UNIVERSITI TEKNOLOGI MALAYSIA

BORANG PENGESAHAN
LAPORAN AKHIR PENYELIDIKAN

TAJUK PROJEK : THE DEVELOPMENT OF A COMMERCIALLY VIABLE DATABASE ENCRYPTION TOOL FOR ORACLE8i RDBMS

Saya MOHD NAZRI BIN KAMA
(HURUF BESAR)

Mengaku membenarkan **Laporan Akhir Penyelidikan** ini disimpan di Perpustakaan Universiti Teknologi Malaysia dengan syarat-syarat kegunaan seperti berikut :

1. Laporan Akhir Penyelidikan ini adalah hakmilik Universiti Teknologi Malaysia.
2. Perpustakaan Universiti Teknologi Malaysia dibenarkan membuat salinan untuk tujuan rujukan sahaja.
3. Perpustakaan dibenarkan membuat penjualan salinan Laporan Akhir Penyelidikan ini bagi kategori TIDAK TERHAD.
4. * Sila tandakan (/)

[/] SULIT (Mengandungi maklumat yang berdarjah keselamatan atau Kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972).

[ ] TERHAD (Mengandungi maklumat TERHAD yang telah ditentukan oleh Organisasi/badan di mana penyelidikan dijalankan).

[ ] TIDAK TERHAD

TANDATANGAN KETUA PENYELIDIK

Nama & Cop Ketua Penyelidik

Tarikh : 18 July 2005

CATATAN : ***** *Jika Laporan Akhir Penyelidikan ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/ organisasi berkenaan dengan menyatakan sekali sebab dan tempoh laporan ini perlu dikelaskan sebagai SULIT dan TERHAD.*

UNIVERSITI TEKNOLOGI MALAYSIA
Research Management Centre

PRELIMINARY IP SCREENING & TECHNOLOGY ASSESSMENT FORM

(To be completed by Project Leader submission of Final Report to RMC or whenever IP protection arrangement is required)

1. **PROJECT TITLE IDENTIFICATION** :

The Development of A Commercially Viable Database Encryption Tool for Oracle8i RDBMS

_ Vote No: 74228

2. **PROJECT LEADER** :

Name : Mohd Nazri Bin Kama

Address :
Centre for Advanced Software Engineering (CASE), Fakulti Sains Komputer dan Sistem Maklumat,
Universiti Teknologi Malaysia, Jalan Semarak, 54100 Kuala Lumpur

Tel : 012-2198764 Fax : 03-26930933 e-mail : nazrikama@citycampus.utm.my

3. **DIRECT OUTPUT OF PROJECT** *(Please tick where applicable)*

Secientific Research	Applied Research	Product/Process Development
☐ Algorithm	☐ Method/Technique	☐ Product / Component
☐ Structure	☒ Demonstration / Prototype	☐ Process
☐ Data		☐ Software
☐ Other, please specify	☐ Other, please specify	☐ Other, please specify
_____	_____	_____
_____	_____	_____
_____	_____	_____

4. INTELLECTUAL PROPERTY *(Please tick where applicable)*

☒ Not patentable	☐ Technology protected by patents
☒ Patent search required	☐ Patent pending
☐ Patent search completed and clean	☐ Monograph available
☐ Invention remains confidential	☐ Inventor technology champion
☐ No publications pending	☐ Inventor team player
☐ No prior claims to the technology	☐ Industrial partner identified

5. LIST OF EQUIPMENT BOUGHT USING THIS VOT

Notebook – 2 Units

Desktop – 1 Unit

Printer – 1 Unit

6. STATEMENT OF ACCOUNT

a)	APPROVED FUNDING	RM : 70,500
b)	TOTAL SPENDING	RM : 70,500
c)	BALANCE	RM : NIL

7. TECHNICAL DESCRIPTION AND PERSPECTIVE

Please tick an executive summary of the new technology product, process, etc., describing how it works. Include brief analysis that compares it with competitive technology and signals the one that it may replace. Identify potential technology user group and the strategic means for exploitation.

a) Technology Description

Refer to Appendix A

b) Market Potential

Refer to Appendix B

c) Commercialization Strategies

Refer to Appendix C

8. RESEARCH PERFORMANCE EVALUATION

a) FACULTY RESEARCH COORDINATOR

Research Status	( )	( )	( )	( )	( )	( )
Spending	( )	( )	( )	( )	( )	( )
Overall Status	( )	( )	( )	( )	( )	( )
	Excellent	Very Good	Good	Satisfactory	Fair	Weak

Comment/Recommendations:

……

Signature and stamp of
JKPP Chairman

Name : Mohd Nazri Bin Kama

Date : 18 July 2005

b) RMC EVALUATION

Research Status	( )	( )	( )	( )	( )	( )
Spending	( )	( )	( )	( )	( )	( )
Overall Status	( )	( )	( )	( )	( )	( )
	Excellent	Very Good	Good	Satisfactory	Fair	Weak

Comments :-

_____

_____

_____

_____

_____

_____

Recommendations :

☐ Needs further research

☐ Patent application recommended

☐ Market without patent

☐ No tangible product. Report to be filed as reference

…….. Name : ……

Signature and Stamp of Dean / Deputy Dean Date : ……..
Research Management Centre

Benefits Report Guidelines

A. Purpose

The purpose of the Benefits Report is to allow the IRPA Panels and their supporting experts to assess the benefits derived from IRPA-funded research projects.

B. Information Required

The Project Leader is required to provide information on the results of the research project, specifically in the following areas:

- Direct outputs of the project;
- Organisational outcomes of the project; and
- Sectoral/national impacts of the project.

C. Responsibility

The Benefits Report should be completed by the Project Leader of the IRPA-funded project.

D. Timing

The Benefits Report is to be completed within three months of notification by the IRPA Secretariat. Only IRPA-funded projects identified by MPKSN are subject to this review. Generally, the Secretariat will notify Project Leaders of selected projects within 18 months of project completion.

E. Submissin Procedure

One copy of this report is to be mailed to :

IRPA Secretariat
Ministry of Science, Technology and the Environment
14th, Floor, Wisma Sime Darby
Jalan Raja Laut
55662 Kuala Lumpur

Benefit Report

1. Description of the Project

A. Project identification: 04-02-06-0086-EA001

1. Project number : 74228

2. Project title : The Development of A Commercially Viable Database Encryption Tool for Oracle8i RDBMS

3. Project leader : Mohd Nazri Bin Kama

B. Type of research

Indicate the type of research of the project (Please see definitions in the Guidelines for completing the Application Form)

[ ] Scientific research (fundamental research)

[x] Technology development (applied research)

[ ] Product/process development (design and engineering)

[ ] Social/policy research

C. Objectives of the project

1. Socio-economic objectives

Which socio-economic objectives are adressed by the project? (Please indentify the sector, SEO Category and SEO Group under which the project falls. Refer to the Malaysian R&D Classification System brochure for the SEO Group code)

Sector : Services and IT (04)

SEO Category : Information and Communication Services (S20900)

SEO Group and Code : Computer Software and Services (S20901)

2. Fields of research

Which are the two main FOR Categories, FOR Groups, and FOR Areas of your project? (Please refer to the Malaysia R&D Classification System brochure for the FOR Group Code)

a. Primary field of research

FOR Category : F10500 Information, Computer and Communication Technology

FOR Group and Code : F10501 Information Systems

FOR Area : Cryptography, Encryption and Security Systems

b. Secondary field of research

FOR Category : F10500 Information, Computer and Communication Technology

FOR Group and Code : F10506 Security System

FOR Area : Authentication System - Databases

D. Project duration

What was the duration of the project ?

Eighteen (18) _____ Months

E. Project manpower

How many man-months did the project involve?

Fourty Four (44) _____Man-months

F. Project costs

What were the total project expenses of the project?

RM_ 70,500 _

G. Project funding

Which were the funding sources for the project?

Funding sources	Total Allocation (RM)
_*IRPA*_____	70,500 _____
_____	_____
_____	_____
_____	_____

ll. Direct Outputs of the Project

A. Technical contribution of the project

1. **What was the achieved direct output of the project :**

For scientific (fundamental) research projects?

- [ ] Algorithm
- [ ] Structure
- [ ] Data
- [ ] Other, please specify : _____

For technology development (applied research) projects :

- [ ] Method/technique
- [x] Demonstrator/prototype
- [ ] Other, please specify : _____

For product/process development (design and engineering) projects:

- [ ] Product/component
- [ ] Process
- [ ] Software
- [ ] Other, please specify : _____

2. How would you characterise the quality of this output?

- [x] Significant breakthrough
- [ ] Major improvement
- [ ] Minor improvement

B. Contribution of the project to knowledge

1. How has the output of the project been documented?

- [ ] Detailed project report
- [x] Product/process specification documents
- [ ] Other, please specify : _____

2. Did the project create an intellectual property stock?

- [ ] Patent obtained
- [ ] Patent pending
- [x] Patent application will be filed
- [ ] Copyright

3. What publications are available?

- [ ] Articles (s) in scientific publications — How Many: _____
- [x] Papers(s) delivered at conferences/seminars — How Many: 1
- [ ] Book
- [ ] Other, please specify : _____

4. How significant are citations of the results?

- [ ] Citations in national publications — How Many: _____
- [x] Citations in international publications — How Many: 1
- [ ] None yet
- [ ] Not known

lll. Organisational Outcomes of the Project

A. Contribution of the project to expertise development

1. How did the project contribute to expertise?

- [ ] PhD degrees — How Many: _____
- [ ] MSc degrees — How Many: _____
- [ ] Research staff with new specialty — How Many: 2
- [ ] Other, please specify: _____

2. How significant is this expertise?

- [x] One of the key areas of priority for Malaysia
- [ ] An important area, but not a priority one

B. Economic contribution of the project?

1. How has the economic contribution of the project materialised?

- [ ] Sales of manufactured product/equipment
- [x] Royalties from licensing
- [ ] Cost savings
- [ ] Time savings
- [ ] Other, please specify : _____

2. How important is this economic contribution ?

- [x] High economic contribution — Value: RM 500,000
- [ ] Medium economic contribution — Value: RM_____
- [ ] Low economic contribution — Value: RM_____

3. When has this economic contribution materialised?

[ ] Already materialised

[ ] Within months of project completion

[x] Within three years of project completion

[ ] Expected in three years or more

[ ] Unknown

C Infrastructural contribution of the project

1. What infrastructural contribution has the project had?

[ ] New equipment Value: RM _____

[ ] New/improved facility Investment : RM _____

[ ] New information networks

[x] Other, please specify: ___ New scheme for DB Encr and Decryr _____

2. How significant is this infrastructural contribution for the organisation?

[ ] Not significant/does not leverage other projects

[x] Moderately significant

[ ] Very significant/significantly leverages other projects

D. Contribution of the project to the organisation's reputation

1. How has the project contributed to increasing the reputation of the organisation

[x] Recognition as a Centre of Excellence

[ ] National award

[ ] International award

[x] Demand for advisory services

[x] Invitations to give speeches on conferences

[x] Visits from other organisations

[ ] Other, please specify: _____

2. How important is the project's contribution to the organisation's reputation ?

- [ ] Not significant
- [ ] Moderately significant
- [x] Very significant

1V. National Impacts of the Project

A. Contribution of the project to organisational linkages

1. Which kinds of linkages did the project create?

[x] Domestic industry linkages

[ ] International industry linkages

[x] Linkages with domestic research institutions, universities

[ ] Linkages with international research institutions, universities

2. What is the nature of the linkages?

[ ] Staff exchanges

[ ] Inter-organisational project team

[x] Research contract with a commercial client

[x] Informal consultation

[ ] Other, please specify: _____

B. Social-economic contribution of the project

1. Who are the direct customer/beneficiaries of the project output?

Customers/beneficiaries:	Number:
Security of public databases like JPJ	_____
Financial and banking databases	_____
Malaysian Arm Forces and Police databases	

2. How has/will the socio-economic contribution of the project materialised ?

[ ] Improvements in health

[ ] Improvements in safety

[ ] Improvements in the environment

[ ] Improvements in energy consumption/supply

[ ] Improvements in international relations

[x] Other, please specify: Economic contribution in the locally developed

3. How important is this socio-economic contribution?

[x] High social contribution

[ ] Medium social contribution

[ ] Low social contribution

4. When has/will this social contribution materialised?

[ ] Already materialised

[x] Within three years of project completion

[ ] Expected in three years or more

[ ] Unknown

Date: 18 July 2005 **Signature:** 

VOT 74228

THE DEVELOPMENT OF A COMMERCIALLY VIABLE DATABASE ENCRYPTION TOOL FOR ORACLE8i RDBMS

(PEMBANGUNAN ALATAN ENKRIPSI PANGKALAN DATA KOMERSIL BAGI ORACLE8i RDBMS)

MOHD NAZRI BIN KAMA

CENTRE FOR ADVANCED SOFTWARE ENGINEERING (CASE)
FAKULTI SAINS KOMPUTER DAN SISTEM MAKLUMAT
UNIVERSITI TEKNOLOGI MALAYSIA

2005

VOT 74228

THE DEVELOPMENT OF A COMMERCIALLY VIABLE DATABASE ENCRYPTION TOOL FOR ORACLE8i RDBMS

(PEMBANGUNAN ALATAN ENKRIPSI PANGKALAN DATA KOMERSIL BAGI ORACLE8i RDBMS)

MOHD NAZRI BIN KAMA

RESSEARCH VOTE NO:
74228

Centre For Advanced Software Engineering (Case)
Fakulti Sains Komputer Dan Sistem Maklumat
Universiti Teknologi Malaysia

2005

ABSTRACT

In database security, access control is a major research issue. Discretionary access controls have been handled well by many database management systems through user roles and privileges. Mandatory access controls, on the other hand, remains a big problem when users with lower security clearance accessing data of higher security class. Data with classifications and users have clearances developed multilevel access controls, thus the problem of multilevel security. Many researches have been conducted using methods like object labeling, trusted systems, security filters, database views and etc. Many a times the problem remains unsolved due to either too theoretical or not practical to be implemented. Recent developments in research showed cryptography to be the promising solution to the multilevel security problem. With appropriate key management and good multilevel security scheme design, the problem can be solved in both theory and implemented in practice. This research endeavor is one such effort. It presents an investigation into the applications of modern cryptography for the security of databases. The investigation yields a new multilevel security scheme based on indigenous cryptographic primitives and supported by a new key management technique. The cryptographic primitives include enhanced block cipher and a new stream cipher design successfully implemented in a commercial database. The system yields a new approach in accessing and processing encrypted data using Initialization Vectors and provides solutions for hierarchical and direct access controls. The novel scheme allows the encryption of data at the tuple, attribute, and data element levels of a relation. The security of the scheme is guaranteed with no keys present in the system but stored securely in smartcards. The outcome from this research is realized in OraCrypt application which is implemented by usign Oracle 8i RDBMS.

Key researchers:

Mr. Mohd Nazri Kama

Assc. Prof. Dr. Zailani Mohamed Sidek

E-mail : nazrikama@citycampus.utm.my

Tel No : 03-26154344

Vot No : 74228

ABSTRAK

Dalam keselamatan pangkalan data, kawalan capaian merupakan satu isu penyelidikan yang besar. Kawalan capaian "budi bicara" telah ditangani dengan baik oleh sistem pengurusan pangkalan data menggunakan peranan dan hak istimewa pengguna. Kawalan capaian mandatori pula masih memberi masalah besar apabila pengguna yang mempunyai kebenaran keselamatan lebih rendah cuba mencapai data yang mempunyai kelas keselamtan lebih tinggi. Data mempunyai kelas dan pengguna mempunyai kebenaran tertentu menyebabkan kawalan capaian menjadi berbilang aras dan masalah keselamatan berbilang aras. Banyak penyelidikan dijalankan untuk menyelesaikan masalah seperti menggunakan kaedah melabel objek, sistem beramanah, penapis keselamatan, pandangan pangkalan data, dan lain-lain. Dalam banyak keadaan, masalah ini tidak dapat diselesaikan samada kerana terlalu teori atau tidak praktik untuk dilaksanakan. Perkembangan terkini penyelidikan menunjukkan kriptografi adalah satu penyelesaian yang baik kepada masalah ini. Dengan sistem pengurusan kunci yang sesuai dan skim keselamatan berbilang aras yang baik, masalah ini dapat diselesaikan secara teori dan praktik. Penyelidikan ini merupakan satu usaha ke arah ini menggunakan kriptografi bagi menyelesaikan masalah keselamatan dalam pangkalan data. Penyiasatan ini menghasilkan satu skim keselamatan berbilang aras yang baru menggunakan primitif kriptografi asli dan disokong oleh satu teknik pengurusan kunci baru. Primitif kriptografi ini termasuk sifer blok dipertingkatkan dan satu rekabentuk baru strim sifer yang telah dilaksanakan dengan jayanya dalam satu pangkalan data perdagangan. Sistem memberi satu pendekatan pencapaian dan pemprosesan data tersulit menggunakan Vektor Peng-awalan data. Ia mampu memberi penyelesaian kawalan capaian data secara terus dan berhirarki pada tiga peringkat iaitu baris, lajur, dan unsur data. Keselamatan skim ini dijamin kerana kunci disimpan dalam kad pintar.

Hasil kajian penyelidikan ini direalisasikan di dalam satu aplikasi yang iaitu OraCrypt yang menggunakan pangkalan data Oracle8i RDBMS.

