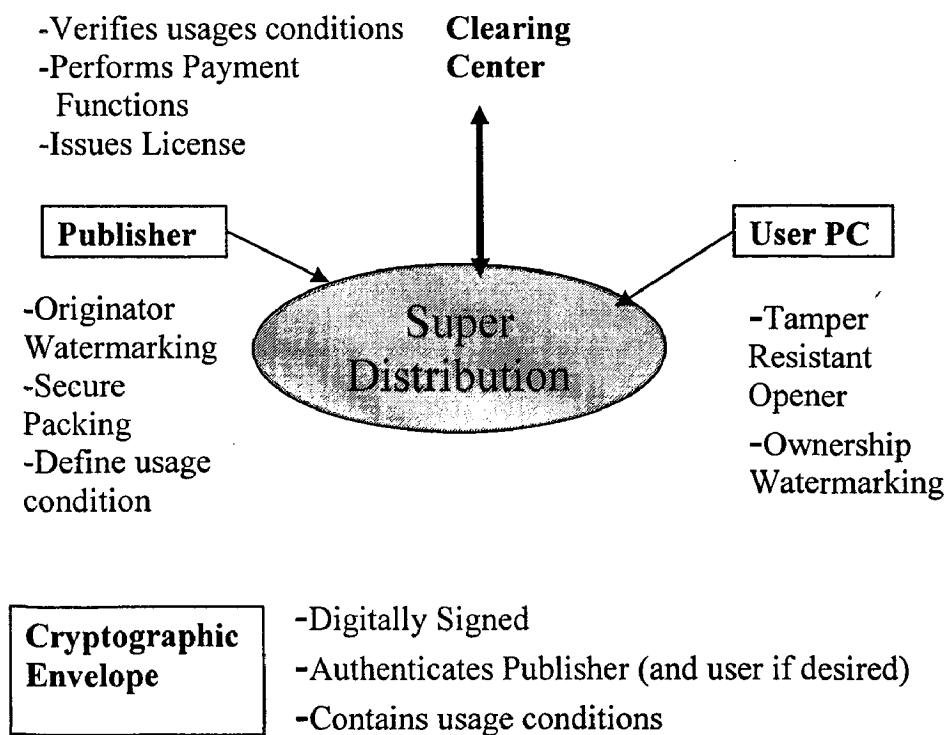


Fig 4.3: Proposed use of Security Building Blocks for Cryptographic Envelope Architecture

4.3 Implementation

4.3.1 Document Protection with Digital Signature Technique (Fingerprinted with Message Digest Function)- A digital signature is a value that is computed from a sequence of bytes using a secret key. They often use public-key encryption algorithms, whose approach is described below.

Signature generation follows these steps:

- A message digest is computed.
- The message digest is encrypted using the private key of a public/private key pair, producing the message's digital signature.

Signature verification follows these steps:

- The signature is decrypted using the public key of a public/private key pair, producing a message digest value.
- The message digest value is compared with the message digest calculated from the original message.
- If both values match, the signature is authentic. Otherwise, either the signature or the message has been tampered with.

The algorithm being used here is the Digital Signature Algorithm (DSA), being proposed by National Institute of Standards and Technology. DSA depends on the difficulty of computing discrete logarithms for its security. It makes use of the following:

- p = a prime number that is between 512 and 1024 bits in length and whose length is a multiple of 64 bits.
- q = a prime number that is a divisor of $p-1$ and is 160 bits in length.
- $g = h^{(p-1)/q} \bmod p$ where $1 < h < p-1$ is greater than 1.
- x = a random integer greater than 0 and less than q .
- $y = g^x \bmod p$
- k = a random integer greater than 0 and less than q . a new random value of k is generated for each signature.

The values of p , q , and g are public. The value of k is private and must be kept secret.

The public key is y and the private key is x .

A signature is calculated as follows:

- In the case of digital signatures, one-way hash function are used to condense the data to be signed to a fixed length hash value, which is then encrypted with the sender's private key. Examples for one-way hash functions are MD5 or SHA. If the hash function is not only based on the data but also on a secret key, it is called a Message Authentication Code (MAC).

The uses of asymmetric algorithms require the existence of trusted public keys. While this relationship of trust could be established for every potential recipient individually, this process is typically delegated to a trusted third party, which verifies the identity of the participating entities and certifies the resulting binding of a name to a public key with its signature. These signed public keys are referred to as public key certificates or 'digital ids'; the signing entity is called a certificate authority (CA). The existing X.509 standard was adapted for exchanging certificates over the internet.

An application of cryptographic algorithms, which is of particular interest, is the area of tamper resistant software. Every purely software based intellectual protection scheme has one or more vital components which has to be protected from tampering.

Electronic Commerce has not only triggered advances in the area of cryptography; a different and relatively new area of research is digital watermarking. Digital watermarks are barely perceptible transformations of digital data (often a digital multimedia object like image, audio or video data), which can be extracted computationally. The use of digital watermarks opens different scenarios:

- Ownership watermarks can be used in order to convey ownership information. In this scenario they identify the recipient of digital documents and facilitate the detection of copyright violations.
- Originator watermarks are applied by the content owners in order to automatically trace slight alterations of their intellectual property.
- Captioning is an application of digital watermarking which refers to the integration of the metadata into the digital documents. This allows for example to embed usage restrictions such as 'copy once' to be encoded into digital movies.

Another area of research is the development of flexible licensing mechanisms with so called rights management languages (RML). An RML is used to specify the credentials required by user to access a digital document and determines the resulting usage rights.

Securing Connections: Given the layered structure of the Internet protocols, security services can be introduced at different levels.

Fig 4.1 shows the four-layered protocol stack, which is commonly used to illustrate the structure of the Internet protocols: the Internet protocol (IP), which is mostly independent of the underlying physical network's structure, is a connectionless, packet-oriented transport layer protocol. Fig 4.1 uses the application level protocols, HTTP (WWW) and SMTP (e-mail) as examples.

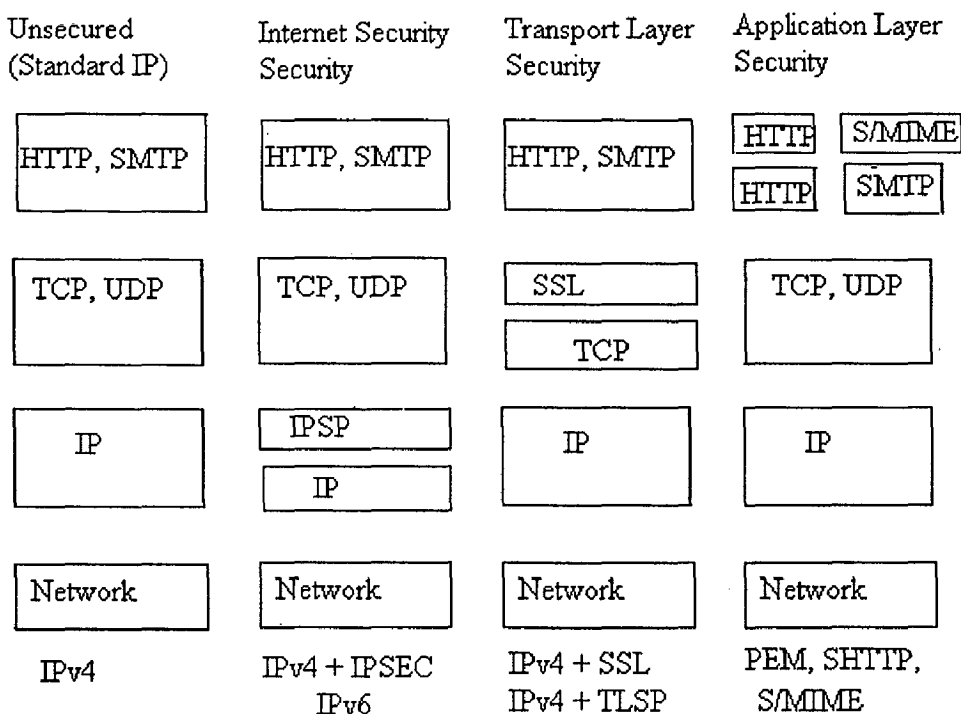


Fig 4.1: Integrating Security Services in the Internet Layer

Section-III

Internet Layer Security: An IETF working group called IPSEC is in the process of standardizing the necessary protocol structures, which will be available as an addition to the Internet Protocol in its current version (the so called IP Security Protocol IPSP) and as a part of the next version IPv6. The proposals are based on the use of DES for bulk data encryption and MD5 for hashing; a mechanism for performing key-exchange has not yet been standardized.

The application independence of Secure Socket Layer (SSL) has the disadvantage that that it can only offer point-to-point protection of the data during the communication process. SSL provides the services to authenticate a server, and optionally a client, to encrypt a session, and to authenticate messages. The application independence of SSL

has the disadvantage that it can only offer point-to-point protection of the data during the communication process.

Finally, security services can be integrated into the Internet protocols at the application level. This refers to the design of new or the adaption of existing application protocols in order to integrate security features into the protocol elements. One example of this approach is SHTTP, an extension of HTTP for security services. For a Digital Library application, the solutions presented have several shortcomings:

- The protected documents are separated from the associated usage rights and conditions.
- There is no way for the user to prove that a document was received under certain usage conditions. Once the document is transmitted to the user, these usage conditions cannot be enforced.
- A connection oriented security mechanism does not allow superdistribution of larger amounts of data.

Document protection requires the document to be wrapped in a secure container at the publisher's site, and only to be unwrapped at the end-user's computer. As a result, no further protection is needed, neither for the communication channel, nor for the intermediate tiers. Also, all of the intents of the publisher (protection, marking, etc) and all the terms and conditions he is offering, can be expressed in a tamper-evident digitally signed package. This enables superdistribution; the package can be moved freely from place to place without losing its intactness, its authenticity, and its associated terms and conditions.

Section- IV

Case study of IBM Cryptolope Technology: IBM has coined the name Cryptolope (cryptographic envelope) for its document protection technology.

A cryptolope consists of multiple parts. In addition to the encrypted document, a cryptolope contains a clear text description of the encrypted content, which serves to support a user's purchase decision. The metadata gives information about the contents as a whole, such as the author, size or format and instructions on how the content may be purchased. For each part, a different part encryption key (PEK) is chosen. The PEKs are themselves encrypted using a master key and stored in the key records of the cryptolope. The cryptolope further contains the terms and conditions describing the rights associated with the content and fingerprinting and watermarking

instructions which specify when and how identifying information is to be added to the documents. Digital signatures and certificates included in the cryptolope serve the purpose to authenticate the contents and optionally the users.

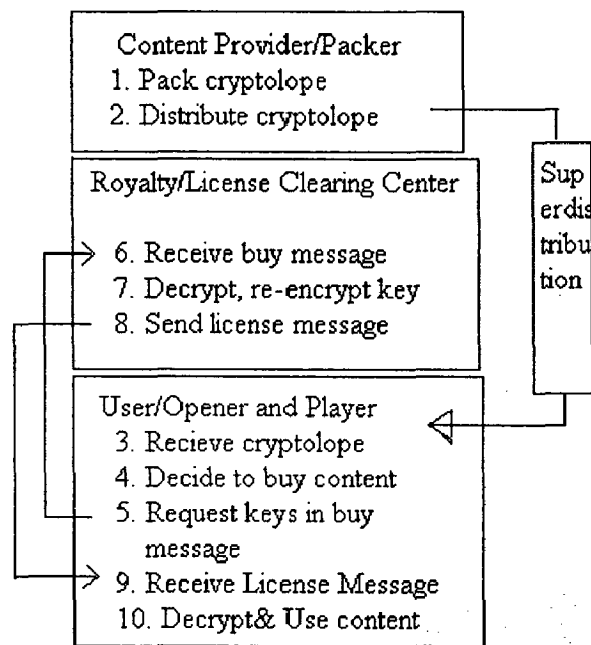


Fig 4.2: Cryptolope Processing

A cryptolope is created by the publisher of the content and can be distributed on arbitrary channels. Its security is inherently guaranteed because everybody can check the checksums and signatures, so nobody can tamper with a cryptolope and nobody can use the content without purchasing the PEKs.

The purchasing transaction requires a clearing center, which acts on behalf of the publisher. A client who decides to buy some content is directed by the cryptolope instructions to an appropriate clearing center. The buy request message contains the encrypted PEK and public key certificate. The clearing house knows the master key (which could be its own private key or a shared secret symmetric key), decrypts the PEK and re-encrypts it using the client's public key. A cryptolope-based solution is well suited to meet the digital library's requirements:

- Entity authentication is needed just between the client and clearing house: the publisher need not to have a special relationship with each other.
- Every cryptolope and every message is digitally signed and includes the certificate of the signer, so it can be checked easily. The signature process is

explicitly driven by the end user, so the signature can be considered as an act of free will.

- Checksums and signatures of the contents parts allow to check the authenticity and integrity of the content.
- Each encrypted part is confidential and can only be decrypted by an owner of the key, i.e. the publisher who created this key and the client who buys the keys from a clearing center. A clearing center is able to decrypt and sell the key, but generally does not decrypt the content. The information is in clear text only at the publisher's and the client's side.
- As cryptolope processing requires dedicated opener and viewer software running on the client, code-signing techniques can be applied to make software on the client side hard to tamper with.

Clients do not need to get a cryptolope directly and online from a publisher, but can copy a cryptolope from the nearest cache and purchase the unlocking key from any authorized clearing house. In order to realize site licensing, a clearing house can be authorized by the publisher to unlock cryptolopes for customers of its own domain for free after the institutional subscription fee has been paid.

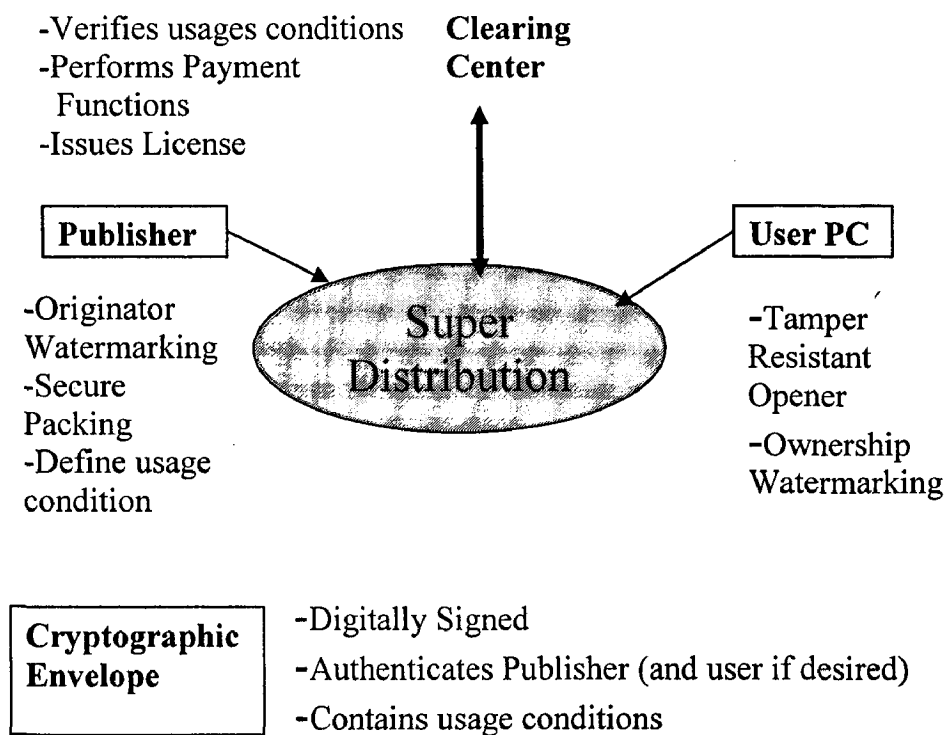


Fig 4.3: Proposed use of Security Building Blocks for Cryptographic Envelope Architecture

4.3 Implementation

4.3.1 Document Protection with Digital Signature Technique (Fingerprinted with Message Digest Function)- A digital signature is a value that is computed from a sequence of bytes using a secret key. They often use public-key encryption algorithms, whose approach is described below.