Penyelidik Utama:

Mr. Mohd Nazri Kama
Assc. Prof. Dr. Zailani Mohamed Sidek

E-mail : nazrikama@citycampus.utm.my
Tel No : 03-26154344
Vot No : 74228

TABLE OF CONTENTS

LIST OF TABLES

TABLE NO TITLE PAGE

LIST OF FIGURES

FIGURE NO TITLE PAGE

LIST OF APPENDICES

APPENDIX NO TITLE PAGE

LIST OF ABBREVIATIONS

CASE	-	Center for Advanced Software Engineering
DBMS	-	Database Management Systems
DL	-	Digital Library
DLL	-	Dynamic Link Library
GUI	-	Graphical User Interface
HLIV	-	Hashed Left Data IV
HRIV	-	Hashed Right Data IV
HUIV	-	Hashed User IV
IRPA	-	Intensity Research and Priority Area
ITK	-	Institut Teknologi Kebangsaan
IV	-	Initialization Vector
I/O	-	Input/Output
LIV and RIV	-	Left and Right IVs
MINDEF	-	Ministry of Defense
OO	-	Object-Oriented
OS	-	Operating System
RDBMS	-	Relational Database Management System
RUP	-	Rational Unified Process
SAD	-	Software Architecture Document
SDK	-	Software Development Kit
SDP	-	Software Development Plan
SRS	-	Software Requirements Specification
TS	-	Tuan Sabri

TSBC	-	TS Block Cipher
TSSC	-	TS Stream Cipher
UML	-	Unified Modeling Language
UTM	-	University of Technology Malaysia

LIST OF TERMINOLOGY

Actor	Someone or something, outside the system that interacts with the system.
Authentication	The process of verifying the identity of a user, device, or other entity in a computer system, often as a prerequisite to allowing access to resources in a system.
Authorization	Permission given to a user, program, or process to access an object or set of objects. In Oracle, authorization is done through the role mechanism. A single person or a group of people can be granted a role or a group of roles. A role, in turn, can be granted other roles.
Availability (of Systems)	Preventing, detecting and/or deterring improper denial of access to services provided by the system.
Beta Testing	Pre-release testing in which a sampling of the intended customer base tries out the product.
Codes	A system for hiding the meaning of a message by replacing each word or phrase in the original message with another character or set of characters.
Computer Security	Protection of information processed by a computer against unauthorized observation, unauthorized or improper modification, and denial of service (ensuring no authorized use of the information is denied).
Construction Phase	The third phase of the Unified Process, in which the software is brought from an executable architectural baseline to the point at which it is ready to be transitioned to the user community.
Cryptography	Mathematical manipulation of data, which the purposes are for reversible or irreversible transformation. The act of writing and deciphering secret code resulting in secure messages.

Data Encryption	The encoding of data so that it cannot be easily deciphered without the use of specialized decryption mechanisms or procedures.
Database	A collection of data, arranged in some meaningful way.
Database Security (DB Security)	A set of measures, policies and mechanisms to provide secrecy, integrity and availability of data and to combat possible attacks on the system (threats) from insiders and outsiders, both malicious and accidental.
Database Management System (DBMS)	A software/program that organizes and operates on the database and auxiliary control information to implement the decisions of the access policy and gives users the means to retrieve information.
Decryption	The process of converting the contents of an encrypted message (ciphertext) back into its original readable format (plaintext).
Dynamic Link Library	A file containing executable code and data bound to a program at run time rather than at link time. The C++ Access Builder generates beans and C++ wrappers that let your Java programs access C++ DLLs.
Elaboration Phase	The second phase of the process where the product vision and its architecture are defined.
Encryption	The process of disguising a message rendering it unreadable to anybody but the intended recipient.
Inception Phase	The first phase of the Unified Process, in which the seed idea, request for proposal, for the previous generation is brought to the point of being (at least internally) funded to enter the elaboration phase.
Phase	The time between two major project milestones, during which a well-defined set of objectives is met, artifacts are completed, and decisions are made to move or not move into the next phase.
Security Threats	A hostile agent that, either casually or by using a specialized technique, can disclose or modify the information managed by a security system.

Smartcard	A plastic card (like a credit card) with an embedded integrated circuit for storing information, including such information as user names and passwords. A smartcard is read by a hardware device at any client or server.
Transition Phase	The fourth phase of the process in which the software is turned over to the user community. A relationship between two states indicating that an object in the first state will perform certain specified actions and enters the second state when a specified event occurs and specified conditions are satisfied. On such a change of state, the transition is said to execute.
TS Block Cipher	Block cipher can process data in blocks of 16 characters at a time.
TS Stream Cipher	Stream cipher can process 255 characters of data at a time.
Unified Modeling Language	A language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system.
User Authentication	The authorization mechanism that uniquely identify the database users before allowing them to access the data.

CHAPTER 1

INTRODUCTION

1.1 Introduction

This chapter describes the introduction of the project in which the researcher has been involved in order to fulfill the requirements for the new technology found for the Research Management Centre (RMC). The project has been operated in eighteen (18) months duration time starting from 14th December 2003 until 14th June 2005.

1.2 Company Background

This section describes the background of the institution that the researcher has involved during the project research and development.

1.2.1 Universiti Teknologi Malaysia

Universiti Teknologi Malaysia (UTM) was established on 14th March 1972 under the name Institut Teknologi Kebangsaan (ITK). On 1st April 1975, this technical school had undergone many reformations and officially known as UTM. UTM is a prestigious university at the frontier of technology with a vision to lead in the development of

creative human resource and technology in line with the aspirations of the nation. It is one of Malaysia's leading universities for engineering and technology, which has:

i. A reputation for innovative education and leading-edge research, educating the technologies and professional
ii. More than 25,000 students at campus in Johor, more than 4,500 students in Kuala Lumpur campus, and more than 5,000 students on distance learning/part-time programmes
iii. More then 2,500 postgraduates students in various fields of specialization
iv. More than 20 specialist institutes and centre, in addition to academic departments to service the technology, education and research needs.

UTM strives for academic excellence through creative learning and state-of-the-art technology. UTM has 2 campuses, the 1,222-hectare main campus in Skudai, Johor and 18-hectare branch campus situated at Jalan Semarak, Kuala Lumpur.

1.2.2 Centre For Advanced Software Engineering (CASE)

The Centre for Advanced Software Engineering (CASE) was established in 1996 as a joint-venture programme between Universiti Teknologi Malaysia (UTM) and Universite Thales, France for enabling the technology transfer into Malaysian industry, especially the software industry. CASE is committed in providing opportunities for advanced studies and professional development for the current and future needs of technology based industries. At CASE, the aim is to create opportunities for local industries or individuals to be trained in important aspects of Information Security and Software Development via the following programmes:

i. Masters and Doctor of Philosophy in Information Security
ii. Master and Doctor of Philosophy in Software Engineering
iii. Continuing Education/Short Courses

In meeting these goals, CASE promotes its expertise and technical capabilities in its effort to become a centre of excellent, majoring in the following specific areas:

i. Research and Development
ii. Software Consultancy and Mentoring
iii. Partnership in Software Development Software engineering project and software methodology
iv. Software engineering techniques and tools

1.2.3 Project Team

OraCrypt™ project team consists of four (4) staff working closely with each other in research and development activities at CASE. The project was funded by Intensification of Research Priority Areas (IRPA) and managed by the Research Management Centre (RMC). RMC is responsible to manage the research and development activities conducted by UTM researchers that sometimes collaborate with the industries outside the UTM. It also regulates the grant for the research project conducted by UTM researchers.

The project team consists of one (1) Project Advisor, one (1) Project Leader, and two (2) System Analysts. *Figure 1.1* illustrates the project organization.

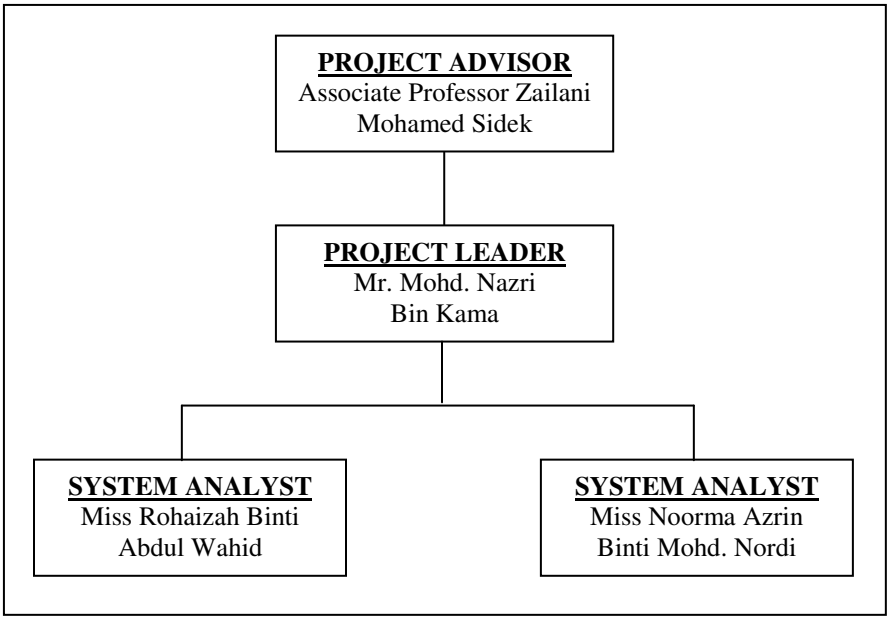


Figure 1.1 OraCrypt™ Project Organizations

Table 1.1 describes the role represented in the project organization diagram above and their primary responsibilities.

Table 1.1 Lists Of Roles Respective Responsibilities

Role	List of Responsibilities
Project Advisor	▪ Advise on the overall project contents and the products that should be delivered on a regular period of time. ▪ Motivate the team members to perform their tasks. ▪ Help the team in allocating the tasks and resolving issues.
Project Leader	▪ Plan and regulate the tasks. ▪ Approve the process development. ▪ Schedule and check the milestone for review and evaluation of the product, tasks and software process. ▪ Maintain a project plan. ▪ Lead the team in producing the development strategy. ▪ Lead the team in producing the preliminary size and time estimates for the products to be produced. ▪ Lead the team in producing the high-level design and design specification. ▪ Motivate the team members to perform their tasks.

Role	List of Responsibilities
	▪ Help the team in allocating the tasks and resolving issues.
System Analyst	▪ Define the responsibilities, operations, attributes, and relationship of one or several classes, and determine how they will be adjusted to the implementation environment. ▪ Devise a coding standard to be used during the implementation process. ▪ Implement the design defined in the SDD. ▪ Strive to produce code that is readable and bug-free. ▪ Develop and test components, in accordance with the projects adopted standards, for integration purposes. ▪ Establish and maintain the team's development standards.

1.3 Project Background

The project is being done under the Intensification of Research and Priority Area (IRPA) project at UTM. The project was initially a work of a doctorate student Associate Professor Zailani Bin Mohamed Sidek of CASE. The research conducted is mainly on database security specifically in the area of cryptography. He argued that authentication, access control, and audit together provide the foundation for information and system security. The database security mechanism should be based on cryptographic technology, which is independent, possibly developed locally, and is based on strong theoretic foundation. With this in mind, efforts are made to serve the purpose of being independent, secure and proactive. OraCrypt™ product is one of the solutions. In this project, the researcher take the approach of building a prototype with highly secure, independent, and implementable database security mechanism for a commercially, widely-used relational database management system.

1.3.1 Project Overview

OraCrypt™ is a product for the encryption and decryption of data in Oracle8i RDBMSs. The product consists of two (2) main modules that are Key Generation, Management and Distribution module, and Encryption and Decryption module. The key

generation, management and distribution module allows the generation of Master and Shared keys. Master keys are for the hierarchical access control to encrypt data while Shared keys are useful for both direct access and hierarchical access controls. The encryption and decryption module is designed with no keys (Master or Shared keys) to be found or stored in the database Management System (DBMS) concern. Instead all keys are stored on smartcard. In this module, it allows users to encrypt and decrypt data at all levels (row, column, and data element). And in each level, users may provide condition for the selective encryption and decryption of data at those three levels.

1.3.2 Project Vision

OraCrypt™ product is expected to become a commercial data encryption software package within a Relational Database Management System (RDBMS). In this project, the implementation is focused on the Oracle Database Management System version 8i only.

1.3.3 Project Objective(s)

The objective of this effort is to develop a prototype software security system that:

i. Design and implement a database security mechanism using independent and local cryptographic technology that will enhance the security of the database for Oracle8i RDBMS.
ii. Implement the new design for IV-based multilevel database encryption scheme in Oracle8i RDBMS environment.
iii. Develop and implement cryptographic key generation, management and distribution system.
iv. Implement the OraCrypt™ product evaluation and validation.

1.3.4 Project Scope(s)

The scopes of this project are as follows:

i. Identify and examine locally developed cryptographic algorithms that are available and suitable in the proposed database security application.
ii. Analyze and implement the new IV-based multilevel database for encryption and decryption.
iii. Implement the cryptographic key generation, management and distribution scheme.
iv. Analyze, modify, and implement the cryptographic algorithms in column encryption-decryption module.
v. Analyze, design and formulate working models and solution procedures for database encryption and decryption. The working models and solution procedures are expected to be better in terms of its security and performance when compared to the existing schemes available.

1.3.5 Project Schedule

This project was scheduled for eighteen (18) months. Refer to *Appendix A* for the details of the project's schedule.

1.4 Problem Background

This section looks into some key definitions on the issue of computer security and database security. The researcher describes the security goals and its design decisions/principles, the specific threats to database security, and the requirements for database security.

1.4.1 Computer Security Definition

Computer security [Schaefer (1993)] has come to a stage where its definition needs to be rethinking due to changes in the environment and the advancement of technologies. Blakley (1996) argues that the traditional model of computer security is no longer viable, and the new definitions of the security problem are needed before the industry can begin to work toward effective security in the new environment. In addition, Schaefer (1993) agrees that much of computer security has been reduced to practice but it is also important to understand that there are no general closed form solutions to computer security problems. This means there is no generally applicable technique that applies to securing arbitrary system environments.

They are few definitions of computer security for the sake of discussions. Castano, *et. al*. (1995), defines computer security as protection of information processed by a computer against unauthorized observation, unauthorized or improper modification, and denial of service vis-à-vis ensuring no authorized use of the information is denied. Gollmann (1999) provides a simpler definition of computer security that deals with the prevention and detection of unauthorized actions by users of a computer system. Schneier (1998) describes security as about risk management that detection and response are just as important as prevention, and that reducing the *window of exposure* for an enterprise is security's real purpose.

1.4.2 Security Goals

The three goals of secure computing are confidentiality, integrity, and availability [Pfleeger (2000), Gollmann (1999), Abrams *et. al*. (1995), Pernul (1994)]. Confidentiality means that assets of a computing system are accessible only by authorized parties. It is also called secrecy or privacy. Integrity means that assets can be modified only by authorized parties or only in authorized ways. Availability means

assets are accessible to authorized parties. It is sometimes known by its opposite, denial of service. These three goals can overlap, and they can even be mutually exclusive.

1.4.3 Database Security Definition

Database security comprises a set of measures, policies, and mechanisms to provide secrecy, integrity and availability of data and to combat possible attacks on the system from insiders and outsiders, both malicious and accidental [Castano *et. al*. 1995), Ciechanowicz (2001), Al-Salqan *et. al*. (1996)]. Database security encompasses physical, logical and organizational issues. Physical database security focuses on tools, devices, and hardware or software techniques able to prevent or detect unauthorized physical accesses to data storage facilities, and to provide database backup and/or recovery. Logical database security consists of control measures, models and techniques to prevent, detect, or deter unauthorized logical accesses to data. Organizational database security concentrates on management constraints, operational procedures, and supplementary controls established to provide database protection.

Secrecy has the objective of keeping information unreadable for outsiders while making it available for authorized users. It also means preventing, detecting, or deterring the improper disclosure of information and is very much relevant to protection of data in highly protected environments, such as the military and commercial environments. Related to this is privacy and confidentiality. Privacy reflects information about individuals or legal entities and is normally protected by laws and rules in many countries. Confidentiality is a service used to keep the content of information from all but those authorized to have it [Menezes *et. al*. (1997)]. Data integrity covers methods and techniques, which addresses the unauthorized alteration of data. To ensure data integrity, Smith (1989) specify that one or more fields of a DBMS table are designated as a unique, non null primary key, thereby ensuring that duplicate rows do not occur within one table. While Sandhu and Jajodia (1990) describes ensuring integrity of information as means of preventing/detecting/deterring the

improper modification of information. Availability of data means to ensure authorized users can have access to data when they require.

1.4.4 Threats To Database Security

A threat can be defined as a hostile agent who intentionally or unintentionally gains an unauthorized access to the protected database resources and able to disclose or modify the information managed by a system [Castano *et. al*. (1995)]. Dastjerdi, et.al (1996) classify threats into physical and logical threats. Physical threats are violations to databases in the form of a forced disclosure of passwords, a theft, a destruction of physical storage devices to a power failure, and any form of natural disaster. The methods and techniques for this type of database protection include restriction of physical access to database storage facilities and the database backup and recovery. Logical threats involve unauthorized logical access to information in databases via software. They may result in information disclosure, integrity violation, and inaccessibility to an authorized individual.

These threats may occur intentionally or by accident. Accidental threats include natural disasters like earthquakes, flood and fire, or by human error, and hardware and software failures. Malicious attacks are always intentionally carried out by disgruntled employees or abused of privileges by authorized users. These hostile users execute their acts through some hidden codes within some legitimate functions in order to violate the security of the system. Examples of such codes are Trojan Horses, trapdoors, and viruses.

1.4.5 Security Requirements For Database System

In the last section, the researcher looked at some possible threats to database security. With this understanding, the researcher selects or develops appropriate security policies and mechanisms for controlling or eliminating those threats or malicious attacks

on protected data. Security policies are policies, which outline an organization's position on security issues [Theriault and Heney (1998)]. The security policy is made more concrete with the mechanism necessary to implement the requirements of the policy [Jajodia (1996)]. Security mechanisms define how the security system should achieve the security goals. Following is a list of requirements for security of database systems [Pfleeger (2000), Castano *et. al*. (1995)]:

- Physical database integrity – the data in the database is immune to physical problems such as power failures and it can be reconstructed if destroyed through a catastrophe.
- Logical database integrity – the structure of the database is preserved. With logical integrity of a database, a modification to the value of one field does not affect other fields.
- Element integrity – the data contained in each element is accurate.
- Auditability – able to track who has accessed the elements in the database.
- Access control – a user is allowed to access only authorized data and different users can be restricted to different modes of access (read or write).
- User authentication – ensure that every user is positively identified, both for the audit trail and for permission to access certain data.
- Availability – users can access the database in general and all the data for which they are authorized.
- Encryption – ensures that only the intended user can decipher any data processed.

1.5 Problem Statement

According to Sandhu and Samarati (1996), authentication, access control, and audit together provide the foundation for information and system security. The basic premise is to have some form of protection for the precious computer and information resources. With the advent of the Internet, access to data and information has been greatly enhanced and this caused a major concern for safety and security of the

resources. The Internet has brought a borderless world of today and the need for adequate security mechanisms is becoming increasingly critical especially with the ever-rising rate of security breaches. Firewall seems to be a popular solution. However, firewalls are not immune to penetrations; once an outsider is successful in penetrating a system, they typically do not provide any protection to internal resources. We may only view Firewalls as our first line of defense since they do not protect against illegal actions by insiders, an organization's authorized users. Many security experts are of the opinion that most of the computer related crimes are done by insiders.

Protection of internal resources, though not a trivial task, needs to be supported by appropriate tools and techniques. Though there are many security measures, better protection of computer information can be obtained if several of these measures are applied simultaneously whenever possible. Such an approach is especially necessary when protection of databases is concerned. We have been introduced to the kinds of threats to database security and the security requirements of database systems against these threats. This research then is an attempt to contribute to this effect. The main statement of the research problem is:

> What is a "simple" cryptographic database security mechanism, with due respect to all security requirements and constraints, that is applicable and implemental in commercial database management products popularly in use today?

What constitutes a simple security mechanism, in our context, is that the database security mechanism needs to be a practical and workable solution, general in nature, and that suits variety of commercial DBMS products. The database security mechanism should be based on cryptographic technology which is independent, possibly developed locally, and is based on strong theoretic foundation.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This literature review will focus on knowledge representation, research on existing product in the market, comparison between the existing product and the process modeling standards that was used during the development of OraCrypt™ product.

2.2 Cryptographic Application In Database Security

Database management systems are a part of computer systems, as such protection of information in databases are very much dependent on the operating systems and the DBMSs. Traditionally, physical security and operating system security have been used to provide security for the databases. Physical security has the disadvantages of becoming very expensive, prevents remote access, and cannot address the requirement of a wide variety of legitimate users with disparate types of access privileges to information stored in the database. The operating system supervises user access to raw data in the operating memory. The DBMSs, though is the primary line of defense in controlling user access to databases have the weakness of not able to guarantee that software errors. This database weakness is mainly due to the not advance

enough of the current state of software certification, database vendor policies or governmental politics.

Using cryptographic controls in database security, results in data sets being stored in databases in the form of suitable cryptograms. Cryptograms are non-readable to anyone without the appropriate cryptographic key to apply. Illegal leaking of such data will not compromise protection of the database. As cryptographic operations are usually carried out under the supervision of the operating system, transmission of information between the computer and the auxiliary memory is also protected [Seberry and Pieprzyk (1989)]. Gudes *et. al* (1976) and Seberry and Pieprzyk (1989) introduced cryptographic protection into a database system. They presented some basic cryptographic transformation techniques that preserve the database structure.

These are ten (10) characteristics of a database encryption system or ten (10) constraints imposed by typical database organizations and their application environments.

i. The encryption system should either be theoretically secure or require an extremely high work factor to break.
ii. Encryption and decryption should be fast enough not to unacceptably degrade system performance.
iii. The encrypted data should not have significantly greater volume than the unencrypted data.
iv. The encipherment must be record oriented, thus most stream ciphers are effectively eliminated from consideration.
v. The encryption scheme should support the logical subschema approach to databases (each field should be accessible under a different key(s)).
vi. An encrypted record must not consist of a series of individually encrypted fields. Rather, an encrypted record should be a single encrypted value, which is a function of all fields.

vii. The encryption scheme should be as flexible as possible with regard to the combinations of read and write privileges which can be granted and the conditions under which reads and writes must take place.

viii. The DBMS should be able to detect and reject records, which are created or updated under a false encryption key without being able to determine what the data are.

ix. The DBMS should not be forced by the encryption system to keep duplicate copies of data items in order to allow subschema to be represented.

x. Since records are likely to be built over a period, it must be possible to recover information from partially completed records in the same way it is recovered from full records.

Database encryption and authentication at the field level is not usually recommended for security reasons. The reasons are using encryption to hide individual data elements is vulnerable to cipher text searching; and using cryptographic checksums to authenticate individual data elements is vulnerable to plaintext or cipher text substitution. However, field based protection is advantageous as it allows projections to be performed and individual data elements decrypted or authenticated. To take advantage of this and also solving the above two (2) weaknesses, Denning (1983) proposed techniques for secure encryption and authentication at the field level. The principle idea behind her techniques is to generate a different key to encrypt (or authenticate) each data element in the database.

2.3 Cryptographic Capabilities Of Commercial DBMS

Research in the application of cryptography in databases has made some progress compared to its implementation in commercial database management systems. The current commercial database management systems have some form of cryptographic controls. The next paragraphs provide a brief preview of the cryptographic features pertaining to database encryption in commercial databases.

DB2 Universal Database is IBM's proprietary DBMS. IBM® DB2 Version 7.2 has the capability for encryption and decryption of string data through user-defined functions. With the built-in encryption and decryption functions provided, users can encrypt data to add an additional layer of security. The ENCRYPT function encrypts data using a password-based encryption method. The encryption function also allows for a password hint to be stored and another function is provided to get the hint without using the password. The DECRYPT functions decrypt data using a password-based decryption method.

Microsoft® SQL Server™2000 does not have any direct capability to encrypt data in its database. However, it uses Kerberos to support mutual authentication between the client and the server, as well as the ability to pass the security credentials of a client between computers. So that work on a remote server can proceed using the credentials of the impersonated client. Microsoft® SQL Server™ (Microsoft, 2001) has the capability to encrypt:

- Login and application role passwords stored in SQL Server.
- Any data sent between the client and the server as network packets.
- Stored procedure definitions.
- User-defined function definitions.
- View definitions.
- Trigger definitions.
- Default definitions.

- Rule definitions.

Oracle™ Corporation is undoubtedly the largest database software vendor and the largest provider of software for e-business today [Oracle Press (2000)]. Being the leader in secure, highly available database software Oracle first introduced Trusted Oracle7 MLS RDBMS in 1992. The product supports strict multilevel security policy on trusted operating system. In 1998, the Secure Access Tier 3 product was merged to produce Oracle Label Security in 1999, which is built on the Oracle8i Virtual Private Database (VPD). Oracle8i has the integration capability with external services like Distributed Computing Environment (DCE). Early 2001, with the release of Oracle9i, the database software now provides a wide range of solution ranging from data encryption across the network and within the database, consolidated user access across multiple applications, user authentication, server-defined access control policies, and extended auditing capabilities [Oracle (2001)].

Oracle has supported encryption of network data through Oracle Advanced Security, since Oracle7. Starting with Oracle8i release 2, Oracle has allowed selective data to be protected via encryption within the database as well. To address the need for selective data encryption, Oracle9i provides the *DBMS_OBFUSCATION_TOOLKIT PL/SQL* package to encrypt and decrypt stored data. The package supports bulk data encryption using DES algorithm, and includes procedures to encrypt and decrypt using DES. It also supports triple DES (3DES) encryption, in both two (2) and three (3) key modes. Oracle9i supports two (2) methods of encrypting data prior to storage in the database, which are a PL/SQL interface and a Java Cryptographic Extension (JCE) provider. In this proprietary package, users are not able to plug in a different encryption algorithm, nor make multiple passes of the function.

2.4 Smartcard Usage

The main purpose of using smartcard in the OraCrypt™ product is to store keys of the database owners. The keys cannot be stored in the database itself because of security reason. Some techniques were proposed to solve this problem either by storing the keys in diskettes, CD-ROMs or a file to store all the owner's keys. All the proposed ideas were not suitable due to the criteria of media size, storage space, processor needed and security features. Then the smartcard idea comes out. After doing some research, the project found that storing the keys in smartcard is more appropriate and secrecy than storing the keys in the database.

A smartcard is a portable, tamper-resistant computer with a programmable data store. Smartcards are different with magnetic stripe cards. The advantages of using smartcard are security, economical, convenience, customization and multi-functional. A smartcard has two (2) types of fundamental software; there are host software (runs on a computer connected to a smartcard and referred to as reader-side software) and card software (runs on the smartcard itself and referred to as card-side software). Host software is the most smartcard software used and will be written against existing smartcards, either commodity off-the-shelf smartcard available from manufacturers or created by major smartcard issuers such as bank associations, retailers, and national governments.

One of the first tasks of a host software programmer is to choose the smartcard that will be included in the system. There are a lot of smartcards developers and manufacturers in the market nowadays such as Atmel, EDS, Gemplus, IBM, NTRU Cryptosystems Inc, Philips Electronics, SC Solutions, SchlumbergerSema**,** SCM Microsystems, Turtle Mountain Communications, and Allied Solution. The current principles of smartcard standards include PKCS#11, which are Cryptographic Token Interface Standard, PC/SC, OpenCard, Java Card, Common Data Security Architecture, Microsoft Cryptographic API, Embossed and Difference between OpenCard and PC/SC.

Other industry standard groups are GSM, CEPS, EMV, Global Platform, and Open Platform.

2.5 Research On Existing Product

This section covers issues on several existing product that related to encryption and decryption process, and database security.

2.5.1 DBEncrypt

DBEncrypt is a flexible solution providing a means of encrypting rows and columns in a database. DBEncrypt provides a complete database encryption solution including a variety of strong encryption algorithms to pick from, templates to build own encryption procedures from, as well as a point-and-click user interface for installing and managing the encryption.

2.5.1.1 Turn-Key Solution

DBEncrypt transparently encrypts data on a per column basis. The user picked the columns for encryption and decryption and DBEncrypt does the rest in seconds. As an encryption solution, DBEncrypt not only saves thousands of hours of research and development but also provides a secure solution that has been reviewed by the security community. DBEncrypt allows database administrators and developers to encrypt data develop an entire key management system themselves.

2.5.1.2 Low Level Application Programming Interface (API)

DBEncrypt provides a direct interface to encryption functions that allow database administrators and developers to design and build their own encryption features. From PL/SQL, developers can access a rich set of encryption features including hashing functions, public key ciphers, block and stream ciphers, ciphers to sign and verify data, and random numbers generators. This low- level API provides the developer the flexibility to choose details such as the padding mode or the key size. As a low- level interface, DBEncrypt provides developers with the ability to utilize encryption as he sees fit. These developers are empowered with the ability to extend the current DBEncrypt features as well as design entirely independent solutions.

2.5.1.3 Features Of DBEncrypt

DBEncrypt features are:

i. Database column and row level encryption
   DBEncrypt provides database column and row-level encryption callable from SQL and PL/SQL. It was designed to be incredibly efficient for PL/SQL programmers and lucidly simple for the business user. Effective simplicity in design directly translates into the organization to saving time and money.

ii. Access to world class security resources that facilitate effective data protection
   DBEncrypt™ provides enterprise with strength protection for the database by giving user full access to a wide variety of block and stream ciphers, public key algorithms, message authentication codes, and one-way hash functions, as below:

   - Symmetric Algorithms (Block) - AES, DES, DESX, Triple DES (2 and 3 key), Blowfish, Twofish, RC2, Serpent, CAST128, CAST256 and Skipjack
   - Symmetric Algorithms (Stream) - RC4_compatible
   - Public Key Algorithms - RSA

- Hashing Algorithms - SHA1, MD5
- Message Authentication Code - HMAC_SHA1

iii. Mechanism in providing authentication, encryption and data integrity

Tools and templates are provided to create strong authentication, encryption, and data integrity of the information within the database. DBEncrypt™ provides an interface to a group of both low- level and high- level encryption functions. It is also provided an interface to generate secure random numbers, strong encryption keys, and initialization vectors.

2.5.1.4 DBEncrypt Pros And Cons

DBEncrypt security solution provides a way to meet security and privacy in the database. This is because DBEncrypt is transparent to encrypt data on a per column basis. With DBEncrypt solution, it allows database administrators and developers to encrypt data without having to invent and develop an entire key management system themselves. In DBEncrypt, tools and templates are provided to create strong authentication, encryption, and data integrity of the information within the database. It also provides an interface to generate secure random numbers, strong encryption keys, and initialization vectors.

DBEncrypt provides a direct interface to encryption functions that allow DBA and developers to design and build their own encryption features. From PL/SQL, developers can access a rich set of encryption features including hashing functions, public-key ciphers, block and stream ciphers, and random numbers generators. As a low-level interface, DBEncrypt provides developers with the ability to utilize encryption as he sees fit. DBEncrypt also provides database column and row-level encryption callable from SQL and PL/SQL.

2.5.2 Protegrity Secure.Data™

Secure.Data™ is a granular privacy solution for sensitive information and records in databases. It released by Protegrity, an information security company, specializing in the encryption of databases. The product includes Secure.Data Manager and Secure.Data Server modules. Secure.Data Manager is the security official that defines user, controls data-access privileges, and specifies data to be encrypted and other security parameters. Secure.Data Server is the active component, performs real-time security processing of encryption and decryption. Secure.Data™ provides encryption protection for information at a data item level. It allows organizations to manage and control all access to information. Secure.Data provides dynamic and real-time enterprise-wide database protection across major platforms and operating systems.

2.5.2.1 Unique Approach

The Protegrity Secure.Data program suite takes a granular approach to information security. Instead of building walls around servers or hard drives, Secure.Data builds a protective layer of encryption around individual data items or objects. This concept protects the data wherever it is stored or processed instead of protecting the server that stores or protects the data (*Figure 2.1*).

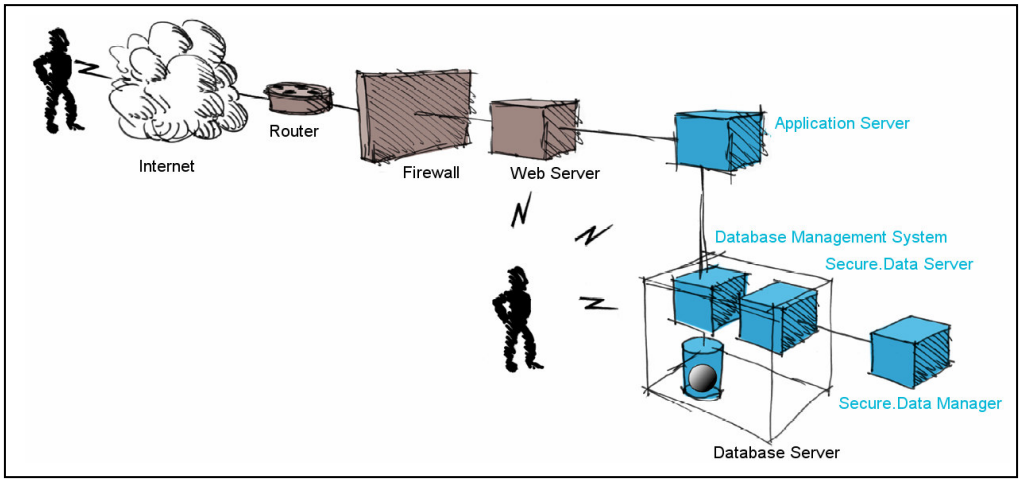


Figure 2.1 Secure.Data Unique Approaches

The Protegrity solution adds a critical and final checkpoint for users accessing sensitive information. Secure.Data remains in place protecting the data at the closest and most effective point. This prevents outside attacks as well as infiltration from within the server itself.

2.5.2.2 Data Item Protection

Secure.Data allows an organization manager to define which data stored in databases that is sensitive and which individuals or groups that should have the authority to access the information. This selection is done on a data item level, thereby focusing protection only on the sensitive data. The sensitive data is then encrypted and stored in the database. The selectivity of Protegrity's data item protection technique not only prevents intruders from gaining access to sensitive data, but also minimizes the delays or burdens on the system that may occur from other bulk encryption methods. The Protegrity approach is transparent to applications and does not affect non-sensitive data.

2.5.2.3 Role Based Access

The role based access model is the cornerstone of the Secure.Data work-enabling design. This method allows managers to decide the levels of user access that are appropriate for various types of information. In Secure.Data, roles are created and determined as well as defined the unique data access privileges for each user. These roles are used to authenticate and authorize any application requests made from user queries. Roles can be assigned to users on a specific, individualized basis or general role types can be created to embody the functions commonly performed by other individuals who need secure access to sensitive data. All roles enterprise-wide are managed from the Secure.Data Manager and enforced through the Secure.Data Server. Secure.Data can import information from the organizations' preexisting access control facilitates to minimize data entry and management.

2.5.2.4 Encryption

Secure.Data utilizes encryption to transform sensitive data into a form that is unreadable to anyone who does not have proper authorization. To achieve maximum protection of data and maintain the best system performance, Protegrity offers its crypto-conductor process. This system allows implementation of encryption down to the column level within databases. While other security technologies encrypt whole files, tables or databases, Secure.Data lets users specify and secure only the sensitive data items that need protection. The crypto-conductor process supports encryption for up to 168 DES, strong (CBC block cipher) encryption with IV. Data encryption is the only way to absolutely protect sensitive data, eliminate potential data sharing issues, ensure that privacy is maintained, minimize the potential for identity theft, and ultimately meet data privacy compliance requirements. There are two (2) basic approaches to encryption of stored data:

i. The entire database can be encrypted. There are two (2) problems with this approach. First, it allows anyone with access to the database to decrypt all data stored therein. Second, it exacts a substantial performance penalty because all requests, whether for sensitive data or not, it requires data to be decrypted and re-encrypted.

ii. Individual database applications can be modified to encrypt sensitive data. This requires the use of crypto toolkits to reengineer in-house applications, a costly and inflexible application-by-application proposition that does not offer a scalable and uniform approach to data privacy. In addition, this approach does not provide essential key management functions and may actually increase the number of people requiring access to databases for development and maintenance purposes.

2.5.2.5 Protegrity Secure.Data™ Pros And Cons

Secure.Data provides a way to meet security and privacy requirements without having to alter the code in the applications. This is called application transparency. With the Protegrity solution, users can no longer bypass security policies embedded in applications because security policies are associated directly with data. This solution unifies access controls and protection of information into a central protected database repository, and at the same time closes the security loopholes that inevitably occur when adding, deleting, removing, and changing user access rights to sensitive information stored in a database.

The Protegrity solution allows the company to predetermine who is authorized to have access to sensitive information within corporate databases and to apply the strongest mechanisms of encryption based security to this select information. Above this core functionality, the Protegrity solution then provides automatic encryption key management. The following are the benefits of the product that ensure protection of corporate information assets, protect its customers, and consumers.

- Protection

  Secure.Data Manager operates in its own self-secure encrypted environment, protected by strong authentication that ensures the confidentiality and integrity of all Secure.Data security parameters. Secure.Data Manager runs on a workstation separate from both application and database servers.

- Simple user interface

  Secure.Data Manager's graphical user interface incorporates the latest Windows techniques, such as *drag and drop* and *browse and pick*. The easy-to-use GUI greatly simplifies implementation and management procedures, such as assigning access privileges and choosing data protection methods.

- Separation of duties

  It allows organizations to separate database security control from the duties of the database and system administrators. The database or system administrator can still perform his duties, but an appointed Security Manager maintains responsibility for the security of the sensitive information through the Secure.Data Manager module, which is operated separately from the database.

- Advanced prevention of internal attacks

  Secure.Data provides a sophisticated system of self-protection that protects the encryption mechanism in a way that it cannot be manipulated by a DBA or System Administrator. Through this self-protection mechanism, no user, even with root/administrator rights, can undetected access the encrypted data.