Signature generation follows these steps:

- A message digest is computed.
- The message digest is encrypted using the private key of a public/private key pair, producing the message's digital signature.

Signature verification follows these steps:

- The signature is decrypted using the public key of a public/private key pair, producing a message digest value.
- The message digest value is compared with the message digest calculated from the original message.
- If both values match, the signature is authentic. Otherwise, either the signature or the message has been tampered with.

The algorithm being used here is the Digital Signature Algorithm (DSA), being proposed by National Institute of Standards and Technology. DSA depends on the difficulty of computing discrete logarithms for its security. It makes use of the following:

- p = a prime number that is between 512 and 1024 bits in length and whose length is a multiple of 64 bits.
- q = a prime number that is a divisor of $p-1$ and is 160 bits in length.
- $g = h^{(p-1)/q} \bmod p$ where $1 < h < p-1$ is greater than 1.
- x = a random integer greater than 0 and less than q .
- $y = g^x \bmod p$
- k = a random integer greater than 0 and less than q . a new random value of k is generated for each signature.

The values of p , q , and g are public. The value of k is private and must be kept secret.

The public key is y and the private key is x .

A signature is calculated as follows:

- A message digest value (m) of the object to be signed is calculated using MD5.
- The value $r = (g^k \bmod p) \bmod q$ is calculated.
- The value $s = (k^{-1}(m + x r)) \bmod q$ is calculated.

The values of r and s are the signature. (If either r or s is zero, a new value of k is generated, and a new signature is calculated.)

Signature verification is performed as follows. The values r and s represent the values of r , and s that are received by the verifier:

- The values of r' and s' are verified to be greater than 0 and less than q .
- The message digest m of the received object is calculated. (The value of m should equal m if the object has not been modified.)
- The value $w = (s')^{-1} \bmod q$.
- The value $u_1 = (m' w) \bmod q$.
- The value $u_2 = (r' w) \bmod q$.
- The value $v = ((g^{u_1} y^{u_2}) \bmod p) \bmod q$.

The signature is verified in $v = r'$. It can be proved that $v = r'$ if $m' = m$, $r' = r$, and $s' = s$.

4.3.2 Message Digests- Cryptographic techniques are not limited to preserving the secrecy of messages, files, and other objects. They are also used to support the following:

- Integrity: To detect whether a message or object was modified or replaced.
- Authentication: To verify that a message or other object actually originated from a person or organization with a particular identity.
- Nonrepudiation: To prevent someone from denying that he or she sent a message or performed an operation on an object.

The previous tasks are accomplished through the use of message digests and digital signatures. A message digest is a special kind of function, referred to as a one-way (hash) function. A one-way function is easy to calculate, but difficult to reverse. Message digests are a special kind of one-way function that are used as fingerprints for messages. Good message digest functions have the following properties:

- Given a particular message digest value, it is computationally infeasible to compute a message that will produce that value under the message digest function.
- It is computationally infeasible to find two messages that yield the same message digest value under the message digest function.

It should be noted that there is nothing secret about a message digest function; it is publicly available and uses no keys. The Message Digest 5 (MD5) is one example of message digest algorithms. It is supported by the Java 2 Platform SDK.

MD5 is a message digest function developed by Ron Rivest. MD5 is operated on messages of arbitrary length and produces a 126-bit (16-byte) message digest.

4.3.3 Generation of Hash Function (Message Digest Algorithm) [22]

Here, a word is a 32-bit quantity and a byte is an eight-bit quantity. A sequence of bits can be interpreted in a natural manner as a sequence of bytes, where each consecutive group of eight bits is interpreted as a byte with the high-order (most significant) bit of each byte listed first. Similarly, a sequence of bytes can be interpreted as a sequence of 32-bit words, where each consecutive group of four bytes is interpreted as a word with the low-order (least significant) byte given first.

Let x_i denote x sub i . If the subscript is an expression, it is in braces, as in $x_{\{i+1\}}$. Similarly, we use $^$ for superscripts (exponentiation), so that x^i denotes x to the i -th power.

Let the symbol $+$ denote addition of words (i.e., modulo- 2^{32} addition). Let $X \lll s$ denote the 32-bit value obtained by circularly shifting (rotating) X left by s bit positions. Let $\text{not}(X)$ denote the bit-wise complement of X , and let $X \vee Y$ denote the bit-wise OR of X and Y . Let $X \text{ xor } Y$ denote the bit-wise XOR of X and Y , and let XY denote the bit-wise AND of X and Y .

MD5 Algorithm Description- It is supposed that we have a b -bit message as input, and that we wish to find its message digest. Here b is an arbitrary nonnegative integer; b may be zero, it need not be a multiple of eight, and it may be arbitrarily large. We imagine the bits of the message written down as follows:

$$m_0 m_1 \dots m_{\{b-1\}}$$

The following five steps are performed to compute the message digest of the message:

Step 1. Append Padding Bits:

The message is padded (extended) so that its length is congruent to 448, modulo 512.

That is, the message is extended so that it is just 64 bits shy of being a multiple of 512 bits long. Padding is always performed, even if the length of the message is already congruent to 448, modulo 512.

Padding is performed as follows: a single 1 bit is appended to the message, and then 0 bits are appended so that the length in bits of the padded message becomes congruent to 448, modulo 512. In all, at least one bit and at most 512 bits are appended.

Step 2. Append Length:

A 64-bit representation of b (the length of the message before the padding bits were added) is appended to the result of the previous step. In the unlikely event that b is greater than 2^{64} , then only the low-order 64 bits of b are used. (These bits are appended as two 32-bit words and appended low-order word first in accordance with the previous conventions.)

At this point the resulting message (after padding with bits and with b) has a length that is an exact multiple of 512 bits. Equivalently, this message has a length that is an exact multiple of 16 (32-bit) words. Let $M[0 \dots N-1]$ denote the words of the resulting message, where N is a multiple of 16.

Step 3. Initialize MD Buffer:

A four-word buffer (A, B, C, D) is used to compute the message digest. Here each of A, B, C, D is a 32-bit register. These registers are initialized to the following values in hexadecimal, low-order bytes first:

word A: 01 23 45 67

word B: 89 ab cd ef

word C: fe dc ba 98

word D: 76 54 32 10

Step 4. Process Message in 16-Word Blocks:

We first define four auxiliary functions that each take as input three 32-bit words and produce as output one 32-bit word.

$$F(X,Y,Z) = XY \vee \text{not}(X) Z$$

$$G(X,Y,Z) = XZ \vee Y \text{not}(Z)$$

$$H(X,Y,Z) = X \text{ xor } Y \text{ xor } Z$$

$$I(X,Y,Z) = Y \text{ xor } (X \vee \text{not}(Z))$$

In each bit position F acts as a conditional: if X then Y else Z . The function F could have been defined using $+$ instead of \vee since XY and $\text{not}(X)Z$ will never have 1's in the same bit position. It is interesting to note that if the bits of $X, Y,$ and Z are

independent and unbiased, then each bit of $F(X,Y,Z)$ will be independent and unbiased.

The functions G , H , and I are similar to the function F , in that they act in bitwise parallel to produce their output from the bits of X , Y , and Z , in such a manner that if the corresponding bits of X , Y , and Z are independent and unbiased, then each bit of $G(X,Y,Z)$, $H(X,Y,Z)$, and $I(X,Y,Z)$ will be independent and unbiased. The function H is the bit-wise xor or parity function of its inputs.

This step uses a 64-element table $T[1 \dots 64]$ constructed from the sine function. Let $T[i]$ denote the i -th element of the table, which is equal to the integer part of 4294967296 times $\text{abs}(\sin(i))$, where I is in radians.

Procedure-

Do the following:

```

/* Process each 16-word block. */
For i = 0 to N/16-1 do
  /* Copy block i into X. */
  For j = 0 to 15 do
    Set X[j] to M[i*16+j].
  end /* of loop on j */
  /* Save A as AA, B as BB, C as CC, and D as DD. */
  AA = A
  BB = B
  CC = C
  DD = D
  /* Round 1. */
  /* Let [abcd k s i] denote the operation
     a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s). */
  /* Do the following 16 operations. */
  [ABCD 0 7 1] [DABC 1 12 2] [CDAB 2 17 3] [BCDA 3 22 4]
  [ABCD 4 7 5] [DABC 5 12 6] [CDAB 6 17 7] [BCDA 7 22 8]
  [ABCD 8 7 9] [DABC 9 12 10] [CDAB 10 17 11] [BCDA 11 22 12]
  [ABCD 12 7 13] [DABC 13 12 14] [CDAB 14 17 15] [BCDA 15 22 16]
  /* Round 2. */
  /* Let [abcd k s i] denote the operation
     a = b + ((a + G(b,c,d) + X[k] + T[i]) <<< s). */

```

```

/* Do the following 16 operations. */
[ABCD 1 5 17] [DABC 6 9 18] [CDAB 11 14 19] [BCDA 0 20 20]
[ABCD 5 5 21] [DABC 10 9 22] [CDAB 15 14 23] [BCDA 4 20 24]
[ABCD 9 5 25] [DABC 14 9 26] [CDAB 3 14 27] [BCDA 8 20 28]
[ABCD 13 5 29] [DABC 2 9 30] [CDAB 7 14 31] [BCDA 12 20 32]
/* Round 3. */
/* Let [abcd k s t] denote the operation
   a = b + ((a + H(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
[ABCD 5 4 33] [DABC 8 11 34] [CDAB 11 16 35] [BCDA 14 23 36]
[ABCD 1 4 37] [DABC 4 11 38] [CDAB 7 16 39] [BCDA 10 23 40]
[ABCD 13 4 41] [DABC 0 11 42] [CDAB 3 16 43] [BCDA 6 23 44]
[ABCD 9 4 45] [DABC 12 11 46] [CDAB 15 16 47] [BCDA 2 23 48]
/* Round 4. */
/* Let [abcd k s t] denote the operation
   a = b + ((a + I(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
[ABCD 0 6 49] [DABC 7 10 50] [CDAB 14 15 51] [BCDA 5 21 52]
[ABCD 12 6 53] [DABC 3 10 54] [CDAB 10 15 55] [BCDA 1 21 56]
[ABCD 8 6 57] [DABC 15 10 58] [CDAB 6 15 59] [BCDA 13 21 60]
[ABCD 4 6 61] [DABC 11 10 62] [CDAB 2 15 63] [BCDA 9 21 64]
/* Then perform the following additions. (That is increment each
   of the four registers by the value it had before this block
   was started.) */
A = A + AA
B = B + BB
C = C + CC
D = D + DD
end /* of loop on i */

```

Step 5. Output.

The message digest produced as output is A, B, C, D. That is, we begin with the low-order byte of A, and end with the high-order byte of D.

4.3.4 The Digital Signature Algorithm [23]

Digital signature algorithm (DSA) depends on the difficulty of computing discrete logarithms for its security. The DSA makes use of the following parameters:

- p = a prime modulus, where $2^{L-1} < p < 2^L$ for $512 \leq L \leq 1024$ and L a multiple of 64.
- q = a prime divisor of $p - 1$, where $2^{159} < q < 2^{160}$
- $g = h^{(p-1)/q} \bmod p$, where h is any integer with $1 < h < p - 1$ such that $h^{(p-1)/q} \bmod p > 1$
- (g has order $q \bmod p$)
- x = a randomly or pseudorandomly generated integer with $0 < x < q$
- $y = g^x \bmod p$
- k = a randomly or pseudorandomly generated integer with $0 < k < q$

The integers p , q , and g can be public and can be common to a group of users. A user's private and public keys are x and y , respectively. They are normally fixed for a period of time. Parameters x and k are used for signature generation only, and must be kept secret. Parameter k must be regenerated for each signature. Parameters p , q , x and k shall be generated as described later.

DSA Signature Generation

The signature of a message M is the pair of numbers r and s computed according to the equations below:

$$r = (g^k \bmod p) \bmod q \text{ and}$$

$$s = (k^{-1}(M + xr)) \bmod q.$$

Here, k^{-1} is the multiplicative inverse of $k \bmod q$; i.e., $(k^{-1} k) \bmod q = 1$ and $0 < k^{-1} < q$. As an option, one may wish to check if $r = 0$ or $s = 0$. If either $r = 0$ or $s = 0$, a new value of k should be generated and the signature should be recalculated (it is extremely unlikely that $r = 0$ or $s = 0$ if signatures are generated properly).

The signature is transmitted along with the message to the verifier.

DSA Signature Verification

Prior to verifying the signature in a signed message, p , q and g plus the sender's public key and identity are made available to the verifier in an authenticated manner.

Let M' , r' , and s' be the received versions of M , r , and s , respectively, and let y be the public key of the signatory. To verify signature, the verifier checks to see that $0 < r' <$

q and $0 < s' < q$; if either condition is violated the signature shall be rejected. If these two conditions are satisfied, the verifier computes

$$w = (s')^{-1} \bmod q$$

$$u_1 = (M'w) \bmod q$$

$$u_2 = (r'w) \bmod q$$

$$v = ((g^{u_1} y^{u_2}) \bmod p) \bmod q.$$

If $v = r'$, then the signature is verified and the verifier can have high confidence that the received message was sent by the party holding the secret key x corresponding to y .

If v does not equal r' , then the message may have been modified, the message may have been incorrectly signed by the signatory, or the message may have been signed by an impostor. The message should be considered invalid.

Proof that $v = r'$

In the DSA, if $M' = M$, $r' = r$ and $s' = s$ in the signature verification then $v = r'$. We need the following easy result:

Lemma: Let p and q be primes so that q divides $p-1$, h a positive integer less than p , and $g = h^{(p-1)/q} \bmod p$. Then $g^q \bmod p = 1$, and if $m \bmod q = n \bmod q$, then $g^m \bmod p = g^n \bmod p$.

Proof: We have

$$\begin{aligned} g^q \bmod p &= (h^{(p-1)/q} \bmod p)^q \bmod p \\ &= h^{(p-1)} \bmod p \\ &= 1 \end{aligned}$$

by Fermat's Little Theorem. Now let $m \bmod q = n \bmod q$, i.e., $m = n + kq$ for some integer k . Then

$$\begin{aligned} g^m \bmod p &= g^{n+kq} \bmod p \\ &= (g^n g^{kq}) \bmod p \\ &= ((g^n \bmod p) (g^q \bmod p)^k) \bmod p \\ &= gn \bmod p \end{aligned}$$

since $g^q \bmod p = 1$.

The main result is proved as follows:

Theorem: If $M' = M$, $r' = r$, and $s' = s$ in the signature verification, then $v = r'$.

Proof: We have

$$w = (s')^{-1} \bmod q = s^{-1} \bmod q$$

$$u_1 = (M'w) \bmod q = (Mw) \bmod q$$

$$u_2 = (r'w) \bmod q = (rw) \bmod q.$$

Now $y = g^x \bmod p$, so that by the lemma,

$$v = ((g^{u_1} y^{u_2}) \bmod p) \bmod q$$

$$= ((g^{(M)w} y^{rw}) \bmod p) \bmod q$$

$$= ((g^{(M)w} g^{xrw}) \bmod p) \bmod q$$

$$= ((g^{((M+xr)w)}) \bmod p) \bmod q.$$

Also

$$s = (k^{-1}(M + xr)) \bmod q.$$

Hence

$$w = (k(M + xr)^{-1}) \bmod q$$

$$(M + xr)w \bmod q = k \bmod q.$$

Thus by the lemma,

$$v = (g^k \bmod p) \bmod q$$

$$= r$$

$$= r'.$$

Generation of Primes for the DSA

The algorithms for generating the primes p and q used in the DSA are discussed here.

These algorithms require a random number generator and an efficient modular exponentiation algorithm, which are described below.

A Probabilistic Primality Test- In order to generate the primes p and q , a primality test is required.

There are several fast probabilistic algorithms available. The following algorithm is a simplified version of a procedure due to M.O. Rabin, based in part on ideas of Gary L. Miller. If this algorithm is iterated n times, it will produce a false prime with probability no greater than $1/4^n$.

Therefore, $n \geq 50$ will give an acceptable probability of error. To test whether an integer is prime:

Step 1: Set $i = 1$ and $n \geq 50$.

Step 2: Set $w =$ the integer to be tested, $w = 1 + 2^a m$, where m is odd and 2^a is the largest

power of 2 dividing $w - 1$.

Step 3: Generate a random integer b in the range $1 < b < w$.

Step 4: Set $j = 0$ and $z = b^m \bmod w$.

Step 5: If $j = 0$ and $z = 1$, or if $z = w - 1$, go to step 9.

Step 6: If $j > 0$ and $z = 1$, go to step 8.

Step 7: $j = j + 1$. If $j < a$, set $z = z^2 \bmod w$ and go to step 5.

Step 8: w is not prime. Stop.

Step 9: If $i < n$, set $i = i + 1$ and go to step 3. Otherwise, w is probably prime.

Generation of Primes- The DSA requires two primes, p and q , satisfying the following three conditions:

a. $2^{159} < q < 2^{160}$

b. $2^{L-1} < p < 2^L$ for a specified L , where $L = 512 + 64j$ for some $0 \leq j \leq 8$

c. q divides $p - 1$.

This prime generation scheme starts by a user supplied SEED to construct a prime, q , in the range $2^{159} < q < 2^{160}$. Once this is accomplished, the same SEED value is used to construct an X in the range $2^{L-1} < X < 2^L$. The prime, p , is then formed by rounding X to a number congruent to 1 mod $2q$ as described below.

An integer x in the range $0 \leq x < 2^g$ may be converted to a g -long sequence of bits by using its binary expansion as shown below:

$$x = x_1 * 2^{g-1} + x_2 * 2^{g-2} + \dots + x_{g-1} * 2 + x_g \rightarrow \{x_1, \dots, x_g\}.$$

Conversely, a g -long sequence of bits $\{x_1, \dots, x_g\}$ is converted to an integer by the rule

$$\{x_1, \dots, x_g\} \rightarrow x_1 * 2^{g-1} + x_2 * 2^{g-2} + \dots + x_{g-1} * 2 + x_g.$$

The first bit of a sequence corresponds to the most significant bit of the corresponding integer and the last bit to the least significant bit.

Let $L - 1 = n * 160 + b$, where both b and n are integers and $0 \leq b < 160$.

Step 1: Choose an arbitrary sequence of at least 160 bits and call it SEED. Let g be the length of SEED in bits.

Step 2: Compute

$$U = [\text{SEED}] \text{ XOR } [(\text{SEED} + 1) \bmod 2^g].$$

Step 3: Form q from U by setting the most significant bit (the 2^{159} bit) and the least significant bit to 1. In terms of boolean operations, $q = U \text{ OR } 2^{159} \text{ OR } 1$. Note that $2^{159} < q < 2^{160}$.

Step 4: Use a robust primality testing algorithm to test whether q is prime.

Step 5: If q is not prime, go to step 1.

Step 6: Let counter = 0 and offset = 2.

Step 7: For $k = 0, \dots, n$ let

$$V_k = [(SEED + \text{offset} + k) \bmod 2^8].$$

A robust primality test is one where the probability of a non-prime number passing the test is at most 2^{-80} .

Step 8: Let W be the integer

$$W = V_0 + V_1 * 2^{160} + \dots + V_{n-1} * 2^{(n-1) * 160} + (V_n \bmod 2^b) * 2^{n * 160}$$

and let $X = W + 2^{L-1}$. Note that $0 \leq W < 2^{L-1}$ and hence $2^{L-1} \leq X < 2^L$.

Step 9: Let $c = X \bmod 2q$ and set $p = X - (c - 1)$. Note that p is congruent to 1 mod $2q$.

Step 10: If $p < 2^{L-1}$, then go to step 13.

Step 11: Perform a robust primality test on p .

Step 12: If p passes the test performed in step 11, go to step 15.

Step 13: Let counter = counter + 1 and offset = offset + n + 1.

Step 14: If counter $\geq 2^{12} = 4096$ go to step 1, otherwise (i.e. if counter < 4096) go to step 7.

Step 15: Save the value of SEED and the value of counter for use in certifying the proper generation of p and q .

Random Number Generation for the DSA

Any implementation of the DSA requires the ability to generate random or pseudorandom integers. Such numbers are used to derive a user's private key, x , and a user's message secret number, k . These randomly or pseudorandomly generated integers are selected to be between 0 and the 160-bit prime q . The algorithms employ a one-way function $G(t,c)$, where t is 160 bits, c is b bits ($160 \leq b \leq 512$) and $G(t,c)$ is 160 bits. One method for constructing G is to use the Data Encryption Standard (DES).

In the succeeding algorithm, a secret b -bit seed-key is used. The algorithm optionally allows the use of a user provided input. If DES is used to construct G , then b is equal to 160.

Algorithm for computing m values of x - Let x be the signer's private key. The following may be used to generate m values of x :

Step 1: Choose a new, secret value for the seed-key, XKEY.

Step 2: In hexadecimal notation let

$t = 67452301 \text{ EFCDAB89 } 98\text{BADCFE } 10325476 \text{ C3D2E1F0}$.

This is the initial value for $H_0 \parallel H_1 \parallel H_2 \parallel H_3 \parallel H_4$ in the SHS.

Step 3: For $j = 0$ to $m - 1$ do

- a. $XSEED_j =$ optional user input.
- b. $XVAL = (XKEY + XSEED_j) \bmod 2^b$.
- c. $x_j = G(t, XVAL) \bmod q$.
- d. $XKEY = (1 + XKEY + x_j) \bmod 2^b$.

Algorithm for precomputing one or more k and r values- This algorithm can be used to precompute k , k^{-1} , and r for m messages at a time.

Algorithm:

Step 1: Choose a secret initial value for the seed-key, KKEY.

Step 2: In hexadecimal notation let

$t = \text{EFCDAB89 } 98\text{BADCFE } 10325476 \text{ C3D2E1F0 } 67452301$.

This is a cyclic shift of the initial value for $H_0 \parallel H_1 \parallel H_2 \parallel H_3 \parallel H_4$ in the SHS.

Step 3: For $j = 0$ to $m - 1$ do

- a. $k = G(t, KKEY) \bmod q$.
- b. Compute $k_j^{-1} = k^{-1} \bmod q$.
- c. Compute $r_j = (g^k \bmod p) \bmod q$.
- d. $KKEY = (1 + KKEY + k) \bmod 2^b$.

Step 4: Suppose M_0, \dots, M_{m-1} are the next m messages. For $j = 0$ to $m - 1$ do

- a. Let $h = M_j$.
- b. Let $s_j = (k_j^{-1}(h + xr_j)) \bmod q$.
- c. The signature for M_j is (r_j, s_j) .

Step 5: Let $t = h$.

Step 6: Go to step 3.

Step 3 permits precomputation of the quantities needed to sign the next m messages. Step 4 can begin whenever the first of these m messages is ready. The execution of step 4 can be suspended whenever the next of the m messages is not ready. As soon as steps 4 and 5 have completed, step 3 can be executed, and the results saved until the first member of the next group of m messages is ready.

In addition to space for KKEY, two arrays of length m are needed to store r_0, \dots, r_{m-1} and $k_0^{-1}, \dots, k_{m-1}^{-1}$ when they are computed in step 3. Storage for s_0, \dots, s_{m-1} is only needed if the signatures for a group of messages are stored; otherwise s_j in step 4 can be replaced by s and a single space allocated.

Constructing the function G from the DES- Let a XOR b denote the bitwise exclusive-or of bit strings a and b . Suppose a_1, a_2, b_1, b_2 are 32-bit strings. Let b_1' be the 24 least significant bits of b_1 . Let $K = b_1' \parallel b_2$ and $A = a_1 \parallel a_2$. Define $DES_{b_1, b_2}(a_1, a_2) = DES_K(A)$

In the above, $DES_K(A)$ represents ordinary DES encryption of the 64-bit block A using the 56-bit key K . Now suppose t and c are each 160 bits. To compute $G(t, c)$:

Step 1: Write

$$t = t_1 \parallel t_2 \parallel t_3 \parallel t_4 \parallel t_5$$

$$c = c_1 \parallel c_2 \parallel c_3 \parallel c_4 \parallel c_5$$

In the above, each t_i and c_i is 32 bits.

Step 2: For $i = 1$ to 5 do

$$x_i = t_i \text{ XOR } c_i$$

Step 3: For $i = 1$ to 5 do

$$b_1 = c_{((i+3) \bmod 5) + 1}$$

$$b_2 = c_{((i+2) \bmod 5) + 1}$$

$$a_1 = x_i$$

$$a_2 = x_{(i \bmod 5) + 1} \text{ XOR } x_{((i+3) \bmod 5) + 1}$$

$$y_{i,1} \parallel y_{i,2} = DES_{b_1, b_2}(a_1, a_2) \quad (y_{i,1}, y_{i,2} = 32 \text{ bits})$$

Step 4: For $i = 1$ to 5 do

$$z_i = y_{i,1} \text{ XOR } y_{((i+1) \bmod 5) + 1, 2} \text{ XOR } y_{((i+2) \bmod 5) + 1, 1}$$

Step 5: Let

$$G(t, c) = z_1 \parallel z_2 \parallel z_3 \parallel z_4 \parallel z_5$$

Generation of other quantities for the DSA- The algorithms discussed below may be used to generate the quantities g, k^{-1} , and s^{-1} used in the DSA.

To generate g :

Step 1: Generate p and q as described above.

Step 2: Let $e = (p - 1)/q$.

Step 3: Set $h =$ any integer, where $1 < h < p - 1$ and h differs from any value previously

tried.

Step 4: Set $g = h^e \text{ mod } p$.

Step 5: If $g = 1$, go to step 3.

To compute the multiplicative inverse $n^{-1} \text{ mod } q$ for n with $0 < n < q$, where $0 < n^{-1} < q$:

Step 1: Set $i = q$, $h = n$, $v = 0$, and $d = 1$.

Step 2: Let $t = i \text{ DIV } h$, where DIV is defined as integer division.

Step 3: Set $x = h$.

Step 4: Set $h = i - tx$.

Step 5: Set $i = x$.

Step 6: Set $x = d$.

Step 7: Set $d = v - tx$.

Step 8: Set $v = x$.

Step 9: If $h > 0$, go to step 2.

Step 10: Let $n^{-1} = v \text{ mod } q$.

Note that in step 10, v may be negative. The $v \text{ mod } q$ operation should yield a value between 1 and $q - 1$ inclusive.

EXAMPLE OF THE DSA

Let $L = 512$ (size of p). The values in this example are expressed in hexadecimal notation. The p and q given here were generated by the prime generation standard using the 160-bit SEED:

d5014e4b 60ef2ba8 b6211b40 62ba3224 e0427dd3

With this SEED, the algorithm found p and q when the counter was at 105. x was generated by the algorithm described above and a 160-bit XKEY:

XKEY = bd029bbe 7f51960b cf9edb2b 61f06f0f eb5a38b6

$t = 67452301 \text{ EFCDAB89 } 98\text{BADCFE } 10325476 \text{ C3D2E1F0}$

$x = G(t, \text{XKEY}) \text{ mod } q$

k was generated by the algorithm described above and a 160-bit KKEY:

KKEY = 687a66d9 0648f993 867e121f 4ddf9ddb 01205584

$t = \text{EFCDAB89 } 98\text{BADCFE } 10325476 \text{ C3D2E1F0 } 67452301$

$k = G(t, \text{KKEY}) \text{ mod } q$

$h = 2$

$p = 8df2a494 492276aa 3d25759b b06869cb eac0d83a fb8d0cf7$
 $cbb8324f 0d7882e5 d0762fc5 b7210eaf c2e9adac 32ab7aac$

49693dfb f83724c2 ec0736ee 31c80291
 q = c773218c 737ec8ee 993b4f2d ed30f48e dace915f
 g = 626d0278 39ea0a13 413163a5 5b4cb500 299d5522 956cefcb
 3bff10f3 99ce2c2e 71cb9de5 fa24babf 58e5b795 21925c9c
 c42e9f6f 464b088c c572af53 e6d78802
 x = 2070b322 3dba372f de1c0ffc 7b2e3b49 8b260614
 k = 358dad57 1462710f 50e254cf 1a376b2b deaadfbf
 k-1 = 0d516729 8202e49b 4116ac10 4fc3f415 ae52f917
 M = ASCII form of abc
 M = a9993e36 4706816a ba3e2571 7850c26c 9cd0d89d
 y = 19131871 d75b1612 a819f29d 78d1b0d7 346f7aa7 7bb62a85
 9bfd6c56 75da9d21 2d3a36ef 1672ef66 0b8c7c25 5cc0ec74
 858fba33 f44c0669 9630a76b 030ee333
 r = 8bac1ab6 6410435c b7181f95 b16ab97c 92b341c0
 s = 41e2345f 1f56df24 58f426d1 55b4ba2d b6dcd8c8
 w = 9df4ece5 826be95f ed406d41 b43edc0b 1c18841b
 u1 = bf655bd0 46f0b35e c791b004 804afcb b8ef7d69d
 u2 = 821a9263 12e97ade abcc8d08 2b527897 8a2df4b0
 $g^{u1} \bmod p =$ 51b1bf86 7888e5f3 af6fb476 9dd016bc fe667a65 aafc2753
 9063bd3d 2b138b4c e02cc0c0 2ec62bb6 7306c63e 4db95bbf
 6f96662a 1987a21b e4ec1071 010b6069
 $y^{u2} \bmod p =$ 8b510071 2957e950 50d6b8fd 376a668e 4b0d633c 1e46e665
 5c611a72 e2b28483 be52c74d 4b30de61 a668966e dc307a67
 c19441f4 22bf3c34 08aeba1f 0a4dbec7
 v = 8bac1ab6 6410435c b7181f95 b16ab97c 92b341c0

4.3.2 Envelope Encryption with AES Algorithm [24, 25, 26, 27, 28, 29]

Introduction

The AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information. Encryption converts data to an unintelligible form called ciphertext; decrypting the ciphertext converts the data back into its original form, called plaintext. The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits. This standard

specifies the **Rijndael** algorithm, a symmetric block cipher that can process data blocks of 128 bits, using cipher keys with lengths of 128, 192, and 256 bits.