- High granularity

  By using role-based access control and multilevel security, Secure.Data Manager allows operators to define user roles, workgroups, access rights and encryption requirements down to the data item level.

- Single point of control

  Secure.Data does not require security operators to update security parameters for various servers individually. Secure.Data Manager's centralized administration allows operators to maintain the integrity of all Secure.Data policies and configurations from a single location.

- Auditing and reporting

  Encrypted logs are produced to track both the Secure.Data Manager administrator's activities and Secure.Data Server activities around protected data, such as records of changes to the security policy and unauthorized access attempts. Real-time auditing ensures non-repudiation of transactions on sensitive information and is independent of the DBMS audit mechanism. Secure.Data Manager also includes a report generator with various templates that can be tailored by the security operator and exported for use. All logs are of common format regardless of the Transaction

Protocols and Database Managers in use. Secure.Data encrypts all logs and audits to ensure the fullest protection and security.

2.6 Existing Product Comparative Study

The comparative study was done on the security features and characteristics of four (4) existing database security systems, which uses cryptography to secure data in databases. The existing systems are DBEncrypt™, Secure.Data™, Microsoft SQL Server 2000, and Oracle Database Security.

2.6.1 DBEncrypt vs. Secure.Data Features

Table 2.1 shows the comparative studies on characteristics and features between DBEncrypt and Secure.Data.

Table 2.1 DBEncrypt vs. Secure.Data

Characteristics	Features	DBEncrypt	Secure.Data
Installation and un-installation	Easiness	Fast and easy for non-expert user.	Fast but tedious for non-expert user.
	Modules	Install client-side module, then install server side module.	Install Secure.Data Manager, then install Secure.Data Server.
	Setup and configuration	Provide sample database for tutorial.	Provide sample database for tutorial.
	Platforms	Server-side: ▪ Database - Oracle 8i, MySQL ▪ OS - Sun Solaris, HP-UX, Linux, MsWindows NT/2000 Console: ▪ OS - MsWindows NT 4.0 SP3 / 2000	Server-side: ▪ Database - Oracle 8i, MySQL ▪ OS - Sun Solaris, HP-UX, Linux, MsWindows NT/2000 Console: ▪ OS - MsWindows NT 4.0 SP3 / 2000

Characteristics	Features	DBEncrypt	Secure.Data
		▪ HDD - 10MB ▪ RAM - >32MB	▪ HDD - 10MB ▪ RAM - >32MB ▪ VPN-1/FireWall-1 v4.1 SP1 or higher (gateway)
User Interface		▪ Simple interface for selecting column to be encrypted, and user-key generation. ▪ Provide space to execute PL/SQL commands. ▪ User-friendly GUI interface with *point-and-click*.	▪ Provide Security Policy interface for script generation. ▪ Generate script for user to use for encrypt and decrypt. ▪ GUI Interface with Multiple Document Interface (MDI) appearance. ▪ GUI incorporates the latest Windows techniques, such as *drag and drop* and *browse and pick*.
Capability	Encryption / Decryption	Encrypt and decrypt process on column only.	Encrypt and decrypt process on column, row and data element.
	Data type supported	▪ VARCHAR2 ▪ NUMBER (without precision or scale)	▪ VARCHAR ▪ Real, Float, Integer
	Algorithms	Provide 14-encryption algorithm - AES, DES, DESX (120-bits), Triple DES, Blowfish, Twofish, RC2 compatible, RC4 compatible, Serpent, CAST128, CAST256, SKIPJACK, RSA, SHA1.	168 DES, strong (CBC block cipher) encryption with IV (Initialization Vector).
	Help	▪ Good coverage on the package capability. ▪ Provide examples. ▪ Wizard supported.	▪ Good coverage on the package capability.
	Key management	Secret keys are kept in 3 ways: i. Encrypted with password ii. Operating system iii. Table within database	Using dynamic 2 factor RSA SecureID authentication: i. User knows ii. User possess
	Key generation	Private and Public key model.	Role based model.
	Clustering	Have to install in each database server.	Have to send the encryption and decryption process to database server .
	Impact on	▪ Minimum degradation on	▪ No need to encrypt all data.

Characteristics	Features	DBEncrypt	Secure.Data
	database	encrypted data. ▪ No need to encrypt all data.	
General	Availability	Trial version license provided for 2 weeks.	General.
	Technical documentations	Provided in full.	Provided in full.

2.6.2 Microsoft SQL Server 2000 vs. Secure.Data Features

Table 2.2 shows the comparative studies on features between Microsoft SQL Server 2000 and Secure.Data.

Table 2.2 Microsoft SQL Server 2000 vs. Secure.Data

Microsoft SQL Server 2000	Secure.Data
User authentication	**Maintain the existing database authentication**
User authenticate once with single username/password.	Secure.Data sit on top of the DBMS and it fully utilized the default authentication of the DBMS.
Predefined server and database roles for application users	**Enhanced role, row and time based access control**
Set the access control by predefined roles in the database, allow DBA to assign the relevant user to that role.	Access is enhanced by the roles and workgroups; additional security levels can also be defined for workgroups to further enhance row-level access control. Further enhanced by time based access controls, which allow user to login to the database in specific time range in a day or a period.
Data protection (in transit)	**Data protection (encryption for data at rest)**
Secure Socket Layer (SSL) encryption of the data and other network traffic as it travels between the client and server system.	Information is encrypted during transit and at rest stage. Information is unreadable to anyone who is not authorized, it will only be decrypted back if the user is authorized.
Server-based encrypted data, passwords and stored procedures	**Advances and automatic encryption key management**
Further enhanced to use the Windows Crypto API.	Unified protection across major database platforms. Allow implementation of strong cryptographic algorithms, 3DES based on CBC and IV.

2.6.3 Oracle vs. Secure.Data Features

Table 2.3 shows the comparative studies on features between Oracle Database Security and Secure.Data.

Table 2.3 Oracle vs. Secure.Data

Oracle 8i/9i	Secure.Data
User authentication to identify users throughout all tiers of the network	**Maintain the existing database authentication**
User authenticate once with single username/password. Centralization of user access information in oracle internet directory. PKI integration functions for easier deployment and management.	Secure.Data sit on top of the DBMS and it fully utilized the default authentication of the Oracle DBMS. Secure.Data can maintain all DB username and password.
Role and row based access control	**Enhanced role, row and time based access control**
Set the access control by predefined roles in the organization, and assigned the relevant user to that role. Oracle Virtual Private Database (VPD) offers fine-grained access control or row level securities can only access rows of data that belong to him/her by labeling data at the row level.	Access is enhanced by the roles and workgroups; additional security levels can also be defined for workgroups to further enhance row-level access control. Further enhanced by time based access controls, which allow user to login to the database in specific time range in a day or a period.
Data protection (in transit)	**Data protection (encryption for data at rest)**
Oracle advanced security provide SSL RC4 and 3DES encryption which ensures data confidentially and integrity as data travels across the network.	Information is encrypted during transit and at rest stage. Information is unreadable to anyone who is not authorized, it will only be decrypted back if the user is authorized.
Stored data protection	**Advances and automatic encryption key management**
Oracle Pl/SQL encryption toolkit allows for development of native database encryption capabilities.	Unified protection across major database platforms. Allow implementation of strong cryptographic algorithms, 3DES based on CBC and IV
Auditing for accountability	**Selective and secured auditing**
Audit track users database activity.	Monitoring the activity around protected sensitive information. All audit information is encrypted and stored.

2.7 Software

Humphrey (1990) formally defines software as a program and all the associated information and materials needed to support its installation, operation, repair, and enhancement. This definition has some implications on software projects. Firstly, it widens the scope of software to include documentations (all of them throughout the lifecycle), support, training, and maintenance. Secondly, it extends the notion of software lifecycle: the lifecycle does not end when the product is shipped, but seems rather un-ending instead. These implications stress the potential largeness of software projects, which in turn implies the need for some kind of processes.

2.8 Process

Process is defined as a series of actions and operations by the Webster dictionary. Process thus implies a structured, ordered set of activities. We can also think of process as workflow in the context of product development.

2.9 Software Process

The software development process is the central process of any software development effort. Humphrey (1990) defined software process as a set of activities, methods, and practices that are used in the production and evolution of software. In short, a software process defines who is doing what, when and how to reach a certain goal. Thus, software process could be thought as an ordered set of activities, methods, and practices that are used in anything that is related to software development. In other words, all activities that are related to software projects could be put within some software processes, or software processes are applicable to all activities in a software project. The purpose of a software development process is to produce high quality and timely results without imposing a large overhead on the project. The process is depicted as a model that determines the order of stages involved in development and evolution

establishes transition criteria for progressing between stages by specifying the entrance and exit criteria.

2.10 Rational Unified Process (RUP)

The Rational Unified Process (RUP) is based on the integrated work of three (3) primary methodologists, Ivar Jacobson, Grady Booch and James Rumbaugh. These methodologists, aided by a large and extended methodologist community, were assembled by Rational Corporation to form a unified, cohesive and comprehensive methodology framework for the development of software systems. There work, occurring over several years and based on existing, previously tested methodologies, have lead to significant standards in the development community, including the general acceptance of use cases and the Unified Modeling Language™ (UML).

2.10.1 RUP Definition

Ivar Jacobson, Grady Booch and James Rumbaugh (1999), and Kruchten (1999) state that RUP software process is a generic business process for object oriented software engineering. It describes a family of related software engineering processes sharing a common structure and common process architecture. It provides a disciplined approach to assigning tasks and responsibilities within a development organization. The exactly definition of RUP can be derived from different perspectives, as below:

i. Software engineering process

RUP is a software engineering process that provides a disciplined approach to assigning tasks and responsibilities within a development organization. Its goal is to ensure the production of high-quality software that meets the needs of its end-users, within a predictable schedule and budget [Ivar Jacobson, Grady Booch and James Rumbaugh (1999), Kruchten (1999)].

ii. Process product

RUP is a process product, developed and maintained by Rational® software. The development team for the RUP is working closely with customers, partners, Rational's product groups as well as Rational's consultant organization, to ensure that the process is continuously updated and improved upon to reflect recent experiences and evolving and proven best practices.

iii. Unified Modeling Language (UML)

RUP is a guide for how to effectively use the UML. The UML is an industry-standard language that allows us to clearly communicate requirements, architectures and designs. The UML was originally created by Rational® software, and is now maintained by the standards organization Object Management Group (OMG) [Booch *et. al.* (1988)].

iv. Tools

The RUP is supported by tools, which automate large parts of the process. They are used to create and maintain the various artifacts of the software engineering process, which are visual modeling, programming, and testing. They are invaluable in supporting all the bookkeeping associated with the change management as well as the configuration management that accompanies each of iteration.

v. Best practices

RUP captures many of the best practices in modern software development in a form that is suitable for a wide range of projects and organizations. Deploying these best practices by using the RUP will offers development teams a number of key advantages.

2.10.2 Phases In RUP

RUP consists of cycles that may repeat over the long-term life of a system. A cycle consists of four (4) phases: Inception, Elaboration, Construction, and Transition [Kruchten (1995)]. The software lifecycle is broken into cycles, each cycle working on a new generation of the product. Each phase is concluded with a well-defined milestone (*Figure 2.3*), which means a point in time at which certain critical decisions must be made and therefore key goals must have been achieved [Barry W. Boehm (1996)].

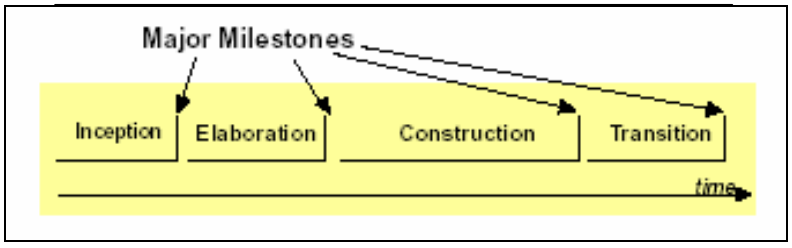


Figure 2.3 The Phases and Major Milestones in the RUP

2.10.2.1 Inception Phase

During the inception phase (*Appendix B-1*) the core idea is developed into a product vision. In this phase, we review and confirm our understanding of the core business drivers. We want to understand the business case for why the project should be attempted. The inception phase establishes the product feasibility and delimits the project scope. To accomplish this you must identify all external entities with which the system will interact and define the nature of this interaction at a high-level. This involves identifying all use cases and describing a few significant ones. The business case includes success criteria, risk assessment, and estimate of their sources needed, and a phase plan showing dates of major milestones [Kruchten (1995), Walker Royce (1998)].

At the end of the inception phase is the Lifecycle Objectives Milestone. The evaluation criteria for the inception phase are:

- Stakeholder concurrence on scope definition and cost/schedule estimates.
- Requirements understanding as evidenced by the fidelity of the primary use cases.
- Credibility of the cost/schedule estimates, priorities, risks, and development process.
- Depth and breadth of any architectural prototype that was developed.
- Actual expenditures versus planned expenditures.

2.10.2.2 Elaboration Phase

The purpose of this phase (*Appendix B-2*) is to analyze the problem domain, establish a sound architectural foundation, develop the project plan, and eliminate the highest risk elements of the project. During the elaboration phase, the majority of the use cases are specified in detail and the system architecture is designed. Architectural decisions have to be made with an understanding of the whole system, which are scope, major functionality and nonfunctional requirements such as performance requirements. At the end of this phase, the hard engineering is considered complete and the decision on whether or not to commit to the construction and transition phases. For most projects, this also corresponds to the transition from a mobile, light and nimble, low-risk operation to a high-cost, high-risk operation with substantial inertia. While the process must always accommodate changes, the elaboration phase activities ensure that the architecture, requirements and plans are stable enough, and the risks are sufficiently mitigated, so you can predictably determine the cost and schedule for the completion of the development.

In the elaboration phase, an executable architecture prototype is built in one or more iterations, depending on the scope, size, risk, and novelty of the project. This effort should at least address the critical use cases identified in the inception phase, which typically expose the major technical risks of the project. While an evolutionary prototype of a production-quality component is always the goal, this does not exclude the development of one or more exploratory, throwaway prototypes to mitigate specific risks such as design/requirements trade-offs, component feasibility study, or demonstrations to investors, customers, and end-users.

At the end of the elaboration phase is the Lifecycle Architecture Milestone. At this point, you examine the detailed system objectives and scope, the choice of architecture, and the resolution of the major risks. The main evaluation criteria for the elaboration phase involve the answers to these questions:

- Is the vision of the product stable?
- Is the architecture stable?
- Does the executable demonstration show that the major risk elements have been addressed and credibly resolved?
- Is the plan for the construction phase sufficiently detailed and accurate? Is it backed up with a credible basis of estimates?
- Do all stakeholders agree that the current vision can be achieved if the current plan is executed to develop the complete system, in the context of the current architecture?
- Is the actual resource expenditure versus planned expenditure acceptable?

2.10.2.3 Construction Phase

During the construction phase (*Appendix B-3*), all remaining components and application features are developed and integrated into the product, and all features are thoroughly tested. The construction phase is, in one sense, a manufacturing process where emphasis is placed on managing resources and controlling operations to optimize costs, schedules, and quality. In this sense, the management mindset undergoes a transition from the development of intellectual property during inception and elaboration, to the development of deployable products during construction and transition. Many projects are large enough that parallel construction increments can be spawned. These parallel activities can significantly accelerate the availability of deployable releases; they can also increase the complexity of resource management and workflow synchronization. A robust architecture and an understandable plan are highly correlated. In other words, one of the critical qualities of the architecture is its ease of construction.

At the end of the construction phase, you decide if the software, the sites, and the users are ready to go operational, without exposing the project to high risks. This release is often called a *beta* release. The evaluation criteria for the construction phase involve answering these questions:

- Is this product release stable and mature enough to be deployed in the user community?
- Are all stakeholders ready for the transition into the user community?
- Are the actual resource expenditures versus planned expenditures still acceptable?

2.10.2.4 Transition Phase

The purpose of the transition phase (*Appendix B-4*) is to ensure that the requirements have been met to the satisfaction of the stakeholders and transit the software product to the user community. Once the product has been given to the end user, issues usually arise that require developing new releases, correcting some problems, or finish the features that were postponed. The transition phase is entered when a baseline is mature enough to be deployed in the end user domain. This typically requires that some usable subset of the system has been completed to an acceptable level of quality and that user documentation is available so that the transition to the user will provide positive results for all parties.

The transition phase focuses on the activities required to place the software into the hands of the users. Typically, this phase includes several iterations, including beta releases, general availability releases, as well as bug fix and enhancement releases. Considerable effort is expended in developing user-oriented documentation, training users, supporting users in their initial product use, and reacting to user feedback. At this point in the lifecycle, user feedback should be confined primarily to product tuning, configuring, installation, and usability issues. The primary objectives of the transition phase include:

- Achieving user self-supportability.
- Achieving stakeholder concurrence that deployment baselines are complete and consistent with the evaluation criteria of the vision.
- Achieving final product baseline as rapidly and cost effectively as practical.

At the end of the transition phase is the Product Release Milestone. At this point, we decide if the objectives were met, and if we should start another development cycle. In some cases, this milestone may coincide with the end of the inception phase for the next cycle. The primary evaluation criteria for the transition phase involve the answers to these questions:

- Is the user satisfied?
- Are the actual resources expenditures versus planned expenditures still acceptable?

CHAPTER 3

PROJECT METHODOLOGY

3.1 Introduction

Almost all operations need some kind of processes. Software development is no exception. Software development is a complex task. The researcher handled complexity by working with different models of the system to develop and each focusing on certain aspects of it. The development process is based on architecture, method, process, and tools. The organizations that developed security software and implemented well software engineering process have improved the quality of the software product and process itself. Therefore, software process is instrumental in the success of software management. To aim this goal, the researcher implements the software engineering process to support the development of OraCrypt™ product.

3.2 Software Development Methodology

Software engineering is known as an engineering discipline frequently characterized the approach as a practical, orderly and measured development of software. The goals of software engineering focus on the development of software that supports the needs of the user and helps the maintainer to adapt the software so that it will continue to support the evolving needs of the user.

In this chapter, the software development methodology focuses on software process, software methods, software techniques and tools used for this project. It comprises of well-defined steps and techniques to built a right system with the effective software development strategy. The choice of the software development methodology is one of the major contributors to the success or failure of software development projects because it defines the way a system is built and organized.

3.3 RUP As A Software Development Process

In RUP based project, the overall architecture of the software development process goes through four (4) phases, which are Inception, Elaboration, Construction, and Transition phase. In this section, the researcher details the objectives, purposes and activities involved for each phases.

3.3.1 Inception Phase

The goal of this phase is to help the project team decided what the objectives of the project would be. It tells everyone what the system is, and may also tell who will use it, why it will be used, what features must be present, and what constraints exist. The researcher established the scope of OraCrypt™ product, what is included, and what is not, and its boundary conditions. The researcher wanted to understand the business case for why the project should be attempted. To accomplish this, all external entities are identified with which the product will interact and defined the nature of this

interaction at a high-level. This involved the identification of actors and use cases for this system. And then, the researcher drafted the most essential use case that will drive the major design trade-off. Businesses and requirements risks were also addressed before the project can proceed. During the inception phase, the researcher has identified an iteration that called *Preliminary Iteration*.

3.3.1.1 Preliminary Iteration

The preliminary iteration involved project management, software requirement analysis, and environment workflow. In this section, the researcher details the activities for each workflow involved in this iteration during the inception phase.

i. Environment Workflow

This workflow describes on the engineering part of the OraCrypt™ product development. The environment discipline focuses on the activities necessary to configure the process for the project. It describes the activities required to develop the guidelines in support of the project. The purpose of the environment activities is to provide the software development organization with the software development environment, with both processes and tools that will support the development team. Tools include those for modeling, testing, requirements, management, coding, planning, and tracking. Practically, all the works and activities in the environment workflow are done during the inception phase. Many of the tasks associated with the process can benefit from the use of tools. The following activities were carried out during the environment workflow (*Appendix D-1*).

- Prepared the Development Case

  The researcher created a first version of the Development Case of the project. It will be developed in increments, covering more and more of the disciplines in each of iteration. The Development Case includes phases and milestones;

artifacts to produce, activities to perform, the way to work in each discipline, and iteration plan descriptions for this project.

- Prepared the software specification

  The minimum software specifications needed to carry out development of OraCryptTM product are stated as follow.

  a. Project Management Workflow

Table 3.1 Lists of Tools in Project Management Workflow

Software	Purpose
Microsoft Project 2000	Project tracking and planning.
Microsoft Word 2000	Documentation tools.

  b. Requirements Workflow

Table 3.2 Lists of Tools in Requirements Workflow

Software	Purpose
Microsoft Word 2000	Documentation tools.
Rational Rose Tool Evaluation Version	Software requirements tools.
Rational Requisite Pro	Requirements management tools.

  c. Analysis and Design Workflow

Table 3.3 Lists of Tools in Analysis and Design Workflow

Software	Purpose
Microsoft Word 2000	Documentation tools.
Rational Rose Tool Evaluation Version	Analysis and design tools.

  d. Implementation Workflow

Table 3.4 Lists of Tools in Implementation Workflow

Software	Purpose
Microsoft Visual Basic 6.0	Software development tools.

Oracle 8i Enterprise Edition Release 8.1.7.0.	Database management system.
Microsoft Word 2000	Documentation tools.

- Prepare the hardware specification

  The minimum hardware specification that is required for the development OraCrypt™ product is stated as follow.

Table 3.3 Hardware Specifications

No	Requirements
1	Operating system – Microsoft Windows 2000 Server
2	Processor – Intel Pentium IV
3	Hard disc – 30 GB
4	RAM – 512 MB
5	Monitor – 14 inch
6	Mouse – serial port

ii. Project Management Workflow

Project management is conducted to provide a framework for managing and understand the structure and the dynamics of the organization in which an OraCrypt™ product is to be deployed. In this workflow, the researcher identified the process and improvement potentials for the organization to ensure everyone have a common understanding of the target organization. The system requirements are derived to support the target organization. Therefore, the researcher provided the guidelines for planning, staffing, managing risk, executing, and monitoring the project. The following activities were carried out during the project management workflow (*Appendix D-2*).

- Established the development team

  It is important to have at least an experience team member to ensure the objectives of software development are catered.

- Formulated the scope of project

The purpose of this activity is to reappraise the project's intended capabilities and characteristics. Therefore, the researcher captured the context and most important requirements in enough detail to derive acceptance criteria for the product to support target organization. The development needs to know what the scope of the project is and how it will satisfy the user of the organization.

- Estimated the potential risks

  Many decisions in an iterative lifecycle are driven by risks. In order to achieve this, the researcher need to get a good grip on the risks the project is faced with, and have clear strategies on how to mitigate or deal with them. So, the researcher estimated all the possible risks, planned for the risk mitigation to reduce the probability or impact, planned for contingencies and monitored the risks.

- Developed the Software Development Plan

  The researcher developed the Software Development Plan that captures the overall envelope of the project, for one cycle and the following cycles, and all of the information necessary to control the project. The researcher planned the project schedule and resource needs, and to track progress against the schedule. In SDP, the researcher also described the approach to the development of the software, and the top-level plan for directing the development effort. To get a good plan and estimation, the researcher getting the technical estimation from the developers who will implement an OraCrypt™ product features.

- Developed the Iteration Plan

The researcher developed the iteration plan that contains set of activities and tasks, with assigned resources, containing task dependencies for the iteration. In this project, there are two (2) iteration plans active at any given point, which are current iteration plan and the next iteration plan. The current iteration plan is used to track the project progress, while the next iteration plan is built towards the second half of the current iteration, and it is ready at the end of the current iteration.

- Closed-out the inception phase

  The researcher brings the phase to closure by ensuring that all major issues from the previous iteration are resolved, and the state of all artifacts is known.

iii. Software Requirements Workflow

Requirement is perhaps the most important process of the software process model. Without the correct requirements, subsequent design and implementation of the application will be practically meaningless. Hence, software requirements analysis is conducted to find out as much requirements related to the OraCrypt™ product. The purpose of requirements analysis is to establish a common understanding between the user and the software development team concerning all requirements for the defined software development project. All the functional and non-functional requirements were gathered using the following techniques:

- Internal discussion

  Discussions were conducted among team members and the supervisor to obtain a deeper understanding of the current system and the additional features required of OraCrypt™ product.

- Study of previous system

  Several documents of previous system were studied and referred in order to understand some features of OraCrypt™ product.

Finally, all requirements are managed and organized in efficient ways to build a common understanding of the project needs and commitments among the user. The following activities were carried out during the software requirements analysis workflow (*Appendix D-3*).

- Analyzed the problem

  Analysis of the problem involved identified the user, defined the system boundary, and identified the constraints imposed on the system to ensure that everyone agreed on what the problem is that needs to be solved. In order to fully understand the problem that need to be addressed, the researcher identified the high-level user view of the system to be built that will be represented in the use case model. It is also important to agree on common terminology, which will be used throughout the project by defined the project terms in a glossary.

- Outlined and clarified the user needs

  The researcher collected and elicited information from the user of the project in order to understand what their needs really are. The collected user request is used as primary input to defining the high-level features of OraCrypt™ product. This activity was done using various techniques such as storyboard and brainstorming.

- Defined the boundaries of the system

  The researcher defined the system boundaries that describe an envelope in which the solution system is contained. All the interactions with the system occur via interfaces between the system and the external world were defined. Therefore, we used Use Case Diagram as a concise summary of information contained actors and use cases to show how the actor interacts with the system and what the system does. The use case has a set of sequence actions and performs observable results to a particular actor, who interacts with the system. A use case diagram for OraCrypt™ product is shown in *Appendix C*. In the earlier phase, there were three (3) actors and six (6) use cases defined in the

OraCrypt™ product. The following are the descriptions of each actor and use case involved.

a. Actor: Security Manager (Sec Mgr)
   Security Manager is the person who responsible to ensure the system is in secured. He manages the system and database users to the proper organization structure. He also responsible to generate, manage and distribute the user key to all users. The system provides the different authentication for the Security Manager to access the OraCrypt™ product.

b. Actor: Normal User
   Normal user has the right access to use the encryption and decryption module, while user with authorized access can also manages his structure.

c. Actor: Smartcard Reader (S/Card Reader)
   Smartcard reader is the external actor for the OraCrypt™ product. It provides the communication between the smartcard and the OraCrypt™ product.

d. Use Case: Login
   This use case shows the process of verification and authentication user to the OraCrypt™ product. It enables the user to establish connection to the database.

e. Use Case: Encrypt Data
   This use case shows the encryption process that allows the user in the database to access his table for encryption purpose. The system provides three (3) levels of encryption, which are data element, column or row. In each level, user may provide condition for the selective encryption of data at those three (3) levels.

f. Use Case: Decrypt Data

This use case shows the decryption process that allows the user in the database to access the appropriate table for decryption purpose. The system provides three (3) levels of decryption, which are data element, column or row. In each level, user may provide condition for the selective decryption of data at those three (3) levels.

g. Use Case: Generate Key

This use case shows the generation of key in the OraCrypt™ product. It produced three (3) types of key, which are System Key, Master Key and Shared Key (with Control and without Control). The generated key will be managed and distributed to the user.

h. Use Case: Manage User Info

This use case is used by Security Manager to manage the user information that related to the system and the smartcard. He can imports the user information from the database to the system, add the user or user information, deletes the user or user information, and manages the smartcard services.

i. Use Case: Manage Key Structure

User used this use case to manage the hierarchical structure. The hierarchy structure can be build for organization, project, compartments, or group purpose.

- Established the scope of the system

The purpose of this activity is to make the scope of the system being developed as explicit as possible, and focused on a manageable body of requirements work for the iteration. The researcher prioritized and refined the input to the selection of requirements that are to be included in the current iteration. The researcher also determined the set of scenarios and use cases that represent some significant central functionality, and have a substantial architectural

coverage. These use cases or scenarios drive the development of the architecture and should be addressed in the current iteration.

The inception phase ends at the *Business Case Review Milestone* that marks the Go/No decision for the project budget and time as planned. At this point, the researcher examined the lifecycle objectives of the project, and decided to proceed with the project. The deliverables produce in this phase are listed as following:

- Glossary
- Development Case
- Software Development Plan
- Preliminary Plan for Inception Phase
- Iteration Plan for Elaboration Phase
- Use Case Specifications
- Supplementary Specifications

3.3.2 Elaboration Phase

The elaboration phase is where the foundation of software architecture is laid. The problem domain is analyzed, bearing in mind the entire system that is being built. The goals of this phase are to refine the support environment, defined, validated and baseline the architecture as rapidly as is practical. During this phase, a detailed plan for the construction phase will be baseline to provide a stable basis for the bulk of the design and implementation workflow. The system architecture evolves out of a consideration of the most significant requirements and an assessment of risk. The stability of the architecture was evaluated through the architectural prototypes. To ensure that the architecture, requirements and plans are stable enough, and the risks sufficiently mitigated, the researcher determined the schedule for the completion of the development. The architecture derived from the significant scenarios, which typically

expose the risks of the project, is established. In this phase, the researcher defined an iteration called *E1 Iteration* that purposely to develop architectural prototype.

3.3.2.1 E1 Iteration (Develop Architectural Prototype)

In this iteration, the workflows involved are environment, project management, software requirement analysis, and analysis and design workflow. In this section, the researcher details the activities for each workflow.

i. Environment Workflow

The purpose of this workflow is to ensure that the project environment is ready for the upcoming iteration. This includes process and tools. The following activities were carried out during the environment workflow (*Appendix D-1*).

- Prepared the environment for the iteration

  The researcher prepared and customized the tools to use within the iteration. The researcher verified that the tools have been correctly configured and installed. The researcher also prepared a set of project-specific templates and guideline to support the project development within the iteration. All the changes made to the project environment are ensuring properly communicated to the project team.

- Supported the environment during the iteration

  The researcher supported the developers in their use of tools and process during iteration. Supporting the project environment is an ongoing task to allow the project team to do their job efficiently without being slowed down due to issues with the development environment. This includes installation of required software, ensuring that the hardware is functioning properly and that potential network issues are resolved without delays.

ii. Project Management Workflow

In this workflow, we focused on refining the Software Development Plan, monitoring and controlling the OraCrypt™ project, and managing the risks. The following activities were carried out during the project management workflow (*Appendix D-2*).

- Refined the Software Development Plan

  The researcher refined the SDP, and mapping out a set of iterations using the prioritized use cases and associated risks. The project plans developed at this point are refined after each subsequent of iteration and become more accurate as iterations are completed.

- Refined the iteration plan, risks and architectural objectives

  The researcher constructed the iteration plan after the previous iteration was assessed and the project scope and risk reevaluated. The evaluation criteria for the architecture are outlined in the Software Architecture Document, taking into consideration the architectural risks that are to be mitigated. The system architecture assists with the planning. Therefore, we concerned on the consistent architecture and development guidelines to ensure it followed the project planning, and adherence to project standards.

- Monitored and controlled the project

  The researcher monitored the project in terms of active risks and objective measurements of progress and quality. The researcher regular reporting the project status, and any change requests are scheduled for the current or future iterations. The researcher also controlled and configured the changes during each of activities, iteration, and phases to ensure it followed the project plan.

- Closed-out the elaboration phase

  The researcher brings the phase to closure by ensuring that all major issues from the previous iteration are resolved, and the state of all artifacts is known.

iii. Software Requirements Workflow

In this workflow, all the requirements of OraCrypt™ product were managed and established to achieve the organization target. The following activities were carried out during the software requirements analysis workflow (*Appendix D-3*).

- Managed changing requirements

  The researcher assessed the impact of requested changes to the requirements, and managed the downstream impact of the changes approved to be action. The researcher evaluated requested changes and determined their impact on the existing requirement set. The researcher has to restructure the use case model based on the changes to requirements.

- Refined the system definition

  The researcher refined the requirements in order to capture the consensus understanding of the system definition by describing the use case flow of events in details, detailing the Supplementary Specifications, and developing a Software Requirements Specification. This work is done by reviewing the existing actor definitions and, then continues with detailing the use cases that have been previously outlined for each actor.

iv. Analysis And Design Workflow

Software analysis and design is conducted to transform the requirements to a design of the system-to-be. It evolves a robust architecture for the system. During this workflow, the design will be adapted to match the implementation environment for designing it for performance. The following activities were carried out during the analysis and design workflow (*Appendix D-4*).

- Defined candidate architecture

  The researcher allocated time to investigate potential candidate architecture. The researcher spent time early in the process to evaluate selecting technologies, and perhaps developing an initial prototype can reduce some major risks for the project. Early elimination of architectural in the project concerns of relationship between the architecture and organizational structures; separation of development concerns, which will provide a basis for separation work, and adherence to standards. The researcher defined the use-case realizations to be addressed in the current iteration

- Analyzed the behavior

  The researcher transformed the behavioral descriptions provided by the requirements into a set of elements upon which the design can be based. The researcher identified the analysis classes that satisfied the required behavior. And then, the researcher determined how these analysis classes fit into the logical architecture of the system.

- Designed the components

  The purpose of this activity is to refine the design of the system. The researcher refined the definitions of design elements by working out the details of how the design elements realize the behavior required of them. The researcher fleshed out the design details for the elements contained in the package by focusing on the recursive decomposition of functionality in the system in terms of classes. The use case realizations must be refined to reflect the evolving responsibilities of the design elements.

- Designed the database

  The researcher identified the design classes to be persisted in a database, and designed the appropriate database structures to store the persistent classes. The researcher defined the mechanisms and strategies for storing and retrieving persistent data in such a way that the performance criteria for the system are met.

The *Lifecycle Architecture* milestone marks the end of the elaboration phase. At this point, the researcher examined the detailed system objectives, scope, the choice of architecture, and the resolution of the major risks. The researcher has tested and evaluated the executable prototypes (R1.0 Release), and it signifies that the major risk elements have been addressed and credibly resolved. The deliverables that considered take into account during the elaboration phase are listed as following:

- Iteration Plan for Construction Phase
- Software Architecture Document
- Architectural Prototype

3.3.3 Construction Phase

The goal of this phase is to decide if the software is ready to be deployed. In this phase, the researcher managed the resources and controlled the operations to optimize the schedules and emphasized the quality of the product. To achieve the degree of parallelism in the development of OraCrypt™ product, all components and software features are implemented and integrated into the software product. The following are the details of general activities for each workflow involved in iterations defined in the construction phase that are environment, project management, software requirements, and analysis and design workflows.

i. Environment Workflow

The purpose of this workflow (*Appendix D-1*) is to ensure the project environment is ready for the upcoming iteration. The researcher prepared and customized the tools to use within the iteration. The researcher verified that the tools have been correctly configured and installed. The researcher also prepared a set of project-specific templates and guideline to support the development of project within the iteration. All the changes made to the project environment are ensuring properly communicated to the project team.

ii. Project Management Workflow

In this workflow, the researcher focused on refining the Software Development Plan, monitoring and controlling the OraCrypt™ project, and managing the risks. The following activities were carried out during the project management workflow (*Appendix D-2*).

- Refined the SDP

  The researcher refined the SDP, and mapping out a set of iterations using the prioritized use cases and associated risks. The project plans developed at this point are refined after each subsequent of iteration and become more accurate as iterations are completed.

- Updated the iteration plan and risks

  The researcher updated the iteration plan after the previous iteration was assessed, and the project scope and risk reevaluated. The researcher updated the iteration plan based on the new functionality added during the new iteration. The researcher also concerned on the risks that need to be mitigated in the upcoming iteration.

- Monitored and controlled the project

  The researcher monitored the project in terms of risks and objective. The researcher regular reporting the project status, and any change requests are scheduled for the current and future iterations. The researcher also controlled and configured the changes during each of activities, iteration, and phases to ensure it followed the project plan.

- Closed-out the construction phase

  The researcher brings the phase to closure by ensuring that all major issues from the previous iteration are resolved, and the state of all artifacts is known.

iii. Software Requirements Workflow

In this workflow, the requirements of OraCrypt™ product were managed and established to achieve the organization goals. This is the process of deriving the system's requirements through observations of existing system. By doing this, the researcher determined the features that are absolutely necessary for the application. The following activities were carried out during the software requirements workflow (*Appendix D-3*).

- Updated the boundaries of the system

  During the third iteration, which is *C3 Iteration*, the researcher defined the new system boundaries that describe an envelope in which the solution system is contained. The researcher have defined two (2) new use cases that are represented in use case diagram as shows in *Appendix C*. The following are the description of each new use case involved.

  a. Use Case: Initialize System

     This use case shows the system initialization during the installation process of the OraCrypt™ product. It decomposed into two categories based on the user authorization, which are initialize system for Security Manager and authorized user. This process shall be done only once during system installation either on the server or client. Overall process shall be done by the system and only a few part need for user interaction.

  b. Use Case: Manage Report

     This use case shows the generating of system reports that can be categorized as encryption, decryption, key generation, user information, and system log report. User can view the report to get the better overview of the process selected.