Design rationale

The three criteria taken into account in the design of Rijndael are the following:

- Resistance against all known attacks.
- Speed and code compactness on a wide range of platforms.
- Design simplicity.

In most ciphers, the round transformation has the Feistel Structure. In this structure typically part of the bits of the intermediate state are simply transposed unchanged to another position. The round transformation of Rijndael does not have the Feistel structure. Instead, the round transformation is composed of three distinct invertible uniform transformations, called layers. By uniform, we mean that every bit of the state is treated in a similar way. The specific choices for the different layers are for a large part based on the application of the Wide Trail Strategy, a design method to provide resistance against linear and differential cryptanalysis. In the Wide Trail Strategy, every layer has its own function:

(i) The linear mixing layer: guarantees high diffusion over multiple rounds.

(ii) The non-linear layer: parallel application of S-boxes that have optimum worst-case nonlinearity properties.

(iii) The key addition layer: A simple EXOR of the Round Key to the intermediate state. Before the first round, a key addition layer is applied. The motivation for this initial key addition is that, any layer after the last key addition in the cipher (or before the first in the context of known-plaintext attacks) can be simply peeled off without knowledge of the key and therefore does not contribute to the security of the cipher. Initial or terminal key addition is applied in several designs, e.g., IDEA, SAFER and Blowfish.

In order to make the cipher and its inverse more similar in structure, the linear mixing layer of the last round is different from the mixing layer in the other rounds. It does not improve or reduce the security of the cipher in any way.

Algorithm Parameters, Symbols, and Functions

The following algorithm parameters, symbols, and functions are used in this standard:

AddRoundKey(): Transformation in the Cipher and Inverse Cipher in which a Round Key is added to the State using an XOR operation. The length of a Round Key equals the size of the State (i.e., for $Nb = 4$, the Round Key length equals 128 bits/16 bytes).

InvMixColumns(): Transformation in the Inverse Cipher that is the inverse of MixColumns().

InvShiftRows(): Transformation in the Inverse Cipher that is the inverse of ShiftRows().

InvSubBytes(): Transformation in the Inverse Cipher that is the inverse of SubBytes().

K: Cipher Key.

MixColumns(): Transformation in the Cipher that takes all of the columns of the state and mixes their data (independently of one another) to produce new columns.

Nb: Number of columns (32-bit words) comprising the state. For this standard, $Nb = 4$.

Nk: Number of 32-bit words comprising the Cipher Key. Here, $Nk = 4, 6, \text{ or } 8$.

Nr: Number of rounds, which is a function of Nk and Nb (which is fixed). For this standard, $Nr = 10, 12, \text{ or } 14$.

Rcon[]: The round constant word array.

RotWord(): Function used in the Key Expansion routine that takes a four-byte word and performs a cyclic permutation.

ShiftRows(): Transformation in the Cipher that processes the State by cyclically shifting the last three rows of the State by different offsets.

SubBytes(): Transformation in the Cipher that processes the state using a nonlinear byte substitution table (S-box) that operates on each of the state bytes independently.

SubWord(): Function used in the Key Expansion routine that takes a four-byte input word and applies an S-box to each of the four bytes to produce an output word.

XOR: Exclusive-OR operation.

\oplus : Exclusive-OR operation.

\otimes : Multiplication of two polynomials (each with degree < 4) modulo $x^4 + 1$.

\cdot : Finite field multiplication.

Inputs and Outputs- The input and output for the AES algorithm each consist of sequences of 128 bits (digits with values of 0 or 1). These sequences will sometimes be referred to as blocks and the number of bits they contain will be referred to as their length. The Cipher Key for the AES algorithm is a sequence of 128, 192 or 256 bits. Other input, output and Cipher Key lengths are not permitted by this standard. The

bits within such sequences will be numbered starting at zero and ending at one less than the sequence length (block length or key length). The number i attached to a bit is known as its index and will be in one of the ranges $0 \leq i < 128$, $0 \leq i < 192$ or $0 \leq i < 256$ depending on the block length and key length (specified above).

Bytes- The basic unit for processing in the AES algorithm is a byte, a sequence of eight bits treated as a single entity. The input, output and Cipher Key bit sequences described above are processed as arrays of bytes that are formed by dividing these sequences into groups of eight contiguous bits to form arrays of bytes. For an input, output or Cipher Key denoted by a , the bytes in the resulting array will be referenced using one of the two forms, a_n or $a[n]$, where n will be in one of the following ranges:

Key length = 128 bits, $0 \leq n < 16$; Block length = 128 bits, $0 \leq n < 16$;

Key length = 192 bits, $0 \leq n < 24$;

Key length = 256 bits, $0 \leq n < 32$.

All byte values in the AES algorithm will be presented as the concatenation of its individual bit values (0 or 1) between braces in the order $\{b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0\}$.

These bytes are interpreted as finite field elements using a polynomial representation:

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0 = b_i x^i \quad (1)$$

For example, $\{01100011\}$ identifies the specific finite field element $x^6 + x^5 + x + 1$.

It is also convenient to denote byte values using hexadecimal notation with each of two groups of four bits being denoted by a single character as in Fig 4.4.

Bit Pattern	Character	Bit Pattern	Character	Bit Pattern	Character	Bit Pattern	Character
0000	0	0100	4	1000	8	1100	c
0001	1	0101	5	1001	9	1101	d
0010	2	0110	6	1010	a	1110	e
0011	3	0111	7	1011	b	1111	f

Fig 4.4: Hexadecimal representation of Bit Patterns

Hence the element $\{01100011\}$ can be represented as $\{63\}$, where the character denoting the four-bit group containing the higher numbered bits is again to the left.

Some finite field operations involve one additional bit (b_8) to the left of an 8-bit byte.

Where this extra bit is present, it will appear as $\{01\}$ immediately preceding the 8-bit byte; for example, a 9-bit sequence will be presented as $\{01\}\{1b\}$.

Arrays of Bytes- Arrays of bytes will be represented in the following form:

$$a_0a_1a_2\dots a_{15}$$

The bytes and the bit ordering within bytes are derived from 128-bit input sequence:

input₀input₁input₂... input₁₂₆input₁₂₇

as follows:

$$\begin{aligned}
 a_0 &= \{\text{input}_0, \text{input}_1, \dots, \text{input}_7\}, \\
 a_1 &= \{\text{input}_8, \text{input}_9, \dots, \text{input}_{15}\}, \\
 &\vdots \\
 a_{15} &= \{\text{input}_{120}, \text{input}_{121}, \dots, \text{input}_{127}\}.
 \end{aligned}$$

The pattern can be extended to longer sequences (i.e., for 192- and 256-bit keys), so that, in general,

$$a_n = \{\text{input}_{8n}, \text{input}_{8n+1}, \dots, \text{input}_{8n+7}\}. \quad (2)$$

Input bit sequence	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	...
Byte number	0							1							2							...			
Bit numbers in byte	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	...

Fig 4.5: Indices for Bits and Bytes

Fig 4.5 shows how bits within each byte are numbered taking equations (1) and (2) together.

The State- Internally, the AES algorithm’s operations are performed on a two-dimensional array of bytes called the State. The State consists of four rows of bytes, each containing Nb bytes, where Nb is the block length divided by 32. In the State array denoted by the symbol *s*, each individual byte has two indices, with its row number *r* in the range $0 \leq r < 4$ and its column number *c* in the range $0 \leq c < \text{Nb}$. This allows an individual byte of the State to be referred to as either *s_{r,c}* or *s[r,c]*. For this standard, Nb=4, i.e., $0 \leq c < 4$.

At the start of the Cipher and Inverse Cipher, the input: the array of bytes *in₀*, *in₁*, ... *in₁₅*: is copied into the State array as illustrated in Fig 4.6. The Cipher or Inverse Cipher operations are then conducted on this State array, after which its final value is copied to the output – the array of bytes *out₀*, *out₁*, ... *out₁₅*.

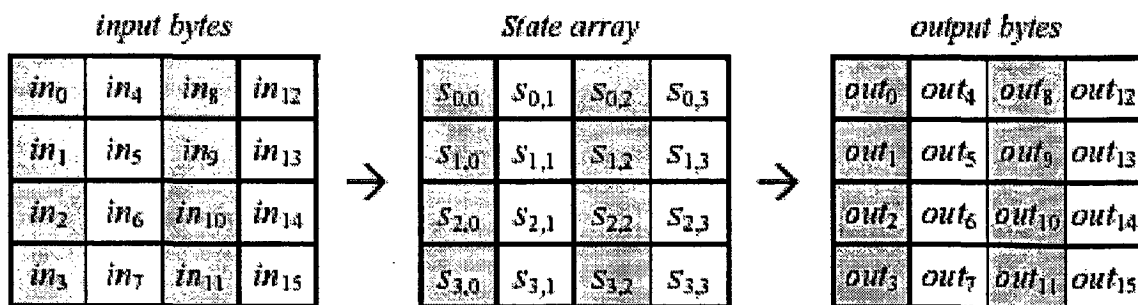


Fig 4.6: State Array as Input and Output

Hence, at the beginning of the Cipher or Inverse Cipher, the input array, in , is copied to the State array according to the scheme:

$$s[r, c] = in[r + 4c] \text{ for } 0 \leq r < 4 \text{ and } 0 \leq c < Nb,$$

and at the end of the Cipher and Inverse Cipher, the State is copied to the output array out as follows:

$$out[r + 4c] = s[r, c] \text{ for } 0 \leq r < 4 \text{ and } 0 \leq c < Nb.$$

The State as an Array of Columns- The four bytes in each column of the State array form 32-bit words, where the row number r provides an index for the four bytes within each word. The state can hence be interpreted as a one-dimensional array of 32 bit words (columns), $w_0...w_3$, where the column number c provides an index into this array. Hence, for the example in Fig. 4.6, the State can be considered as an array of four words, as follows:

$$w_0 = s_{0,0} s_{1,0} s_{2,0} s_{3,0} \quad w_2 = s_{0,2} s_{1,2} s_{2,2} s_{3,2}$$

$$w_1 = s_{0,1} s_{1,1} s_{2,1} s_{3,1} \quad w_3 = s_{0,3} s_{1,3} s_{2,3} s_{3,3}$$

Algorithm Specification

For the AES algorithm, the length of the input block, the output block and the State is 128 bits. This is represented by $Nb = 4$, which reflects the number of 32-bit words (number of columns) in the State.

For the AES algorithm, the length of the Cipher Key, K , is 128, 192, or 256 bits. The key length is represented by $Nk = 4, 6, \text{ or } 8$, which reflects the number of 32-bit words (number of columns) in the Cipher Key.

For the AES algorithm, the number of rounds to be performed during the execution of the algorithm is dependent on the key size. The number of rounds is represented by Nr , where $Nr = 10$ when $Nk = 4$, $Nr = 12$ when $Nk = 6$, and $Nr = 14$ when $Nk = 8$.

The Key-Block-Round combinations that conform to this standard is given in Fig 4.7.

	Key Length (Nk words)	Block Size (Nb words)	Number of Rounds (Nr)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Fig 4.7: Key-Block-Round Combinations

For both its Cipher and Inverse Cipher, the AES algorithm uses a round function that is composed of four different byte-oriented transformations: 1) byte substitution using a substitution table (S-box), 2) shifting rows of the State array by different offsets, 3) mixing the data within each column of the State array, and 4) adding a Round Key to the State.

Cipher- At the start of the Cipher, the input is copied to the State array using the conventions described above. After an initial Round Key addition, the State array is transformed by implementing a round function 10, 12, or 14 times (depending on the key length), with the final round differing slightly from the first $Nr - 1$ rounds. The final State is then copied to the output.

The round function is parameterized using a key schedule that consists of a one-dimensional array of four-byte words derived using the Key Expansion routine. The individual transformations - SubBytes(), ShiftRows(), MixColumns(), and AddRoundKey() – process the State. Here, the array $w[]$ contains the key schedule. Also all Nr rounds are identical with the exception of the final round, which does not include the MixColumns() transformation. The Cipher is described in the pseudo code below:

```

Cipher (byte in [4*Nb]), byte out [4*Nb], word w [Nb*(Nr + 1)])
begin
    byte state [4, Nb]
    state = in
    AddRoundkey (state, w [0, Nb-1])
    for round = 1 step 1 to Nr-1
        SubBytes (state)
        ShiftRows (state)
        MixColumns (state)
        AddRoundKey (state, w [round*Nb, (round + 1)*Nb-1])
    end for
    SubBytes (state)
    ShiftRows (state)
    AddRoundKey (state, w [Nr*Nb, (Nr + 1)*Nb-1])
    out = state
end

```

Key Expansion- The AES algorithm takes the Cipher Key, K , and performs a Key Expansion routine to generate a key schedule. The Key Expansion generates a total of $N_b(N_r + 1)$ words: the algorithm requires an initial set of N_b words, and each of the N_r rounds requires N_b words of key data. The resulting key schedule consists of a linear array of 4-byte words, denoted $[w_i]$, with i in the range $0 \leq i < N_b(N_r + 1)$.

The expansion of the input key into the key schedule proceeds according to the pseudo code described below.

`SubWord()` is a function that takes a four-byte input word and applies the S-box to each of the four bytes to produce an output word. The function `RotWord()` takes a word $[a_0, a_1, a_2, a_3]$ as input, performs a cyclic permutation, and returns the word $[a_1, a_2, a_3, a_0]$. The round constant word array, `Rcon[i]`, contains the values given by $[x^{i-1}, \{00\}, \{00\}, \{00\}]$, with x^{i-1} being powers of x .

It can be seen below that the first N_k words of the expanded key are filled with the Cipher Key. Every following word, $w[i]$, is equal to the XOR of the previous word, $w[i-1]$, and the word N_k positions earlier, $w[i-N_k]$. For words in positions that are a multiple of N_k , a transformation is applied to $w[i-1]$ prior to the XOR, followed by an XOR with a round constant, `Rcon[i]`. This transformation consists of a cyclic shift of the bytes in a word (`RotWord()`), followed by the application of a table lookup to all four bytes of the word (`SubWord()`). It is important to note that the Key Expansion routine for 256-bit Cipher Keys ($N_k = 8$) is slightly different than for 128- and 192-bit Cipher Keys. If $N_k = 8$ and $i-4$ is a multiple of N_k , then `SubWord()` is applied to $w[i-1]$ prior to the XOR. The Pseudo Code is given below:

```

KeyExpansion (byte key[4*Nk], word w[Nb*(Nr + 1), Nk])
begin
    word temp
    i = 0
    while (i < Nk)
        w[i] = word (key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
        i = i+1
    end while
    i = Nk
    while (i < Nb*(Nr+1))

```



```

temp = w[i-1]
if (i mod Nk = 0)
    temp = SubWord (RotWord (temp)) xor Rcon[1/Nk]
elseif (Nk>6 and i mod Nk = 4)
    temp = SubWord (temp)
endif
w[i] = w[i-Nk] xor temp
i = i+1
end while
end

```

Inverse Cipher- The Cipher transformations can be inverted and then implemented in reverse order to produce a straightforward Inverse Cipher. The individual transformations used in the Inverse Cipher: InvShiftRows(), Invsbbytes(), InvMixColumns(), and AddRoundKey(), process the State. The Inverse Cipher is described in the pseudo code below, where the array w[] contains the key schedule:

```

InvCipher (byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
    byte state[4,Nb]
    state = in
    AddRoundKey (state, w[Nr*Nb, (Nr+1)*Nb-1])
    for round = Nr-1 step -1 downto 1
        InvShiftRows (state)
        InvSubBytes (state)
        AddRoundKey (state, w[round*Nb, (round+1)*Nb-1])
        InvMixColumns (state)
    end for
    InvShiftRows (state)
    InvSubBytes (state)
    AddRoundKey (state, w[0, Nb-1])
    out = state
end

```

Equivalent Inverse Cipher- In the straightforward Inverse Cipher, the sequence of the transformations differs from that of the Cipher, while the form of the key schedules for encryption and decryption remains the same. However, several properties of the AES algorithm allow for an Equivalent Inverse Cipher that has the same sequence of transformations as the Cipher. This is accomplished with a change in the key schedule.

The two properties that allow for this Equivalent Inverse Cipher are as follows:

1. The SubBytes() and ShiftRows() transformations commute; that is, a SubBytes() transformation immediately followed by a ShiftRows() transformation is equivalent to a ShiftRows() transformation immediately followed by a SubBytes() transformation. The same is true for their inverses, InvSubBytes() and InvShiftRows.
2. The column mixing operations - MixColumns() and InvMixColumns() – are linear with respect to the column input, which means $\text{InvMixColumns}(\text{state XOR Round Key}) = \text{InvMixColumns}(\text{state}) \text{ XOR } \text{InvMixColumns}(\text{Round Key})$.

These properties allow the order of InvSubBytes() and InvShiftRows() transformations to be reversed. The order of the AddRoundKey() and InvMixColumns() transformations can also be reversed, provided that the columns (words) of the decryption key schedule are modified using the InvMixColumns() transformation.

The equivalent inverse cipher is defined by reversing the order of the InvSubBytes() and InvShiftRows() transformations, and by reversing the order of the AddRoundKey() and InvMixColumns() transformations used in the round loop after first modifying the decryption key schedule for $\text{round} = 1$ to $\text{Nr}-1$ using the InvMixColumns() transformation. The first and last Nb words of the decryption key schedule shall not be modified in this manner. The modification to the Key Expansion routine is also provided in the Pseudo Code below:

```
EqInvCipher (byte in[4*Nb], byte out[4*Nb], word dw[Nb*(Nr+1)])
begin
    byte state[4, Nb]
    state = in
    AddRoundKey (state, dw[Nr*Nb, (Nr+1)*Nb-1])
    for round = Nr-1 step -1 downto 1
        InvShiftRows (state)
```

```

    InvSubBytes (state)
    InvMixColumns (state)
    AddRoundKey (state, dw[round*Nb, (round+1)*Nb-1])
end for
    InvShiftRows (state)
    InvSubBytes (state)
    AddRoundKey (state, dw[0, Nb-1])
    out = state
end

```

Implementation Issues

Key Length Requirements- An implementation of the AES algorithm shall support at least one of the three key lengths: 128, 192, or 256 bits (i.e., $N_k = 4, 6,$ or $8,$ respectively). Implementations may optionally support two or three key lengths, which may promote the interoperability of algorithm implementations.

Keying Restrictions- No weak or semi-weak keys have been identified for the AES algorithm, and there is no restriction on key selection.

Parameterization of Key Length, Block Size, and Round Number- This standard explicitly defines the allowed values for the key length (N_k), block size (N_b), and number of rounds (N_r). However, future reaffirmations of this standard could include changes or additions to the allowed values for those parameters.

Implementation Suggestions Regarding Various Platforms- Implementation variations are possible that may, in many cases, offer performance or other advantages. Given the same input key and data (plaintext or ciphertext), any implementation that produces the same output (ciphertext or plaintext) as the algorithm specified in this standard is an acceptable implementation of the AES.

Example Vectors

AES-128 ($N_k=4, N_r=10$)

PLAINTEXT: 00112233445566778899aabbccddeeff

KEY: 000102030405060708090a0b0c0d0e0f

CIPHER (ENCRYPT):

```

round[ 0]: input 00112233445566778899aabbccddeeff
round[ 0]: k_sch 000102030405060708090a0b0c0d0e0f

```

round[1]: start 00102030405060708090a0b0c0d0e0f0
round[1]: s_box 63cab7040953d051cd60e0e7ba70e18c
round[1]: s_row 6353e08c0960e104cd70b751bacad0e7
round[1]: m_col 5f72641557f5bc92f7be3b291db9f91a
round[1]: k_sch d6aa74fdd2af72fadaa678f1d6ab76fe
round[2]: start 89d810e8855ace682d1843d8cb128fe4
round[2]: s_box a761ca9b97be8b45d8ad1a611fc97369
round[2]: s_row a7be1a6997ad739bd8c9ca451f618b61
round[2]: m_col ff87968431d86a51645151fa773ad009
round[2]: k_sch b692cf0b643dbdf1be9bc5006830b3fe
round[3]: start 4915598f55e5d7a0daca94fa1f0a63f7
round[3]: s_box 3b59cb73fcd90ee05774222dc067fb68
round[3]: s_row 3bd92268fc74fb735767cbe0c0590e2d
round[3]: m_col 4c9c1e66f771f0762c3f868e534df256
round[3]: k_sch b6ff744ed2c2c9bf6c590cbf0469bf41
round[4]: start fa636a2825b339c940668a3157244d17
round[4]: s_box 2dfb02343f6d12dd09337ec75b36e3f0
round[4]: s_row 2d6d7ef03f33e334093602dd5bfb12c7
round[4]: m_col 6385b79ffc538df997be478e7547d691
round[4]: k_sch 47f7f7bc95353e03f96c32bcfd058dfd
round[5]: start 247240236966b3fa6ed2753288425b6c
round[5]: s_box 36400926f9336d2d9fb59d23c42c3950
round[5]: s_row 36339d50f9b539269f2c092dc4406d23
round[5]: m_col f4bcd45432e554d075f1d6c51dd03b3c
round[5]: k_sch 3caaa3e8a99f9deb50f3af57adf622aa
round[6]: start c81677bc9b7ac93b25027992b0261996
round[6]: s_box e847f56514dadde23f77b64fe7f7d490
round[6]: s_row e8dab6901477d4653ff7f5e2e747dd4f
round[6]: m_col 9816ee7400f87f556b2c049c8e5ad036
round[6]: k_sch 5e390f7df7a69296a7553dc10aa31f6b
round[7]: start c62fe109f75eedc3cc79395d84f9cf5d
round[7]: s_box b415f8016858552e4bb6124c5f998a4c
round[7]: s_row b458124c68b68a014b99f82e5f15554c
round[7]: m_col c57e1c159a9bd286f05f4be098c63439
round[7]: k_sch 14f9701ae35fe28c440adf4d4ea9c026
round[8]: start d1876c0f79c4300ab45594add66ff41f
round[8]: s_box 3e175076b61c04678dfc2295f6a8bfc0
round[8]: s_row 3e1c22c0b6fcbf768da85067f6170495
round[8]: m_col baa03de7a1f9b56ed5512cba5f414d23
round[8]: k_sch 47438735a41c65b9e016baf4aebf7ad2
round[9]: start fde3bad205e5d0d73547964ef1fe37f1
round[9]: s_box 5411f4b56bd9700e96a0902fa1bb9aa1
round[9]: s_row 54d990a16ba09ab596bbf40ea111702f
round[9]: m_col e9f74eec023020f61bf2ccf2353c21c7
round[9]: k_sch 549932d1f08557681093ed9cbe2c974e
round[10]: start bd6e7c3df2b5779e0b61216e8b10b689
round[10]: s_box 7a9f102789d5f50b2beffd9f3dca4ea7
round[10]: s_row 7ad5fda789ef4e272bca100b3d9ff59f
round[10]: k_sch 13111d7fe3944a17f307a78b4d2b30c5
round[10]: output 69c4e0d86a7b0430d8cdb78070b4c55a

INVERSE CIPHER (DECRYPT):

round[0]: iinput 69c4e0d86a7b0430d8cdb78070b4c55a
round[0]: ik_sch 13111d7fe3944a17f307a78b4d2b30c5
round[1]: istart 7ad5fda789ef4e272bca100b3d9ff59f
round[1]: is_row 7a9f102789d5f50b2beffd9f3dca4ea7
round[1]: is_box bd6e7c3df2b5779e0b61216e8b10b689
round[1]: ik_sch 549932d1f08557681093ed9cbe2c974e
round[1]: ik_add e9f74eec023020f61bf2ccf2353c21c7
round[2]: istart 54d990a16ba09ab596bbf40ea111702f
round[2]: is_row 5411f4b56bd9700e96a0902fa1bb9aa1
round[2]: is_box fde3bad205e5d0d73547964ef1fe37f1
round[2]: ik_sch 47438735a41c65b9e016baf4aebf7ad2
round[2]: ik_add baa03de7a1f9b56ed5512cba5f414d23
round[3]: istart 3e1c22c0b6fcbf768da85067f6170495
round[3]: is_row 3e175076b61c04678dfc2295f6a8bfc0
round[3]: is_box d1876c0f79c4300ab45594add66ff41f
round[3]: ik_sch 14f9701ae35fe28c440adf4d4ea9c026
round[3]: ik_add c57e1c159a9bd286f05f4be098c63439
round[4]: istart b458124c68b68a014b99f82e5f15554c
round[4]: is_row b415f8016858552e4bb6124c5f998a4c
round[4]: is_box c62fe109f75eedc3cc79395d84f9cf5d
round[4]: ik_sch 5e390f7df7a69296a7553dc10aa31f6b
round[4]: ik_add 9816ee7400f87f556b2c049c8e5ad036
round[5]: istart e8dab6901477d4653ff7f5e2e747dd4f
round[5]: is_row e847f56514dadde23f77b64fe7f7d490
round[5]: is_box c81677bc9b7ac93b25027992b0261996
round[5]: ik_sch 3caaa3e8a99f9deb50f3af57adf622aa
round[5]: ik_add f4bcd45432e554d075f1d6c51dd03b3c
round[6]: istart 36339d50f9b539269f2c092dc4406d23
round[6]: is_row 36400926f9336d2d9fb59d23c42c3950
round[6]: is_box 247240236966b3fa6ed2753288425b6c
round[6]: ik_sch 47f7f7bc95353e03f96c32bcfd058dfd
round[6]: ik_add 6385b79ffc538df997be478e7547d691
round[7]: istart 2d6d7ef03f33e334093602dd5bfb12c7
round[7]: is_row 2dfb02343f6d12dd09337ec75b36e3f0
round[7]: is_box fa636a2825b339c940668a3157244d17
round[7]: ik_sch b6ff744ed2c2c9bf6c590cbf0469bf41
round[7]: ik_add 4c9c1e66f771f0762c3f868e534df256
round[8]: istart 3bd92268fc74fb735767cbe0c0590e2d
round[8]: is_row 3b59cb73fcd90ee05774222dc067fb68
round[8]: is_box 4915598f55e5d7a0daca94fa1f0a63f7
round[8]: ik_sch b692cf0b643dbdf1be9bc5006830b3fe
round[8]: ik_add ff87968431d86a51645151fa773ad009
round[9]: istart a7be1a6997ad739bd8c9ca451f618b61
round[9]: is_row a761ca9b97be8b45d8ad1a611fc97369
round[9]: is_box 89d810e8855ace682d1843d8cb128fe4
round[9]: ik_sch d6aa74fdd2af72fadaa678f1d6ab76fe
round[9]: ik_add 5f72641557f5bc92f7be3b291db9f91a
round[10]: istart 6353e08c0960e104cd70b751bacad0e7
round[10]: is_row 63cab7040953d051cd60e0e7ba70e18c

round[10]: is_box 00102030405060708090a0b0c0d0e0f0
round[10]: ik_sch 000102030405060708090a0b0c0d0e0f
round[10]: ioutput 00112233445566778899aabbccddeeff

EQUIVALENT INVERSE CIPHER (DECRYPT):

round[0]: iinput 69c4e0d86a7b0430d8cdb78070b4c55a
round[0]: ik_sch 13111d7fe3944a17f307a78b4d2b30c5
round[1]: istart 7ad5fda789ef4e272bca100b3d9ff59f
round[1]: is_box bdb52189f261b63d0b107c9e8b6e776e
round[1]: is_row bd6e7c3df2b5779e0b61216e8b10b689
round[1]: im_col 4773b91ff72f354361cb018ea1e6cf2c
round[1]: ik_sch 13aa29be9c8faff6f770f58000f7bf03
round[2]: istart 54d990a16ba09ab596bbf40ea111702f
round[2]: is_box fde596f1054737d235febad7f1e3d04e
round[2]: is_row fde3bad205e5d0d73547964ef1fe37f1
round[2]: im_col 2d7e86a339d9393ee6570a1101904e16
round[2]: ik_sch 1362a4638f2586486bff5a76f7874a83
round[3]: istart 3e1c22c0b6fcbf768da85067f6170495
round[3]: is_box d1c4941f7955f40fb46f6c0ad68730ad
round[3]: is_row d1876c0f79c4300ab45594add66ff41f
round[3]: im_col 39daee38f4fla82aaf432410c36d45b9
round[3]: ik_sch 8d82fc749c47222be4dad3e9c7810f5
round[4]: istart b458124c68b68a014b99f82e5f15554c
round[4]: is_box c65e395df779cf09ccf9e1c3842fed5d
round[4]: is_row c62fe109f75eedc3cc79395d84f9cf5d
round[4]: im_col 9a39bfl d05b20a3a476a0bf79fe51184
round[4]: ik_sch 72e3098d11c5de5f789dfe1578a2cccb
round[5]: istart e8dab6901477d4653ff7f5e2e747dd4f
round[5]: is_box c87a79969b0219bc2526773bb016c992
round[5]: is_row c81677bc9b7ac93b25027992b0261996
round[5]: im_col 18f78d779a93eef4f6742967c47f5ffd
round[5]: ik_sch 2ec410276326d7d26958204a003f32de
round[6]: istart 36339d50f9b539269f2c092dc4406d23
round[6]: is_box 2466756c69d25b236e4240fa8872b332
round[6]: is_row 247240236966b3fa6ed2753288425b6c
round[6]: im_col 85cf8bf472d124c10348f545329c0053
round[6]: ik_sch a8a2f5044de2c7f50a7ef79869671294
round[7]: istart 2d6d7ef03f33e334093602dd5bfb12c7
round[7]: is_box fab38a1725664d2840246ac957633931
round[7]: is_row fa636a2825b339c940668a3157244d17
round[7]: im_col fc1fc1f91934c98210fbfb8da340eb21
round[7]: ik_sch c7c6e391e54032f1479c306d6319e50c
round[8]: istart 3bd92268fc74fb735767cbe0c0590e2d
round[8]: is_box 49e594f755ca638fda0a59a01f15d7fa
round[8]: is_row 4915598f55e5d7a0daca94fa1f0a63f7
round[8]: im_col 076518f0b52ba2fb7a15c8d93be45e00
round[8]: ik_sch a0db02992286d160a2dc029c2485d561
round[9]: istart a7be1a6997ad739bd8c9ca451f618b61
round[9]: is_box 895a43e485188fe82d121068cbd8ced8
round[9]: is_row 89d810e8855ace682d1843d8cb128fe4

```

round[ 9]: im_col ef053f7c8b3d32fd4d2a64ad3c93071a
round[ 9]: ik_sch 8c56dff0825dd3f9805ad3fc8659d7fd
round[10]: istart 6353e08c0960e104cd70b751bacad0e7
round[10]: is_box 0050a0f04090e03080d02070c01060b0
round[10]: is_row 00102030405060708090a0b0c0d0e0f0
round[10]: ik_sch 000102030405060708090a0b0c0d0e0f
round[10]: ioutput 00112233445566778899aabbccddeeff

```

Advantages and limitations

Advantages- Implementation aspects:

- Rijndael can be implemented to run at speeds unusually fast for a block cipher on a Pentium (Pro). There is a trade-off between table size/performance.
- Rijndael can be implemented on a Smart Card in a small amount of code, using a small amount of RAM and taking a small number of cycles. There is some ROM performance trade-off.
- The round transformation is parallel by design, an important advantage in future processors and dedicated hardware.
- As the cipher does not make use of arithmetic operations, it has no bias towards big or little endian processor architectures.
- Simplicity of Design:
- The cipher is fully “self-supporting”. It does not make use of another cryptographic component, S-boxes “lent” from well-reputed ciphers, bits obtained from Rand tables, digits of π or any other such jokes.
- The cipher does not base its security or part of it on obscure and not well understood interactions between arithmetic operations.
- The tight cipher design does not leave enough room to hide a trapdoor.
- Variable block length:
- The block lengths of 192 and 256 bits allow the construction of a collision-resistant iterated hash function using Rijndael as the compression function. The block length of 128 bits is not considered sufficient for this purpose nowadays.
- Extensions:
- The design allows the specification of variants with the block length and key length both ranging from 128 to 256 bits in steps of 32 bits.