- Managed changing requirements

  During the system development, the requirements frequently evolve. The researcher managed and controlled the requirements change include activities such as established a baseline, determined which dependencies are important to trace, and established traceability between related items.

iv. Analysis And Design Workflow

Once the necessary requirements have been collected, then the researcher proceeds to the next step in the model. This step is the analysis and design of the application. The design model represents the intent of the implementation, and is the primary input to the implementation workflow. The following activities were carried out during the analysis and design workflow (*Appendix D-4*).

- Analyzed the behavior

  The researcher transformed the behavioral descriptions provided by the requirements into a set of elements upon which the design can be based. The model elements are analyzed and refined by allocating responsibilities to specific elements (classes), and updated their relationships and attributes. New elements are added to support possible design and implementation constraints.

- Refined use case realization

  The researcher identified the analysis classes from the architecturally significant use cases. And then, the researcher updated the use-case realizations with analysis class interactions.

- Identified the design scheme

  When designing the application, it is important to be aware of the fact that the design process encompasses many aspects of the application. The researcher ensured that the design should correctly implement the specifications and requirements. The researcher broke down the application into user interface design and architectural design parts but the researcher rarely focus on the

design of the entire application. Dividing them into smaller modules make them easier to manage.

a. User Interface Design

The design for the user interface can be done independently. It is absolutely critical that the user interface is intuitive and easy to use. It should include the most basic and fundamental functions.

b. Architectural Design

This is the design of the overall architecture of the application. It consists of various sub-components. The integration of these sub-components together makes up the complete application.

- Designed the components

  The researcher decomposed the application into a set of interacting components. The researcher designed the packages, subsystems and components that contains of design elements in terms of functionality and classes. The use case realizations are refined to reflect the evolving responsibilities of the design elements.

- Designed the database

  The researcher identified the design classes to be persisted in a database, and designed the database structures to store the persistent classes. The researcher defined the mechanisms for storing and retrieving persistent data in such a way that the performance criteria for the system are met.

In the construction phase, the researcher divided the activities into three (3) iterations, which are iteration for developing R1 Beta Version, R1 Release, and R2 Release. The new use case is implemented during each of iteration. In the following subsection, the researcher details all activities involved in each iteration.

3.3.3.1 C1 Iteration

The *C1 Iteration* was implemented in developing R1 Beta Version. This iteration involved environment, project management, software requirement, analysis and design, and implementation workflow. In the implementation workflow, the researcher defined the organization of the code. All the classes and object will be implemented in terms of components and were make as simple as possible. The researcher will test the developed components as units. And then, the researcher integrated the results into an executable system. The following activities were carried out during the implementation workflow (*Appendix D-5*).

- Structured the Implementation Model

  The researcher structured the implementation model to ensure a smooth implementation and integration process for R1 Beta Version. The researcher is moving from the design to the implementation space started by mirroring the structure of the design model into the implementation model. The mapping from the design model to the implementation model may change as each implementation subsystem is allocated to a specific layer in the architecture. Structuring the implementation model generally results in a set of implementation subsystems that can be developed relatively independently.

- Planned the integration

  The purpose of this activity is to plan the integration of the system for the current iteration. Planning the integration is focused on which implementation subsystems should be implemented, and the order in which the implementation subsystems should be integrated in the current iteration. In the iteration plan, it specifies all use cases and scenarios that should be implemented in this iteration. So, the researcher identified which implementation subsystems participate in the use cases and scenarios for the current iteration. The researcher also identified which other implementation subsystems are needed and to make it possible to compile.

- Implemented the components

  The purpose of this activity is to complete a part of the implementation so that it can be delivered for integration. The researcher wrote source code, adapted existing source code, compiled, linked and performed unit tests, as they implement the elements in the design model. The researcher also fixed code defects and performed unit tests to verify any changes. Then the code is reviewed to evaluate quality and compliance with the Programming Guidelines.

- Integrated the components

  The researcher integrated changes from developer to create a new consistent version of an implementation subsystem. The researcher tested the components to ensure that the components perform to its specification. And then, the researcher delivered the implementation subsystem for component integration.

3.3.3.2 C2 Iteration

The *C2 Iteration* was implemented in developing R1 Release. This iteration involved environment, project management, software requirement, analysis and design, implementation, and testing workflow. In this section, the researcher details the activities involved in implementation and testing workflow.

i. Implementation Workflow

After the researcher has completed the designs for the application, the next step is to use that design scheme to implement the purpose of each design component. This workflow consists of writing, compiling, and executing source code. All the classes and object will be implemented in terms of components. The researcher tested the developed components as units. And then, the researcher integrated the results into an executable system. The following activities were carried out during the implementation workflow (*Appendix D-5*).

- Structured the Implementation Model

  The researcher structured the implementation model to ensure a smooth implementation and integration process of R1 Release. The researcher moved from the design to the implementation space by mirroring the structure of the design model into the implementation model to produce a set of subsystems that can be developed relatively independently.

- Planned the integration

  The researcher planned the integration of the system (R1 Release) for the current iteration. The integration plan specified all use cases and scenarios that should be implemented in this iteration. The researcher also identified other implementation subsystems are needed in developing R1 Release.

- Implemented the components

  The researcher wrote the source code, adapted the existing source code, and compiled it. The researcher also fixed the code defects and performed unit tests to verify any changes. Then, the code is reviewed to evaluate quality and compliance with the Programming Guidelines.

- Integrated the components

  The researcher tested the components to ensure that the components perform to its specification. Then, the tested components are delivered for component integration.

- Integrated the system

  The purpose of this activity is to integrate the implementation subsystems to create a new consistent version of the overall system (R1 Release). The researcher combined the components and subsystems into an executable build set and delivered them incrementally into the system integration workspace. Hence, the researcher integrated the system by implement the bottom-up based with respect to the layered structure, to ensure that the versions of the subsystems are consistent.

ii. Testing Workflow

This workflow acts as a service provider to the other workflows in many respects. The researcher focused on the evaluating and assessing product quality. The researcher validated and proved the assumptions made in the design and requirement specifications through concrete demonstration. The researcher proved that the software product works as designed and all the requirements are implemented appropriately. The following activities were carried out during the testing workflow (*Appendix D-6*).

- Defined evaluation mission

  The researcher identified the test effort for the iteration, and gained the agreement with user on goals and scopes that are direct to the test effort. Then, the researcher outlined the approach that will be used for the testing.

- Verified test approach

  The researcher demonstrated that the techniques outlined in the test approach would facilitate the planned test effort, produced accurate results, and is appropriate for the available resources. The researcher established the basic infrastructure to enable and support the test strategy by identified the scope, boundaries, limitations and constraints of each technique.

- Tested and evaluated

  The researcher achieved the test efforts that enable a sufficient evaluation of the items being targeted by the tests. The researcher provided ongoing evaluation and assessment of the target test, recorded the information necessary to diagnose, and resolved any identified issues. The researcher also provided the feedback on the most likely areas of potential quality risk.

- Achieved acceptable mission

  The researcher delivered a useful evaluation result to the user of the test effort. The researcher prioritized the set of tests that must be conducted, and the test results are evaluated against testing objectives to achieve the evaluation

mission. The researcher also advocated the resolution of important issues that have a significant negative impact on the evaluation mission.

3.3.3.3 C3 Iteration

The *C3 Iteration* was implemented in developing R2 Release. This iteration involved environment, project management, software requirement, analysis and design, implementation, and testing workflow. In this section, the researcher details the activities involved in implementation and testing workflow.

i. Implementation Workflow

This workflow consists of writing, compiling, and executing source code. In this workflow, the researcher wrote the code to allow the components to execute it task. The following activities were carried out during the implementation workflow (*Appendix D-5*) at the last iteration.

- Structured the Implementation Model

  The researcher structured the implementation model to ensure a smooth implementation and integration process of R2 Release. The researcher mirrored the structure of the design model into the implementation model. By structuring the implementation model, the researcher produced a set of implementation subsystems that can be developed relatively independently.

- Planned the integration

  The researcher planned the integration of the system (R2 Release) that specified all use cases and scenarios that should be implemented in this iteration. The researcher also identified other subsystems are needed in developing R2 Release. The results of this planning should be reflected in the SDP.

- Implemented the components

  The researcher wrote the source code, adapted the existing source code, compiled, linked it to allow the component to execute it task. The researcher also fixed the code defects and performed unit tests to verify any changes in the implementation model. Then, the code is reviewed to evaluate quality and compliance with the Programming Guidelines.

- Integrated the components

  The researcher tested the components to ensure that the components perform to its specification, and then the researcher released the tested implementation component for component integration.

- Integrated the system

  The purpose of this activity is to integrate the implementation subsystems to create a new consistent version of R2 Release. The researcher combined the components and subsystems into an executable build set and delivered them incrementally into the system integration workspace. In the process of resolved any merge conflicts, the researcher ensured that the versions of R2 Release are consistent and performed to its specification.

ii. Testing Workflow

After implementing the application, the researcher should have a program that is executable. Before the researcher rushed into publishing the product to the mass market, it is absolutely essential to subject the program to various testing. There are two (2) types of testing that the application can undergo. First type is debugging. Debugging simply means that the researcher subject the application to various testing scenarios to determine if there exists any kind of logic or programming errors. The second type is validation of the requirements. This means that the researcher created test scenarios with a different purpose in mind. This process is designed to determine whether the software application is meeting its requirements and services originally stated. There are evident in the testing that the services are indeed present and functional as well. This also determined

the inaccuracy of that functionality. The following activities were carried out during the testing workflow (*Appendix D-6*).

- Defined the evaluation mission

  The researcher setup the system-testing environment, identified the test effort, and gained the agreement with user on the goals and scopes that are direct to the test effort. The researcher outlined the approach that will be used for the testing.

- Verified the test approach

  The researcher demonstrated that the techniques outlined in the test approach would facilitate the planned test effort, produced accurate results, and is appropriate for the available resources. The researcher implemented the test infrastructure to verify that the test approach will work.

- Validated the build stability

  The researcher validated that the build is stable enough for the detailed test and evaluation effort to begin. The researcher referred this work as acceptance into testing that helps to prevent the test resources being wasted on a futile and fruitless testing effort.

- Tested and evaluated

  The researcher executed the test cases, provided ongoing evaluation and assessment of the target test, recorded the information necessary to diagnose, and resolved any identified issues. The researcher also provided the feedback on the most likely areas of potential quality risk.

- Achieve acceptable mission

  The researcher determined if the software source code satisfied system requirements allocated to software and software requirements specifications documents, based on testing result. The test results are evaluated against testing objectives to achieve the evaluation mission. The researcher advocated

the resolution of important issues that have a significant negative impact on the evaluation mission.

The *Initial Operational Capability* milestone marks the end of the construction phase. At this point, the product is ready to be handed over to the transition phase. All functionality has been developed. The researcher have tested and evaluated the executable prototypes, and it signifies that the major risk elements have been addressed and have been credibly resolved. In addition to the software, a user manual has been developed. The deliverables that considered take into account during the construction phase are listed as following.

- Iteration Plan for Transition Phase
- Software Design Description
- R1.0 (Beta Release)
- Release 1.0
- Release 2.0
- Test Cases

3.3.4 Transition Phase

The transition phase ensured that the software is available for its end users. The transition phase includes testing the product in preparation for release and making minor adjustments based on user feedback. At this point in the lifecycle, user feedback needs to focus mainly on fine-tuning the product, configuring, installing, and usability issues. This phase focuses on the required activities to lace the software into hands of the user. In this phase, the researcher defined an iteration called *T1 Iteration*.

3.3.4.1 T1 Iteration

T1 Iteration was implemented in deployment of R1 Release and R2 Release. This iteration involved project management, software requirement analysis, and

deployment workflow. In this subsection, the researcher details all activities involved in this iteration.

i. Project Management Workflow

In this workflow, the researcher monitored and controlled the OraCrypt™ project, and managed the risks occur in the development of this system. The following activities were carried out during the project management workflow (*Appendix D-2*).

- Monitored and controlled the project

  The researcher monitored the project in terms of active risks and objective measurements of progress and quality. The researcher regular reported the project status, and any change requests are scheduled for the particular iteration. The researcher also controlled and configured the changes during each of activities, iteration, and phases to ensure it followed the project plan.

- Closed-out the transition phase

  The researcher brings the phase to closure by ensuring that all major issues from the previous iteration are resolved, and the state of all artifacts is known. Any deployment problems are addressed.

- Closed-out the project

  The researcher ensured that the project is formally accepted. The researcher archived all project documentation and records. The researcher prepared the final status assessment, which if successful, the user formally accepts ownership of the software product. In the end, the researcher signed the agreement with user that all contracted deliveries have been made, meet the contracted requirements and are accepted into ownership by the user. The Project Manager then completed the closeout of the project by disposed of the remaining assets and reassigned the remaining staff.

ii. Software Requirements Workflow

In this workflow (*Appendix D-3*), all the system requirements were managed and established to achieve the project goals. Because of the requirements frequently evolve, the researcher managed and controlled the requirements change include activities such as established a baseline, determined which dependencies are important to trace, and established traceability between related items.

iii. Deployment Workflow

The deployment workflow describes the activities associated with ensuring that the software product is available for its end users. There are two (2) modes of product deployment that have been implemented in this project, which are custom install and shrink wrap product. There is an emphasis on testing the product at the development site, followed by beta testing before the product is finally released to the user. The following activities were carried out during the deployment workflow (*Appendix D-7*).

- Planned the deployment

  The researcher planned the product deployment that needs to take into account how and when the product will be available to the end user. Deployment planning requires a high degree of user collaboration and preparation. It is concerned with establishing the schedule and resources for development of end-user support material, acceptance testing, and production, packaging and distribution of software deployment units to ensure that end users can successfully use the delivered software product.

- Developed the support material

  The researcher produced the collateral needed to effectively deploy the product to the user. The researcher developed the support material that covers the full range of information required by the end-user to install, operate, use, and maintain the delivered system. It includes the training material for all of the various positions to effectively use the new system.

- Managed Beta testing

  This activity solicited feedback on the product from the users while it is still under active development. The beta test begins partway into the iteration, and may continue until the end of the iteration. The researcher runs the beta test and, in the case of shrink-wrap products, the researcher deal with the manufacturers to ensure that adequate quality is achieved in the product.

- Managed the acceptance test

  This activity ensured that the developed product fulfills its acceptance criteria at both the development, and target installation site and deemed acceptable to the user prior to its general release. The researcher organized the installation of the product on the test environment configurations that represents an environment acceptable to the user. Once installed, the researcher runs through a pre-selected set of tests and determined the test results. Then, the researcher reviewed the test results for anomalies.

- Packaged the product

  This activity described the necessary activities in creating a *shrink-wrapped* product. The idea is to take the deployment unit, installation scripts, and user manuals, and then packaged them for mass production.

- Managed the product baseline

  This activity ensured that all developed artifacts are captured and archived, at given points in time, as a basis for further product development. A baseline identifies one and only one version of an element. When the combined set of artifacts reached a specified level of maturity, the artifacts will be baselined to assist managing availability for release and reuse. Then, the artifacts are available for released and reused in the subsequent project iterations or other projects.

This final iteration in the transition phase culminates in the delivery to the user of a complete system with functionality and performance as specified, and demonstrated in

acceptance testing. The user takes ownership of the software after a successful acceptance test. The following are the artifact that will be concerned during the transition phase.

- Software User Manual
- Installation Instructions
- Release 2.0

3.4 Software Technique

Object-Oriented (OO) methodology is a new system development approach encouraging and facilitating reuse of software components. With this methodology, a system can be developed on a component basis, which enables the effective reuse of existing components and facilitates the sharing of its components by other systems. Through the adoption of OO, higher productivity, lower maintenance cost, and better quality can be achieved. OO is a new system development approach, focusing on the objects and classes in a system. The object-oriented means that the researcher organized the software as a collection of discrete objects. An object is defined as a software package that contains data and methods. The object can represents both a physical or conceptual item. A class is a reusable description that can be used to builds objects. OO codifies software engineering principles in terms of object and class abstraction.

Besides that, OO models represent real objects in real world problems. Real world objects will have certain properties and attributes that are mapped to methods and data. An object that comprises of both data and functions can only be accessed via the methods that make it publicly available. This is to ensure that all details of its implementation are hidden from all other objects. This strong encapsulation provides the basis for the improvements in traceability, quality, maintainability and extensibility of object-oriented software.

The main objective of OO is enables the construction of new business solution from existing components. The software components can be assembled to from a complete software or application. By using OO, a complete development can be breaking down the complex solution into several different components and it allows several components to be combined together to meet and fulfill the user's requirements. It is thus concluded that the OO is the best approach in developing OraCrypt™ product that is to transform the project development style from traditional structured based into object-oriented based. The following benefits can be achieved in developing OraCrypt™ product by adopting the OO.

- Facilities reuse

  The obvious way to produce software efficiently is to reuse existing software component rather than rewrite them from scratch every time. With this approach, a system can be developed on a component basis that enables the effective reuse of existing component.

- Improved productivity

  OO can improve the productivity when facilities reuse is implemented in software development. It helps in ensure the software development can be finished during specific time in schedule.

- Delivered high quality system

  The quality of software can be improved as the system is built up by integrating the existing component, which are well tested and proven of performance and capability. Every component is carefully tested in isolation before being plugged in the system.

- Lower maintenance

  It ensures the impact of change is localized and the errors that occur in the software are easily to be traced. As a result, the maintenance cost is reduced.

- Managed complexity of software

  The complex development can be better managed by breaking down the complex solution into different components or modules.

3.5 Software Tools

There are a wide range of OO methodologies, which have different views of development process. To achieve the target in implementation of RUP software process and OO methodology, Unified Modeling Language (UML) is used throughout the development of OraCrypt™ product for specifying, visualizing, constructing, and documenting the deliverables of software product. It enables all users that are involved in software development, documentation, and described the software in standard way.

UML is a process independent. It is not tied to any particular software development life cycle. This modeling technique is refers to the use of notation to represent both the object-oriented analysis and object-oriented design model. Notations play an important part in methodology because in order to ensure UML can provide the most benefit, software process must also be implemented during the software development. The benefits of UML are listed as following.

- UML is easy enough to use and provides clear thought.
- UML is expressive enough to express the design aspects of software.
- It unambiguous features helps to solve the misunderstandings.
- Supported by suitable tools.

3.6 Problem Solving Methodology

Software development always running into problems when the development is carried out without a proper software process follows the defined methodology. However, OO paradigm can be better use to overcome the problems that describe below.