- Although the number of rounds of Rijndael is fixed in the specification, it can be modified as a parameter in case of security problems.

Limitations- The limitations of the cipher have to do with its inverse:

- The inverse cipher is less suited to be implemented on a smart card than the cipher itself: it takes more code and cycles. (Still, compared with other ciphers, even the inverse is very fast)
- In software, the cipher and its inverse make use of different code and/or tables.
- In hardware, the inverse cipher can only partially re-use the circuitry that implements the cipher.

4.4 Classes and Methods Used in the Implementation

Message Class- We should note the following about the coding of MessageDigest class:

- It is not necessary to import javax.crypto or use JCE. All the classes and interfaces needed to calculate message digests are provided by the Java 2 Platform SDK (in the java.security classes).
- The getInstance() method of the MessageDigest class is used to create a MessageDigest object. The name of the message digest algorithm (MD5) is passed as an argument to getInstance().
- A DigestInputStream object is created to read an input stream and calculate a message digest on that stream. The input stream and the MessageDigest object are passed as arguments to the DigestInputStream () constructor.
- After the stream has been read, the getMessageDigest() method of DigestInputStream is invoked to calculate the final value of the message digest.MessageDigest.

The MessageDigest class provides a concrete (nonabstract) subclass of MessageDigestSpi that serves as the foundation for message digest computation. It supports the MD5 and SHA message digest algorithms.

The MessageDigest constructor is protected. MessageDigest are created via the static getInstance() method. A String identifying the type of message digest algorithm to be used is supplied as an argument.

```
MessageDigest md = MessageDigest.getInstance("MD5");
```

The methods of MessageDigest ARE AS follows:

- static MessageDigest getInstance(String *algorithm*) – Creates a MessageDigest object that implements the specified digest algorithm.

- static MessageDigest getInstance(String *algorithm*, String *provider*) – Creates a MessageDigest object that implements the specified digest algorithm from the specified provider (if available).
- String getAlgorithm() – Returns a String that identifies the algorithm used by the MessageDigest object.
- void update(byte *input*) – Updates the digest computation using the specified byte.
- void update(byte[] *input*) – Updates the digest computation using the specified array of bytes.
- void update(byte[] *input*, int *offset*, int *len*) – Updates the digest computation using the specified portion of an array of bytes.
- byte[] digest() – Finishes the hash computation and returns the message digest as a byte array.
- byte[] digest(byte[] *input*) – Updates the digest computation using the specified array of bytes, finishes the digest computation, and returns the digest as a byte array.
- int digest (byte[] *buf*,int *offset*, int *len*) – Finishes the digest computation and returns the number of bytes in the message digest. The message digest is written to the specified location in *buf*.
- int getDigestLength() – Returns the length of digest in bytes.
- void reset () – Resets the digest to the initial state.
- Provider getProvider() – Returns the provider of the MessageDigest object.
- String toString() – Returns a String representation of the MessageDigest object.
- Object clone() – Returns a clone of the MessageDigest if the implementation starts the Cloneable interface.

Conversion Class- Message class uses the Conversion class. This class provides methods for converting array of bytes to String objects that displays the bytes in hexadecimal and base 64 format. Base 64 format provides an easy way to convert a binary file to printable text so that it can be included in the text of a message. When using base 64, three binary bytes are converted to four printable characters, as follows:

- The bytes are laid out one after another to form a 24-bit sequence.

- The 24-bit sequence is organized into four 6-bit units.
- Each of 6-bit units represents an integer value between 0 and 63.
- Four characters are selected from Table 1 using these four integer values.

In some cases, such as at the end of a message or file, you might only have one or two bytes to convert. In these cases, padding is used to perform the conversion:

- When encoding two bytes, the bytes form a 16-bit sequence. Two zero bits are added to form a 18-bit sequence. The 18-bit sequence is organized into three 6-bit values, which are used to select three characters from Table 1. A fourth padding character (=) is appended to these three characters.
- When encoding a single byte, the byte forms a 8-bit sequence. Four zero bits are added to form a 12-bit sequence. The 12-bit sequence is organized into two 6-bit values, which are used to select two characters from Table 4.8. Two padding characters, each of which are =, are appended to the two characters from Table 4.8.

Value	Code	Value	Code	Value	Code	Value	Code
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	I	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

Table 4.8: Base 64 Encoding

Signature Class- The Signature class provides a default implementation of the SignatureSpi abstract methods and supports both signature generation and verification. It supports an algorithm referred to as SHA1withDSA. This algorithm uses DSA to create and verify DSA digital signatures.

The Signature class is a factory class, and Signature objects are created with the static getInstance() method:

```
Signature sign = Signature.getInstance ("DSA");
```

A second form of method allows a particular service provider to be specified.

The methods of Signature are as follows:

- static Signature getInstance(String *algorithm*) – Creates a Signature object that implements the specified signature algorithm.
- static Signature getInstance(String *algorithm*, String *provider*) – Creates a Signature object that implements the specified signature algorithm and is supplied from the specified service provider(if available).
- void setParameter(AlgorithmParameterSpec *params*) – Initializes the signature engine with the specified parameter set.
- void setParameter(String *param*, Object *value*) – This method has been deprecated in JDK1.2 .Use the previous form of setParameter().
- Object getParameter(String *param*) – This method has been deprecated in JDK1.2.
- Provider getProvider() – Returns the provider of the Signature object.
- String getAlgorithm() – Returns the name of the algorithm used with the Signature object.
- void initSign(PrivateKey *privateKey*) – Initialize the Signature object for signing with the specified private key.
- void initSign(PrivateKey *privateKey*, SecureRandom *random*) – Initialize the Signature object for signing with the specified private key and source of randomness.
- void initVerify(PublicKey *publicKey*) – Initialize the Signature object for verification with the specified public key.
- void update(byte *b*) – Updates the data to be signed or verified by a byte.
- void update(byte[] *data*) – Updates the data to be signed or verified using the specified byte array.

- `void update(byte[] data, int off, int len)` – Updates the data to be signed or verified using the specified portion of byte array.
- `byte[] sign()` – Returns the signature that has been calculated as a byte array.
- `int sign(byte[] outbuf, int offset, int len)` – Stores the calculated signature in *outbuf* and returns the length of the signature in bytes.
- `boolean verify(byte[] signature)` – Verifies the specified signature and returns a boolean value indicating the success and failure of the verification.
- `String toString()` – Returns a representation of the Signature object.
- `Object clone()` – Provides support for implementing the Cloneable interface.

The Signature class also defines the protected state variable, which can be in one of the three states, UNINITIALIZED, SIGN and VERIFY.

Dialog Demo Class- Often we want to use a *dialog box* to hold a set of related controls. Dialog boxes are primarily used to obtain user input. They are similar to frame windows, except that dialog boxes are always child windows of top-level window. Also, they don't have menu bars. In other respects, dialog boxes function like frame windows. Dialog boxes may be modal or modeless. When a modal dialog box is active, all input is directed to it until it is closed. This means that you cannot access other parts of your program until you have closed the dialog box. When a modeless dialog box is active, input focus can be directed to another window in your program. Thus, other parts of your program remain active and accessible. Dialog boxes are of type Dialog. The most commonly used constructors are shown here:

- `Dialog(Frame parentWindow, boolean mode)`
- `Dialog(Frame parentWindow, String title, boolean mode)`

Here, *parentWindow* is the owner of the Dialog box. If *mode* is true, the dialog box is modal. Otherwise, it is modeless. The title of the dialog box can be passed in a *title*. Generally, you will subclass Dialog, adding the functionality required by your application. When the dialog box is closed, `dispose ()` is called.

The above program uses the NIST Digital Signature Algorithm for signing and is organized according to the `generateKeyPair ()`, `performSigning ()`, and `performVerification ()` methods. The `generateKeyPair ()` method returns a `KeyPair` object that is used in both `performSigning ()` and `performVerification ()`. The `performSigning ()` method returns a byte array representation of a Signature object. The `performVerification ()` method verifies this signature.

The `generateKeyPair ()` method creates a `KeyPairGenerator` object for DSA and generates a `KeyPair`. The `performSigning ()` method creates a `Signature` object and then initializes it, via `initSign ()`, for signing, using the private key of the `KeyPair` object produced by `generateKeyPair ()`. It then updates the signature via the `update ()` method and creates the final signature via the `sign ()` method. This signature is returned as the result of `performSigning ()`.

The `performVerification ()` method verifies the signature that was returned by `performSigning ()`. It uses the public key of the `KeyPair` object created by `generateKeyPair ()`.

This method creates a `Signature` object that uses DSA and initializes it for verification using the public key. It invokes the `update ()` method of `Signature` to create a new signature and the `verify ()` method to perform the signature verification.

RESULTS AND DISCUSSIONS

Advantages of using DSA algorithm

The MD5 message-digest algorithm is simple to implement, and provides a fingerprint or message digest of a message of arbitrary length. It is conjectured that the difficulty of coming up with two messages having the same message digest is on the order of 2^{64} operations, and that the difficulty of coming up with any message having a given message digest is on the order of 2^{128} operations. The level of security discussed is considered to be sufficient for implementing very high security hybrid digital-signature schemes based on MD5 and a public-key cryptosystem.

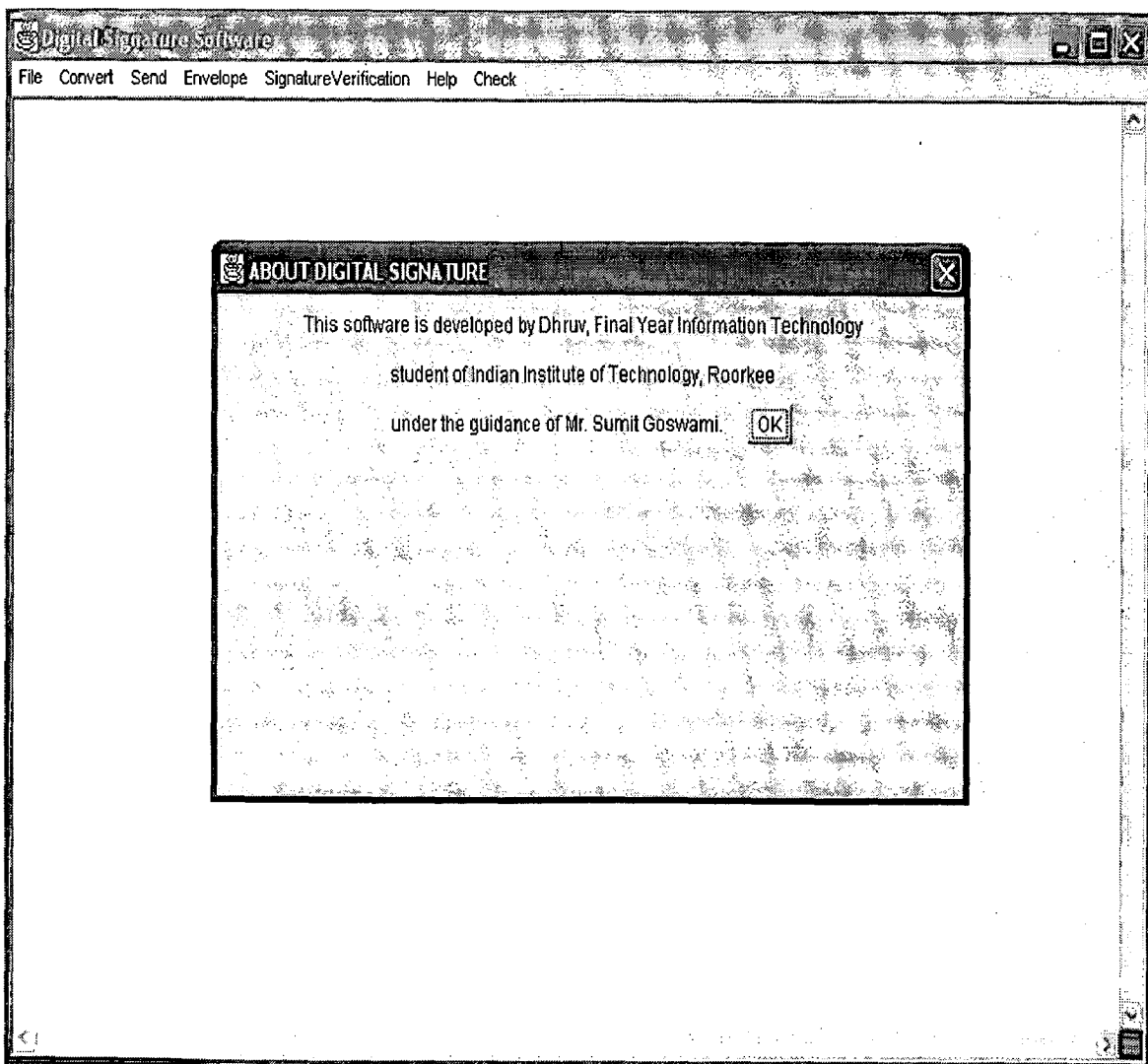


Fig 5.1: About Digital Signature Software

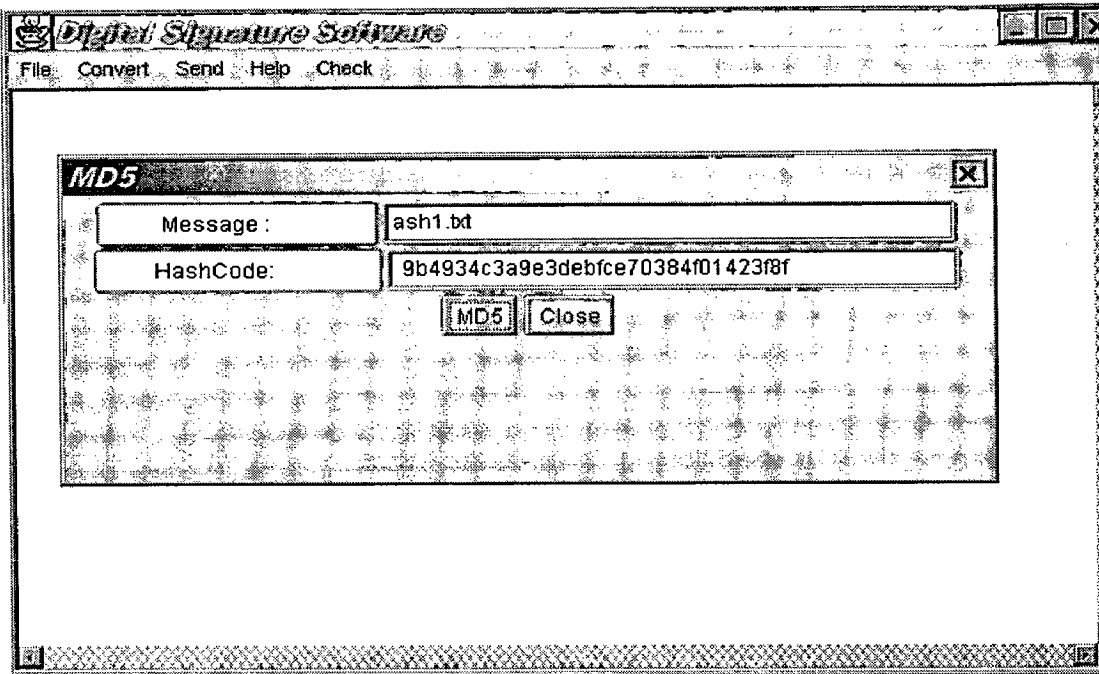


Fig 5.2: Generation of Hash Code using MD5 Algorithm

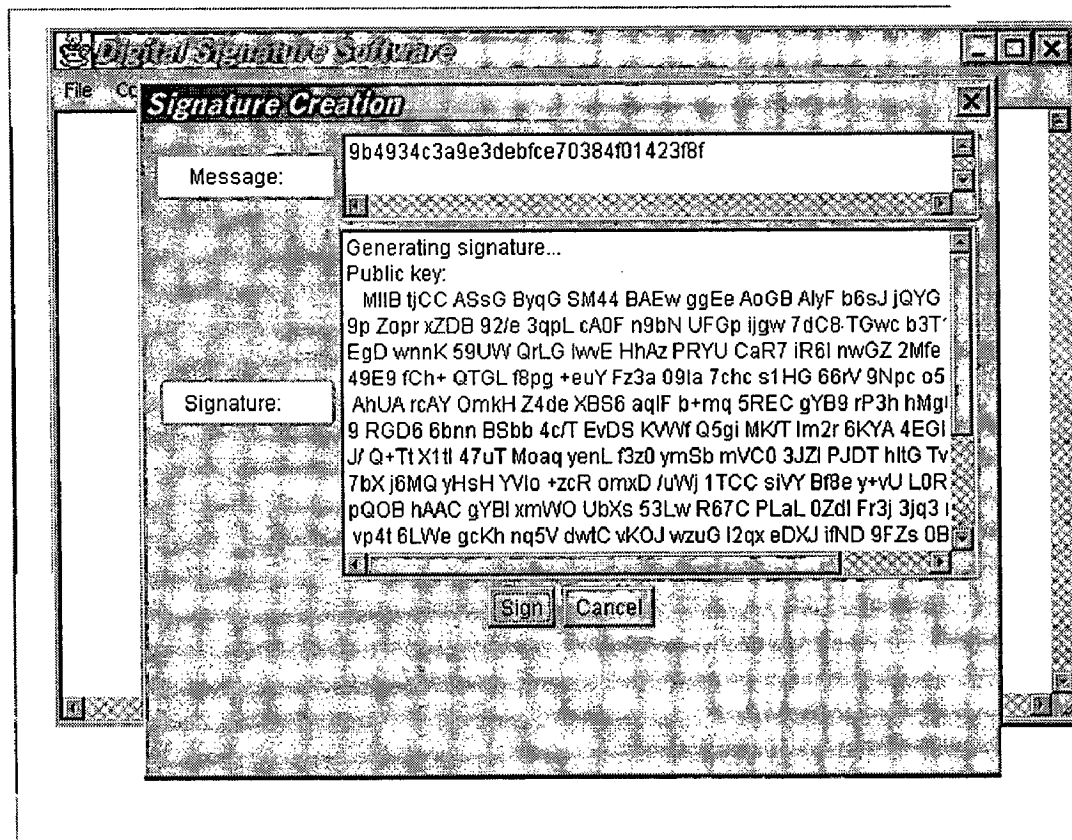


Fig 5.3: Signature Creation

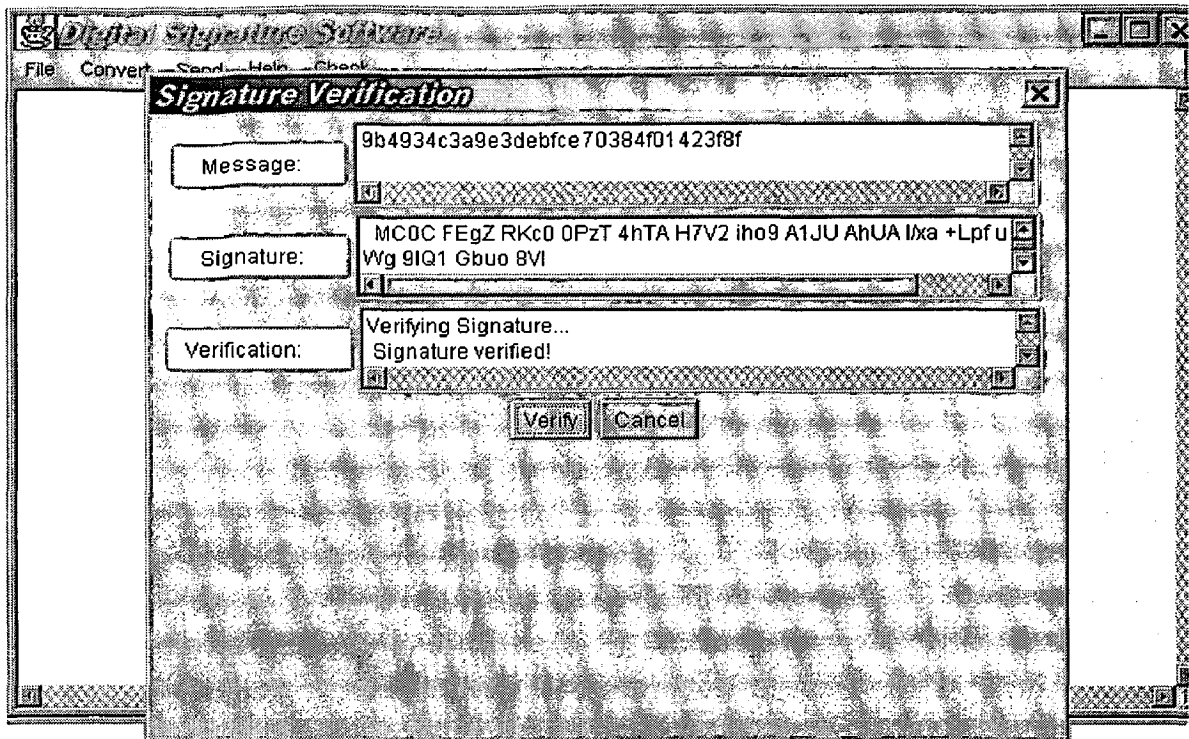


Fig 5.4: Signature Verification

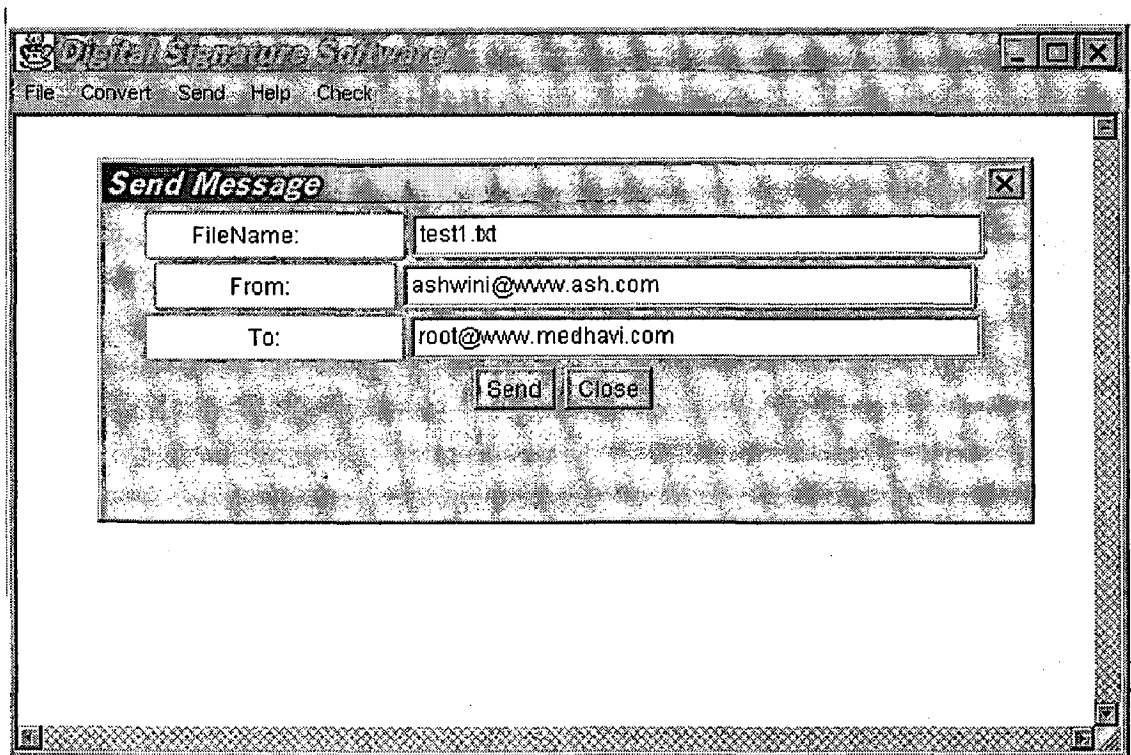


Fig 5.5: Sending Message

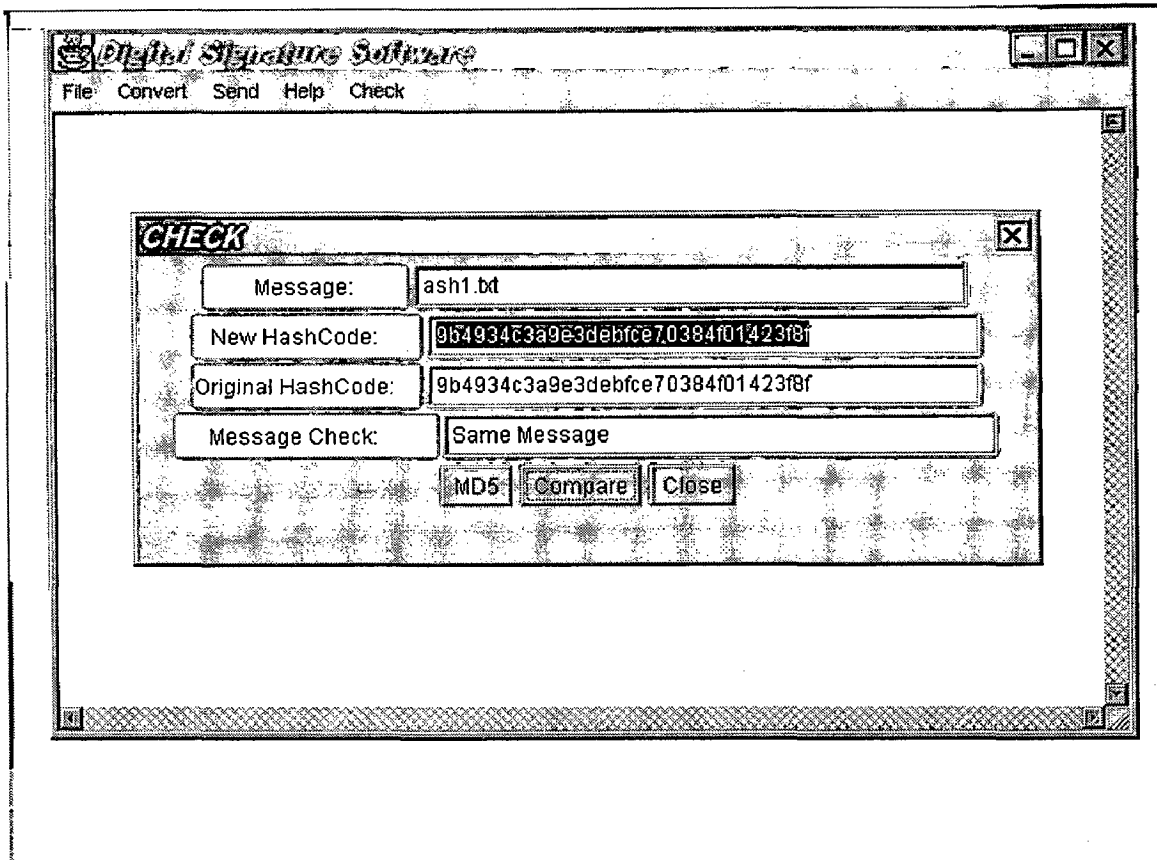


Fig 5.6: Checking Authenticity of Message Sent

Advantages of using AES algorithm

Rijndael can be used for ATM, HDTV, BISDN, Voice and Satellite. Rijndael can be implemented efficiently in software on a wide range of processors, makes use of a limited set of instructions and has sufficient parallelism to fully exploit modern pipelined multi-ALU processors, it is well suited for all mentioned applications.

The weak keys are keys that result in a block cipher mapping with detectable weaknesses. The best known case of weak keys are those of IDEA. Typically, this weakness occurs for ciphers in which the non-linear operations depend on the actual key value. This is not the case for Rijndael, where keys are applied using the EXOR and all non-linearity is in the fixed S-box. In Rijndael, there is no restriction on key selection. Despite the large amount of symmetry, care has been taken to eliminate symmetry in the behavior of the cipher. This is obtained by the round constants that are different for each round. The fact that the cipher and its inverse use different components practically eliminates the possibility for weak and semi-weak keys, as existing for DES. The non-linearity of the key expansion practically eliminates the possibility of equivalent keys.

ABSTRACT

In this dissertation, an aperture-coupled feed has been investigated for a rectangular patch antenna with U-slot. In the U-slot microstrip antenna, there are number of geometrical and electrical parameters which can critically affect the antenna characteristics. An extensive parametric study of these parameters was done through rigorous simulation on Zeland's IE3D simulation tool. During simulation, only one parameter was varied at a time keeping all others constant, which helped us in understanding the influence of each parameter in antenna characteristics. With the help of simulation, we obtained best possible antenna structure. Experimental verification of the optimized antenna was done and it is observed that simulation results are in good agreement with measurement.

Experimental results show that the antenna can attain an impedance bandwidth of 25.7% with average gain of 7 dBi, and has stable radiation characteristics over the frequency band of interest. Addition of U-slot significantly affects the gain characteristics by increasing the gain nearly upto 12.1 dBi near U-slot resonance frequency. Wide impedance characteristics can also be changed into dual-band characteristics by merely changing the stub length. To clearly visualize the effect of the addition of U-slot, a reference aperture-coupled microstrip antenna without U-slot was also studied through simulation as well as experimentally. It was found that the addition of U-slot in the microstrip patch results in an improvement of about 5% in the impedance bandwidth.