- Conformity

  Software should work when it is required to fit together with the existing system. This includes software and hardware interfaces that need to be interface with the software. This conformity can be achieved through the use of OO approach where each of software components can be modeled clearly.

- Changeability

  It is inevitably occur that user always demand major changes to the software. Therefore, the changes are easier to be tackled and organized through the use of OO methodology because of its key features facilitate reusability and maintainability aspects.

- Complexity

  Software is undeniably a complex structure made by human beings. Huge software may leads to a problem in an attempt to understand the software in it is entirely. It is also affects the management of the process. Therefore, the complexity can be better managed by adopting an OO approach, which can help reduce the complexity.

- Invisibility

  A problem with the essence of software is that invisible and invisualizable. This problem can be overcome by using an OO approach to visualize the software model and components by using the UML diagram. Therefore, it provides the better means of communication between the user and researcher.

CHAPTER 4

DATA AND DISCUSSION

3.7 Performance and Evaluation

Computers and the Internet are being utilized in almost all fields and formed as part of our life. The main reason for using computer technology is to simplify works and to perform automated tasks faster and more efficiently. In addition, in light of today's state of computer breaches, the security of computers and its information held within is becoming more significant. This is why performance measurement and computer security is taken so seriously by computer users. Even though performance measurement usually compares only one aspect of computers, the speed, this aspect is often dominant especially when considering their security. Therefore, this chapter is set to describe computer performance with due respect to its security. This chapter will cover the central issue of performance evaluations and benchmarking in multilevel secure databases and in specific for the new multilevel security scheme. Section 4.2 discusses the currently available benchmarking methodologies for multilevel secure databases. Section 4.3 introduces the benchmark methodology appropriate in this research including the benchmark model (section 4.3.1), the benchmark environment (section 4.3.2), the benchmark database (section 4.3.3), and the benchmark procedure (section 4.3.4). The set of performance metrics (measurable results) necessary for performance evaluation is given in Section4.4 while Section 4.5 presents their results and observations. Section 4.6 provides an overall evaluation of the new multilevel security scheme with particular emphasis on its database encryption algorithms. A summary of the chapter is given in Section 4.7.

2.3 4.1 Benchmarking Multilevel Secure DBMS and RDBMS.

In a Multilevel Secure DBMS (MLS/DBMS) performance is influenced by addition factors like the potential overhead associated with enforcement of security and various security options. The security-related architectural and functional factors in MLS/DBMS that affect performance include, among others, the distribution of data among security levels, the session levels at which queries are run, and how the database is physically partitioned into files. Therefore, there is a need to develop performance measures that help characterize the performance aspects of MLS/DBMS. Doshi et al. (1994) presented such a benchmark methodology to measure the performance of MLS/DBMS using the MITRE Benchmark. MITRE started its benchmarking effort in 1991 with the objective to develop a benchmark for MLS/DBMS general performance

measurements and can be tailored for specific application environments (Schlipper et al., 1992). The MITRE benchmark modifies the Wisconsin Benchmark's test query suite (measure the performance of RDBMS) to measure the effect of security-related factors on the performance of MLS/DBMS. Doshi et al. (1994) presented a modified MITRE benchmarking methodology using a modified test database, a modified test query suite, and introduced three performance metrics (uniformity, scale up, and speed up) that characterize DBMS performance with varying data distributions. The modified MITRE benchmarking methodology focused on the query processing in MLS/DBMS with security factors including access control functions and architectural modifications to support security enforcement. The modified test database consisted of relations patterned after the Wisconsin benchmark relations but with an added security label attribute, rowlabel. This attribute was used to store the security level of each tuple in the multilevel relation and to implement various distributions of security label values. The modified test query suite consisted of the set of original experiments specified in the MITRE Benchmark and additional experiments to study the impact of the number of security levels, polyinstantiation, unequal distribution, etc., based on the modified test database. The resulting test query suite contained projection, selection, join, aggregation, and sort queries. All experiments were intended to run on a set of three relations with varying number of tuples, that is, a one-thousand tuple relation, a five-thousand tuple relation, and a tenthousand tuple relation.

2.4 4.2 The Benchmark Methodology

The benchmarks that have been proposed for these systems investigate special aspects like various server architectures and clustering strategies, distributed computing, and the use of database computers but the mainstream concentrates on taking times of database operations. In line with our research objective of conducting performance and benchmarking multilevel secure database systems are the benchmarking effort undertaken by Doshi et al. (1994) to measure the performance of MLS/DBMS using the MITRE Benchmark. However, their modified MITRE benchmarking methodology focused on the query processing with security factors like access control functions (implemented through row label technique) and architectural modifications to support security enforcement but do not include the modifications needed for database

encryption scheme integration. Their benchmark was limited to measuring the speed of the query processing capabilities of the system. The benchmark database was based on the Wisconsin benchmark relations (DeWitt, 1991) with an added security label attribute. But as we know, the basic structure of any database relations in a DBMS requires some modifications to accommodate the data encryption requirements. The Wisconsin benchmark relations are highly suitable for OLTP applications and the general access control security evaluation.

We are proposing a new benchmark methodology for the evaluation of a database encryption scheme that is independent of its underlying DBMS, the OLTP applications, and with mandatory access control (MAC) implementation. By independent of any OLTP applications, we mean that the encryption scheme is functioning independently and is not part of any application systems. Security access controls through MAC is achieved through the proper generation of cryptographic keys. The objective of the proposed benchmark methodology is to measure the performance of the database encryption scheme per se, and not to measure the database system or any application that may run on top of it.

- The scheme architecture – how the scheme was designed,
- The workload – how much encryption or decryption is needed,
- The environment – the architecture and attributes of its resources,
- The encryption algorithm – to convert the plaintext into ciphertext, and
- The key management – to drive the encryption system.

In summary, in this research we postulate desiderata for a general purpose benchmark for the performance evaluation of a database encryption scheme. This general purpose benchmark will be modeled based on the above five factors. Since the implicit purpose of all benchmarks is performance prediction, the ultimate goal of any benchmark is to correlate closely with specific application performance, not with other benchmarks. This is the first of a series of steps on the way towards the benchmark system we are aiming at. The following sections will described in detail the above mentioned benchmark methodology including its model, its database and the data distributions, its procedure, and finally the benchmark results and observations.

2.5 4.3.1 The Benchmark Model

The benchmark model as shown in Figure 4.1 illustrates the benchmark methodology described above. As can be seen in the figure, the Database Encryption Scheme Performance is a direct function to the workload and the efficiency of the encryption algorithm. The overall efficiency of the encryption algorithm depends on factors including the design and architecture of the scheme itself, its key generation capability of the key management module, and of course the computer hardware and software platforms the execution takes place. Due to limited resources and time constraint, the benchmarking effort was conducted on only one database server. This essentially neutralized the environment factor. We further limit the maximum processing time for each run of the system to be less than or equal to one-hour of our manually clocked time and subjected the encryption process to utilize only one encryption key. What this means is that we minimized the effect of the Key Management function of the system. Based on the above conditions, we are essentially left with the key factor, the time, to benchmark the scheme based on the performance of the encryption algorithms with a given workload using the same architecture. Though the maximal clocked time was limited to one-hour, we are of the opinion that such a benchmark results produced should be interpreted as equivalent to processing many user actions in a given working environment. This is to say that the benchmarking was done just like the concept of “brute force” in cryptographic key search. We were able to do this since we operated the encryption schemes in this benchmark tests as a stand-alone and independent system.

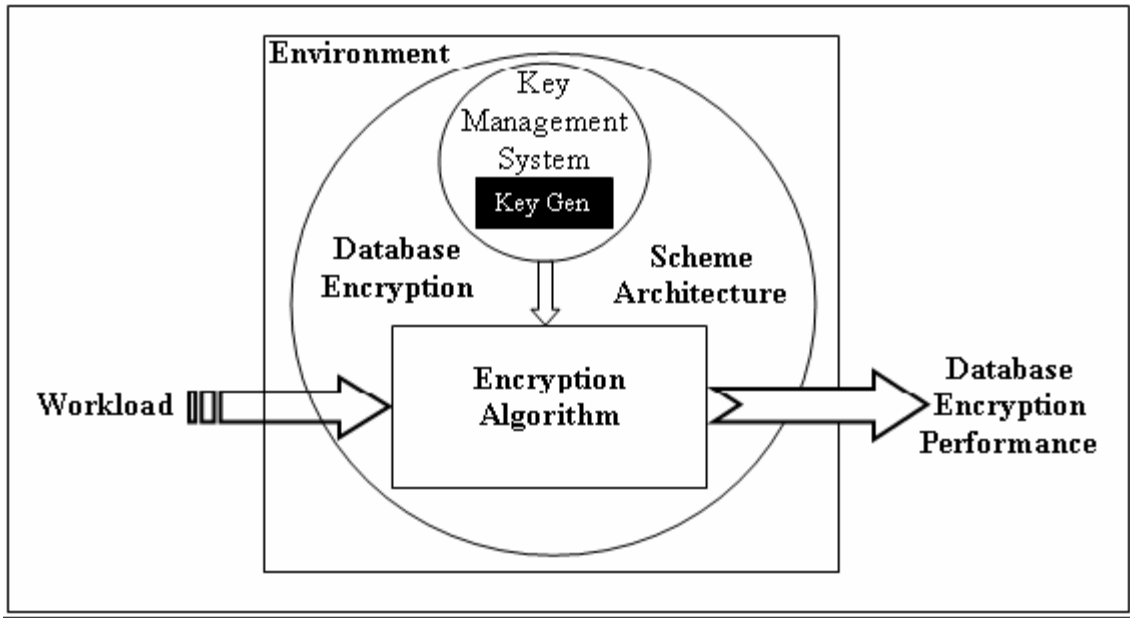


Figure 4.1: A model of the benchmark methodology

2.6 4.3.2 The Benchmark Environment

The benchmarks reported here are conducted in the single-user mode using a client/server configuration with client and server processes on the same workstation. This is of course the best-case analysis, in contrast to a multi-user mode where the amount of traffic, interferences and keeping track of transactions and databases may produce a worst-case analysis. The benchmarking results should therefore be used as an upper bound of the system performance. We study the performance of the proposed database encryption scheme using the two proprietary encryption algorithms namely, the TSBC and the two versions of TSSC (4LFSR and 8LFSR). We further investigated the scheme's implementation by replacing our proprietary algorithms with those available from a major database vendor. In addition, we investigated two other best known commercial database encryption schemes (versions of 2002) for comparison purposes. For commercial reasons, we would like to make them anonymous and will name them DBV (for the database vendor), and DBE1 and DBE2 for the two commercial database encryption schemes. We further constraint our benchmarking to DES and Triple-DES algorithms for the other schemes since these are the common algorithms available.

2.7 4.3.3 The Benchmark Database

The test data used in the performance evaluation was acquired from a local university's library database. Since this is a real life database and it is a library database of books and the University's patrons and students as clients to the library system, the number of records and the database size were huge. However, in this phase of the benchmarking procedure and due to the reasons stated in Section 4.3.1 above, only a portion of the data in the database were used. Based on the "brute-force" approach taken by the benchmarking effort, we utilized only one selected field to be subjected to the benchmark testing. Only one field was taken into account because the main objective here is to test the efficiency of the encryption algorithms involved and not the SQL logic of the programs. If the later was set as the target objective, then there is a requirement to involve more than one field. Based on the huge number of data involved, we considered this to be appropriated for the targeted objective even the processing is on a single field. A detailed analysis of the encipherment process is considered next.

The benchmark database was subjected to the encryption and decryption by both the block and stream ciphers. As The encipherment of data follows a familiar pattern. For example, block ciphers will encrypt data in fixed block sizes and stream ciphers are essentially block ciphers with the block size of one. Based on the fact that the TSBC algorithm was designed with 128-bits block, we have decided to partition the data analysis into blocks of 16 characters. This is appropriate since the logic design of the encryption scheme for TSBC encryption has to follow this block size. Our scheme was able to work with the DBV algorithms, though they process data in blocks of 8 bytes, because all data to be encrypted were maintained to be multiple of 8's before sending them to the encryption algorithms. The block sizes for DBE1 and DBE2 are transparent to users. The benchmark database consists of three files as follows:

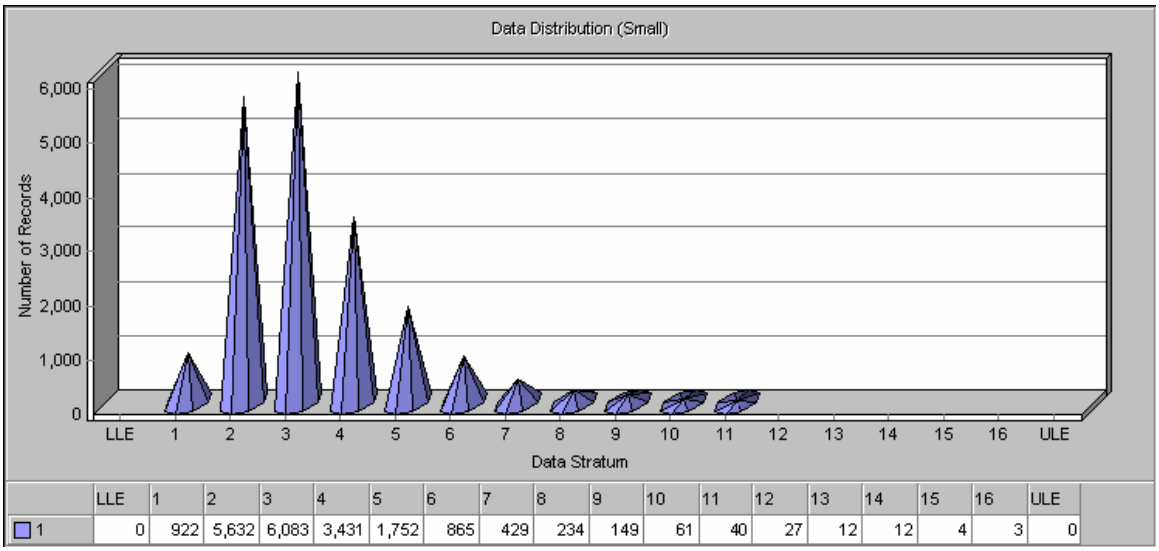
- Small Size in bytes: 890,557 No. of records: 19,656
- Medium Size in bytes: 1,837,746 No. of records: 39,317
- Large Size in bytes: 3,030,673 No. of records: 64,015

These files were extracted from the original database file called bib.txt and loaded into the Oracle database one per round for further processing. The original data file consisted of 261,065 records with 22 fields. However, only the third field, the

TITLE field was subjected to the encryption process. The figures above were written as the number of records since we loaded the data files with all the 22 fields, even though only one field or column was involved. For the purpose of accuracy, simplicity and less confusion, we will refer to them as data items, where possible. In other words, in this phase of the benchmarking the number of records is synonymous to the number of data items. However, the file sizes given above actually refer only for the above field. For the purpose of this benchmarking, we have decided to load about 20,000 records for each round and stopped when the limit of processing (one hour) is achieved. Since we have three different file sizes, there are three rounds conducted and 20 processings (10 for encryption, and 10 for decryption) in each round. In each data loading, there was no duplication in the record samples. For example, when the medium size data was loaded, the previous 19,656 records were also loaded to make up the new record count.

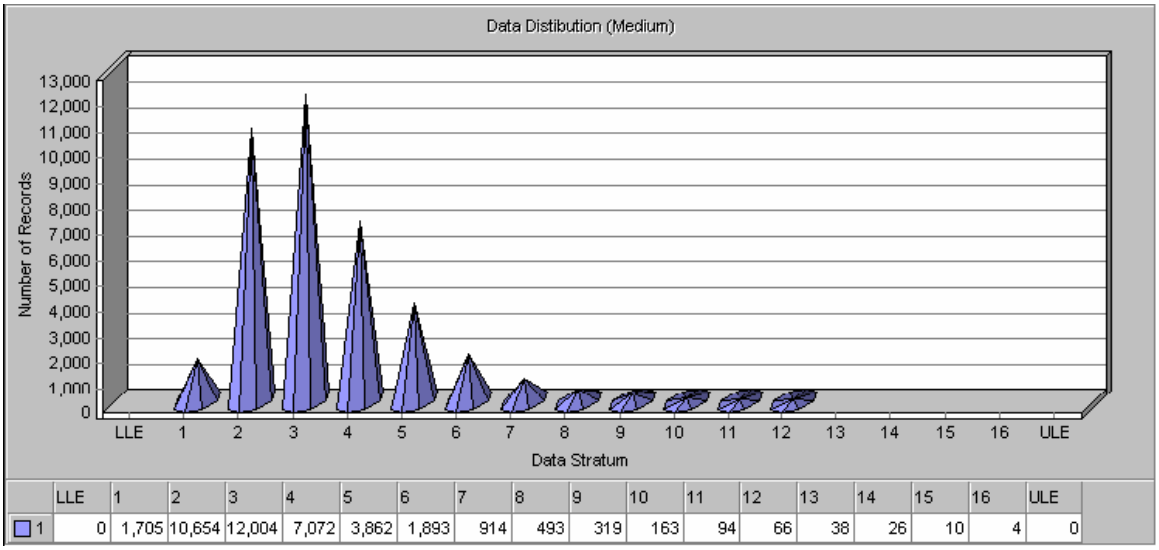
Figures 4.2 to 4.4 below graphed the distribution of data for the various data sizes. In each of the figures, we can notice the distribution of the actual data samples. The X-axis shows the number of records and the Y-axis refers to the data strata with 16 bytes partition per stratum. Coincidentally, all graphs showed the data to be skewed to the left with most of the records fall in stratum 2 or 3. We believed this has the effect on the performance of the scheme compared to if the data was skewed to the right or with a normal distribution. Since the data is from life data, this factor is beyond our control.

Further research can be conducted to analyze this effect but is beyond our current scope. In the Benchmark Data Stratification Reports, details on the distribution of the benchmark data for each size are laid out. We notice in each report, all data are accounted for since the Lower and Upper Limit Exceptions (LLE and ULE, respectively) are zero. There are 16 data stratum and in each stratum (given the lower and upper limits) are given the total number of data items and their percentages, and the subtotal data sizes in bytes for all data items in the stratum plus their relative percentages. At the bottom of the reports are the totals for the whole data items and their sizes.