Since the aperture-coupled feed is especially attractive for array implementation, circularly polarized arrays using the proposed U-slot antenna have also been investigated by using two different feed networks. Circularly polarized radiation is obtained from an array composed of linearly polarized U-slot antenna elements having unique angular and phase arrangements. Simulation results on the array predict substantially increased impedance bandwidth (upto 38%) and average gain (upto 11.5 dBi). In other words, the array configuration has a fairly high gain-bandwidth product. Obtained axial bandwidth is 3.5 % with the minimum axial ratio as 0.4 dB.

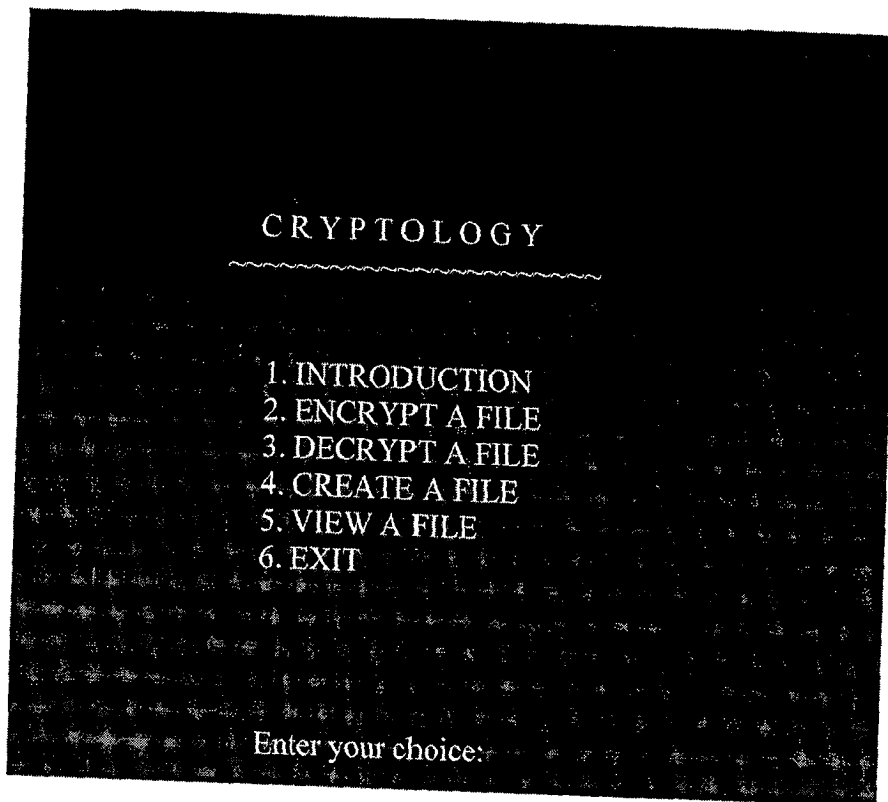


Fig 5.7: Introduction to Cryptographic Envelope Encryption Software

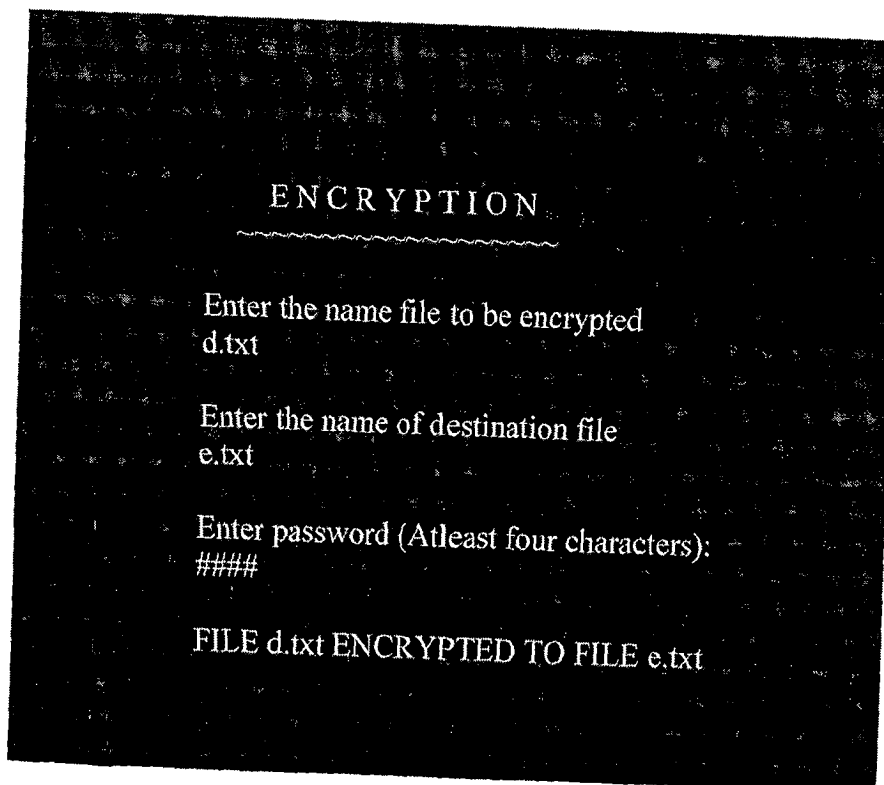


Fig 5.8: Encrypting the Cryptographic Envelope

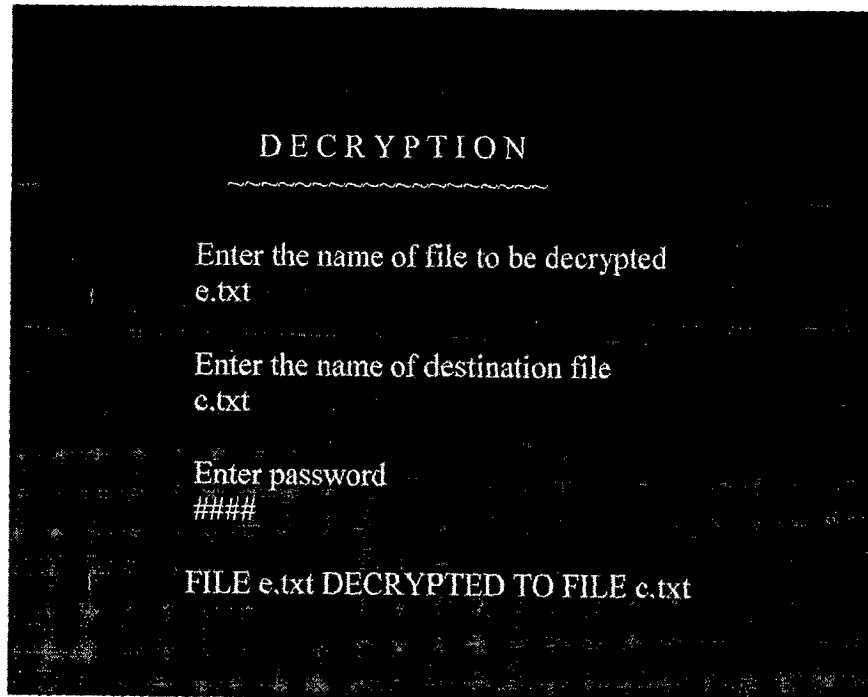


Fig 5.9: Decryption of Encrypted Cryptographic Envelope

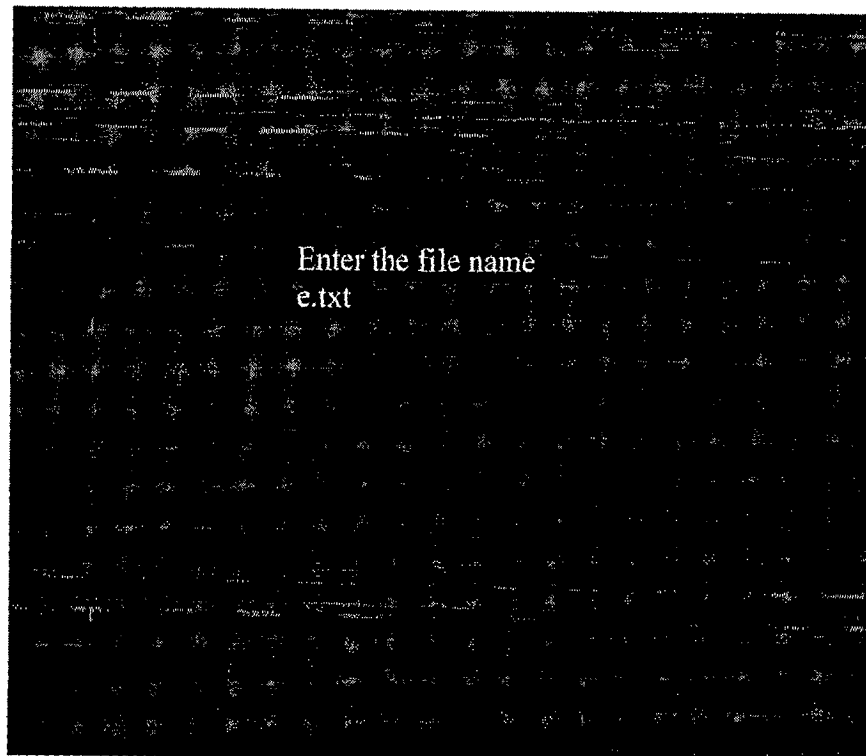


Fig 5.10: View the Output Files (Encrypted or Decrypted)

CONCLUSIONS

Several possibilities for communication security systems and their usability for digital library environments have been discussed. The common security systems, which attempt to establish a secure communication channel between the communicating parties, are useful to protect less complex, point-to-point transactions, but have several weaknesses for digital libraries. On the other hand, secure container technology seems well suited to digital libraries, where the special requirements of intellectual property protection demand specialized security services. Secure containers also seem well suited to the roles of information businesses, which tend to be increasingly executed by digital means.

In order to develop a Secure Container Technology, it is desirable to have a systematic and planned approach towards the proposed use of Security Building Blocks for Cryptographic Envelope Architecture. However to achieve this goal, a proper start is the need of hour, which would get further rectification and enhancement with the ever-exploding technologies.

This Dissertation outlines the basic methods and its real-time application in formulating Content Encapsulation Technology for Digital Library Security. However, it is prudent to mention here that this can further be enhanced and customized to match the specific requirements in the forthcoming scenarios.

REFERENCES

1. "Defence Research and Development Organization", Chapter-31, a NIC Report, <http://www.iisc.ernet.in.insa.ch31.pdf>.
2. Dr. Peter Noerr, "Digital Library Tool Kit", a Sun Microsystems White Paper, <http://www.sun.com/products-n-solutions/edu/whitepapers/pdf>, January 2003.
3. Raj Reddy, Tryg Ager, Rama Chellapa, W Bruce Croft, Beth Davis Brown, Jerry M. Mendel, Michael Ian Shamos, "Digital Information Organization in Japan", an International Technology Research Institute World Technology (WTEC) Panel Report, February 1999.
4. H.M. Gladney and J.B. Lotspiech, "Safeguarding Digital Library Contents and Users, Assuring Convenient Security and Data Quality", IBM Almaden Research Center, San Jose, California 95120-6099, D-Lib Magazine, May 1997.
5. Jack Lacy, Schuyler R. Quackenbush, Amy Reibman, James H. Snyder, "Intellectual Property Protection Systems and Digital Watermarking", an AT&T Research Paper, Florham Park, NJ.
6. F. Bartolini, R. Caldelli, V. Cappellini, A. De Rosa, A. Piva, "Digital Watermarking: a solution to Electronic Copyright Management Systems Requirements", Proceedings of 4th International Conference on Visual Systems and Multimedia, Gifu, Amsterdam, 1999.
7. M.Barni, F.Bartolini, G.Bini, V.Cappelini, A.Fringuelli, G.Meucci, A.Piva, "Enforcement of copyright laws for multimedia through blind, detectable, reversible watermarking", Proceedings of IEEE International Conference on Multimedia Computing and Systems, Florence, Italy, June 7-11, 1999.
8. Ryoichi Mori, Masaji Kawahara, "SuperDistribution: The Concept and the Architecture", Special Issue on Cryptography and Information Security, the Transactions of the IEICE; VOL.E 73, NO.7 JULY 1990.
9. S. R. White, "ABYSS: A trusted architecture for software protection", Proceedings of IEEE Symposium on Security and Privacy, Oakland, CA. pp. 38-51, April 1987.
10. D.J. Albert and S. P. Morse, "Combating software piracy by encryption and key management", IEEE Computer, pp. 68-73, April 1984.
11. S. H. Weingart, "Physical security for the uABYSS system", Proceeding 198 of

- IEEE Symposium on Security and Privacy, Oakland, CA. pp. 52-58, April 1987.
12. MC68881 Floating Point Coprocessor User's Manual, Motorola Inc., 1988.
 13. Jaworski Jamie, Perrone Paul J., "Java Security Handbook", 1st ed, Techmedia, 2001.
 14. Stallings William, "Cryptography and Network Security: Principles and Practice", 3rd ed, Prentice Hall, August 2002.
 15. Naughton Patrick, Schildt Herbert, "Java 2.0: The Complete Reference", 2nd ed, Mac-Graw Hill, 1999.
 16. Zukowski John, "Mastering Java 2, J2SE 1.4", revised ed, Sybex Inc, 2002.
 17. Husain Kamran, Parker Timothy, "Linux Unleashed", 2nd ed, Macmillan Computer Publication, 2001.
 18. Ulrich Kohl, Jeffrey Lotspiech and Stefan Nusser, "Security for the Digital Library- Protecting Documents Rather than Channels", Proceedings of IEEE International Conference on Digital Library Security, 1998 (0-8186-8353).
 19. M. Blaze, J. Feigenbaum, J. Lacy, "Decentralized Trust Management", Proceedings of IEEE Symposium on Security and Privacy, 1996.
 20. J.B.Lotspiech, U.Kohl, M.A.Kaplan: "Cryptographic Containers and the Digital Library", In Proceedings VIS '97, Vieweg Verlag, October 1997.
 21. Marc A. Kaplan: "IBM Cryptolopes, Superdistribution and Digital Rights Management", Working Paper, VI.3.0, 12/96,
<http://www.research.ibm.com/people/k/kaplan>
 22. Ron Rivest, "The MD5 Message-Digest Algorithm", RFC 1321, Network Working Group, MIT Laboratory for Computer Science and Data Security, Inc., April 1992.
 23. U.S. Department of Commerce, "Digital Signature Standard," National Institute of Standards and Technology, Federal Information Processing Standards Publication, FIPS PUB 186-2, January 27, 2000.
 24. U.S. Department of Commerce, "Advanced Encryption Standard", National Institute of Standards and Technology, Federal Information Processing Standards Publication, FIPS PUB 197, November 26, 2001.
 25. J. Daemen and V. Rijmen, "AES Proposal: Rijndael", AES Algorithm Submission, September 3, 1999, <http://www.nist.gov/CryptoToolkit>.
 26. J. Daemen and V. Rijmen, "The block cipher Rijndael", Smart Card research

and Applications, LNCS 1820, Springer-Verlag, pp. 288-296.

27. A. Lee, "Guideline for Implementing Cryptography in the Federal Government", NIST Special Publication 800-21, National Institute of Standards and Technology, November 1999.
28. A. Menezes, P. van Oorschot, S. Vanstone, "Handbook of Applied Cryptography", CRC Press, New York, 1997, p. 81-83.
29. J. Nechvatal, et. al., "Report on the Development of the Advanced Encryption Standard (AES)", National Institute of Standards and Technology, October 2, 2000.