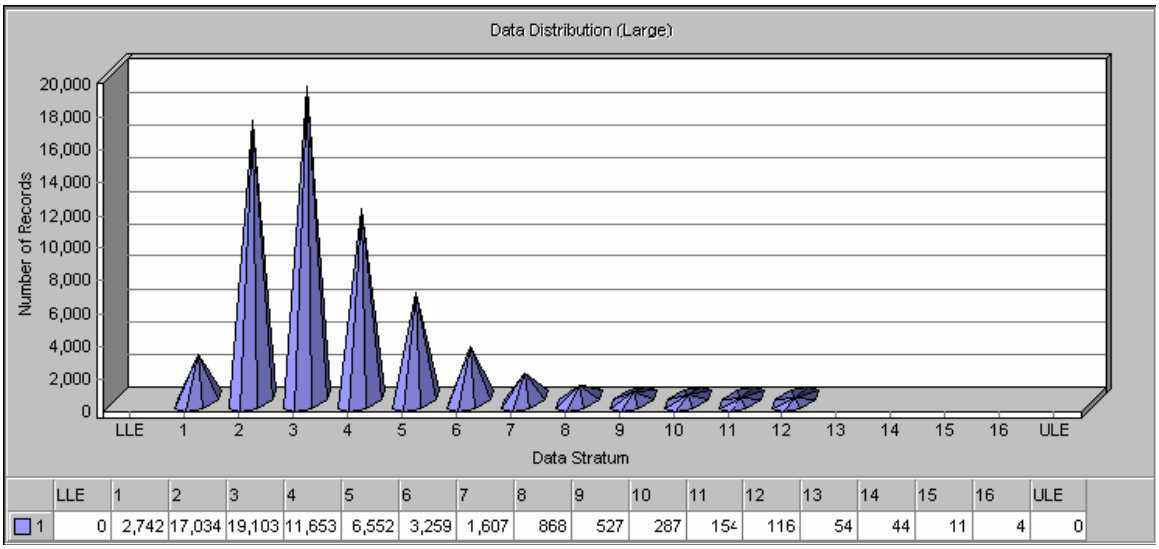
	LLE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	ULE
1	0	922	5,632	6,083	3,431	1,752	865	429	234	149	61	40	27	12	12	4	3	0

4.2: Data distribution for small data size



	LLE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	ULE
1	0	1,705	10,654	12,004	7,072	3,862	1,893	914	493	319	163	94	66	38	26	10	4	0

4.3: Data distribution for medium data size



	LLE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	ULE
1	0	2,742	17,034	19,103	11,653	6,552	3,259	1,607	868	527	287	154	116	54	44	11	4	0

4.4: Data distribution for large data size

2.8 4.3.4 The Benchmark Procedure

The basic procedure used in the benchmarking is to run all the 20 programs stated in Table 4.1, beginning with encryption followed by their pair-wise decryption. In the event an encryption or decryption process is not successful, then the data in the database has to be refreshed or restored to its original plaintext format. We did this by importing the fresh data in binary OS file format which was exported earlier using Oracle's Export utility. The process can be unsuccessful due to one of two reasons:

1) the workload is too much for the algorithm, in this case the system may hang, or

2) the processing time took over an hour of manually clocked time.

In the later case, the performance data was disqualified even though the process went through. We follow strictly the rule set. However, its pairwise process (decryption) not exceeding an hour is taken into consideration. Timing data is obtained by the use of Oracle system calls. We are using the SQL Trace facility and TKPROF program. The two are Oracle's basic performance diagnostic tools that can help monitor and tune applications running against the Oracle Server. The SQL Trace facility provides performance information on individual SQL statements and generates statistics for each statement. From these TKPROF output files, the benchmark data can be analyzed for performance evaluations.

2.9 Performance Metrics

Since this is a singe-user benchmark experiment with the sole objective of performance study for the encryption scheme executed in a stand-alone environment, only the CPU and elapse time measurements for the database "UPDATE" statement are reported. The CPU time measured here is the time in seconds taken by the CPU for the actual execution of the SQL statement which results in the database being encrypted or decrypted. The elapse time is the time in seconds between the initial issuance of the SQL command and the final response from the execution. The SQL UPDATE command represents the database transaction updates by the encryption algorithms and this is proven by the number of rows reported in the trace files. The number of rows reported in

the trace files is the same as those given in the benchmark database. TKPROF output listing consists of the three steps of SQL statement processing:

i. PARSE – translates the SQL statement into an execution plan and checks for the existence of tables, columns, and other referenced objects.
ii. EXECUTE – the actual execution of the SQL statement by Oracle. SELECT statements identify the selected rows and INSERT, UPDATE, and DELETE statements modify the data.
iii. FETCH – only performed for SELECT statements and retrieves rows returned by a query.

From the description above, it is obvious that we are interested only in the EXECUTE step where the actual SQL statement execution takes place and in which the results of the encryption algorithm (to encrypt or to decrypt) modify the data in the database.

2.10 The Benchmarking Results and Observations

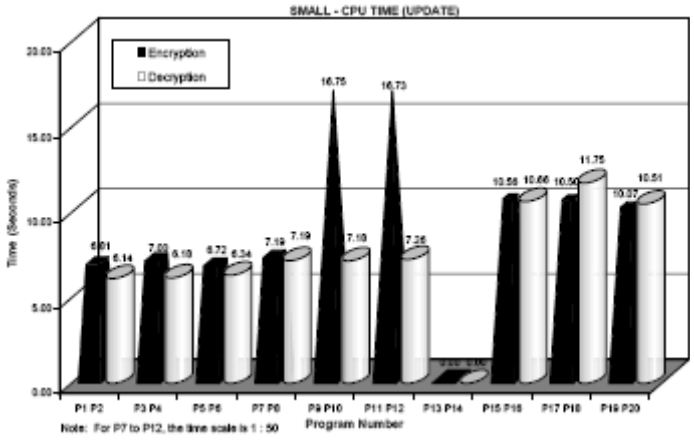
As mentioned in Section 4.3 above, our benchmark methodology serves the purpose to measure the performance of the database encryption scheme per se. We, therefore, have devised the testing criteria in such a way to answer the following three questions regarding the performance of the database encryption scheme:

i. Are there substantial differences in speed or performance between the proposed encryption scheme and other similar schemes?
ii. Is the scheme's performance at an acceptable level, comparatively?
iii. Is the scheme able to withstand the "robustness" test?

This test refers to whether the scheme is able to perform the encryption-decryption process for a given increased in workload or the process takes shorter than one hour, which ever comes first. To answer the above questions, we generated four sets of graphs, the first three derived from the different data file sizes and the fourth is the consolidated graph. Each graph set is presented, along with our analysis and observation, in this section. The first three sets comprise the graphs for CPU and elapse time and the fourth comprises the graphs for CPU and elapse time on encryption, and CPU and elapse time on decryption. To simplify our reporting, the results analysis and observations given below will use the acronym “E” for encryption and “D” for decryption.

2.11 Small Size Graphs

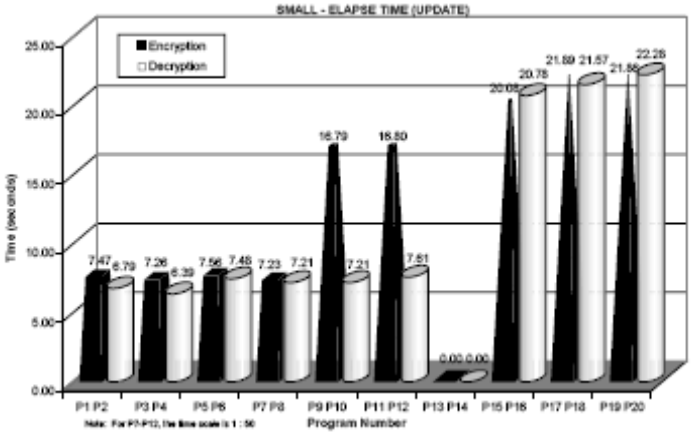
In our first result set, the general observation is that all the test runs were successful except for P13 and P14. P13 and P14 runs represent the scheme from the vendor DBE1. There were no test results obtained (value 0.00 in the graph) because of some reasons. P13 is the encryption by DBE1 scheme and the value 0.00 was because the trace file did not collect any CPU time though the process was well below the 1 hour limit. The value for P14 was zero because the DBE1 scheme was not able to decrypt the encrypted data. As expected, our scheme using TSSC (4LFSR) is slightly faster than the original TSSC (8LFSR) and TSBC for both E and D. However, the E for TSBC (P5) is a bit faster than our expected TSSC (4LFSR) and this may contributed by the fact the data distribution is skewed to the left with most data items falling in the second and third stratum (see Figure 8.2). Our scheme was performing very much better than DBV, DBE1 and DBE2. The note at the bottom left of the graph reminds us the actual values for P7 to P12 is 50 times the figures shown in the graph. For example, the actual value for P9 is 16.75 multiply by 50 which is 837.5 seconds. The reason for doing this was to proportionate the entire graph. This principle applies to all graphs in this benchmark results. On the same token, the DBV-3DES (P9 and P11) suffers greatest amongst the other schemes, i.e. their CPU time clocked were 837.5 and 836.5 seconds, respectively.



Program Performance in Small-size Graph (CPU Time)

The performance of the programs in terms of elapse time is shown in Figure 8.6 below. It was observed that their time follows similar pattern to the CPU time clocked in Figure 8.5. The results from DBE1 scheme still suffer the same problem as above. With the rest of the program performing in similar pattern as in the results above, we noticed

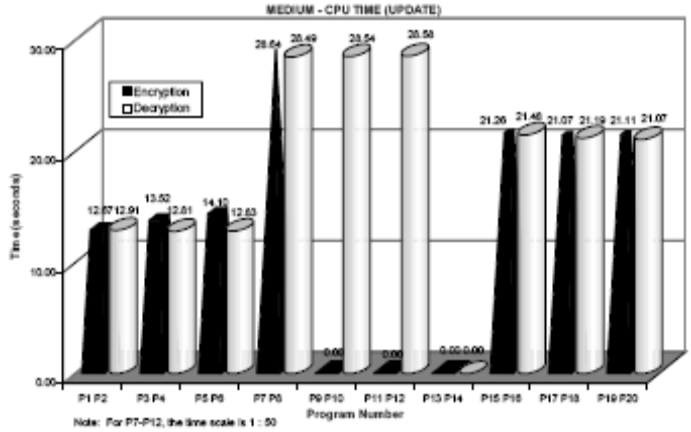
there is almost double in time for all the DBE2 performance. The DBE2 scheme is a full fledged database encryption package and we reckoned the elapse time figures were doubled due to the requirement for the scheme to prepare the base database for E-D. This is a well known fact in any database encryption process.



Program Performance in Small-size Graph (Elapsed Time)

2.12 Medium Size Graphs

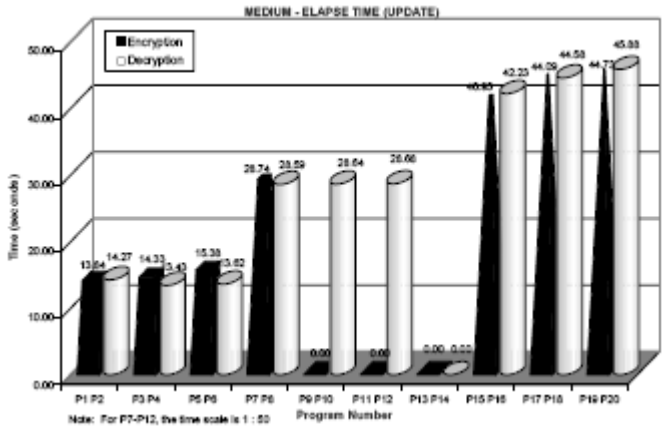
In the medium size database encryption process, our proposed scheme is able to perform as expected. Since the data size increased by about double in size, the effective CPU time is also about double their sizes for the three algorithms, vis-à-vis, TSSC (4LFSR), TSSC (8LFSR) and TSBC. The situation was different when the DBV algorithms were used in our scheme (P7 to P12). It took almost 25 minutes for the DBV-DES algorithm to encrypt or decrypt the medium size data (P7 and P8). The DBV-3DES (2 and 3 key modes) failed the robustness test in this size of data. Their encryption CPU time was greater than an hour (P9 and P11) and is shown as having values 0.00 in the figure. Decryption by DBV-3DES was success in both key modes with CPU time clocked around 25 minutes. The data timing for DBE1 scheme remains 0.00. The observation made in this instance was that the scheme was able to encrypt the medium size data but it was over the one-hour limit (P13). The scheme was not able to decrypt the data to its original plaintext form. The DBE2 scheme did not suffer from any robustness test and the CPU time followed the pattern in the small size data except the time clocked is about double in this situation.



Program Performance in Small-size Graph (CPU Time)

The general observation charted in Figure 8.8 is fairly reasonable and is in agreement with the discussion given above. The proposed scheme had the benchmark database pre-prepared for encryption and suffers no additional overhead as in DBE2 case. That may contribute to the difference in CPU time reported for the two schemes.

The detailed explanations for the rest of the programs are similar to the one given in the CPU time above.



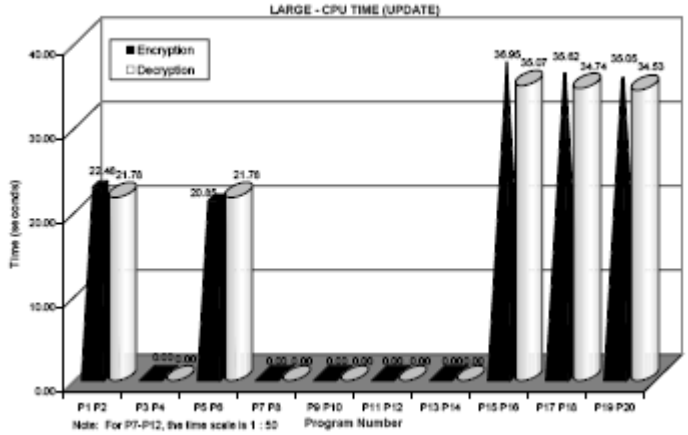
Program Performance in Small-size Graph (Elapsed Time)

2.13 Large Size Graphs

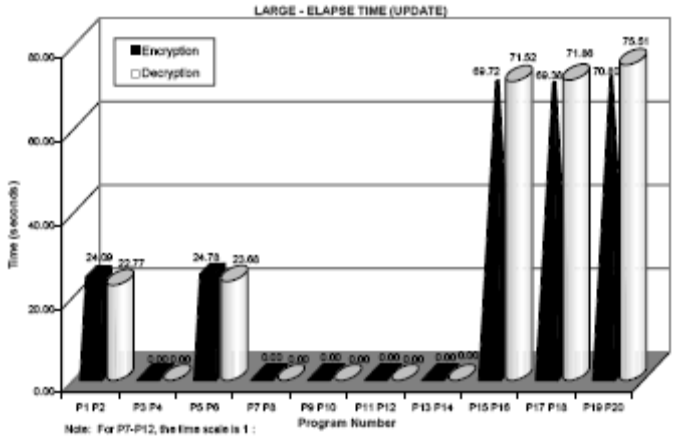
At this stage of the benchmarking effort, the test of robustness is really coming into place. Figure 8.9 below illustrates this point. Half the total number of programs run suffers the robustness test and they include TSSC-8LFSR (P3 and P4), DBV-DES (P7 and P8), DBV-3DES (P9 to P12), and DBE1 (P13 and P14). The programs that survived the robustness test were TSSC-4LFSR (P1 and P2), TSBC (P5 and P6), and the DBE2 scheme (P15 to P20). It is unexpected to notice the TSBC performed better compared to our TSSC-4LFSR. However, this may due to the fact there are more short data items and they lie in the 2nd and 3rd data stratum in the data distribution as shown in Figure 8.4. The cumulative effect of this type of data distribution in the benchmark database is very obvious at this stage of the benchmarking. The numeric stratification report for large data set in Appendix F shows there are 36,137 data items which is 56.45 % of the total records processed.

Our scheme performed better, at the 20 seconds time-line, compared to the DBE2 scheme which performed above the 30 seconds time-line. The observation in performance for elapse time (see Figure 8.10 below) is again similar in pattern and results to the CPU time reported above. However, a marked difference between our scheme and the DBE2 scheme is that our elapse time is still maintained around the 20

seconds time-line. The later scheme's elapse time has shot up to well above the 60 seconds time-line, averaging at the 70 seconds timeline, and the 3DES-3Key program almost reaching the 80 seconds time-line.



Program Performance in large-size Graph (CPU Time)



Program Performance in large-size Graph (Elapsed Time)

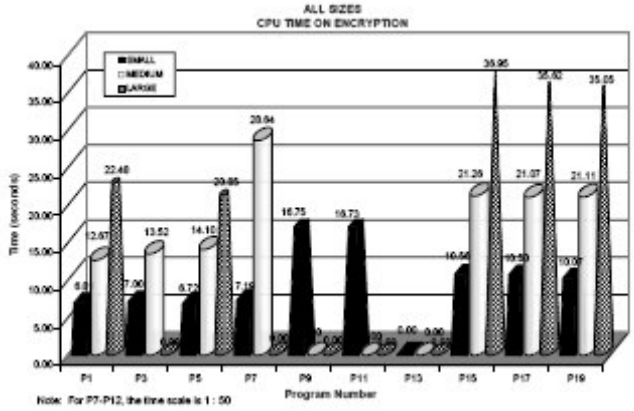
2.14 All Sizes Comparison Graphs

This section reports on the attempt to make a comparative study on performance for all the programs ran in the benchmark in a slightly different perspective. Here, we are interested to conduct a direct comparison among the different program performances by zooming into the specific activities in the encryption process. We compare the performance results for all programs in all database sizes and differentiate them according to the encryption or decryption activities based on CPU or elapse time. Based on this method of categorization, we generated four graphs and their analysis follows. In Figure 8.11, the odd program numbers refer to the encryption process. In the encryption

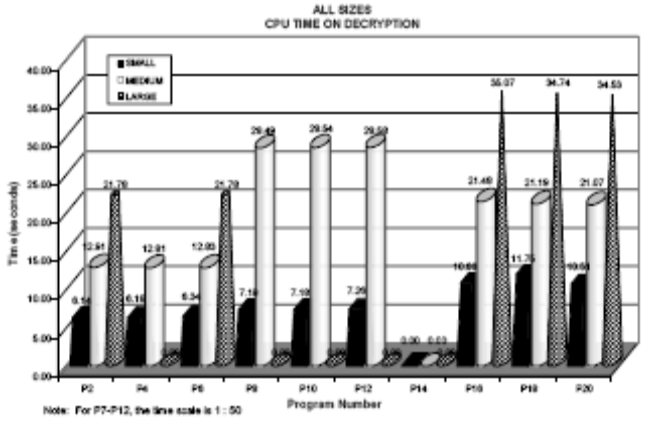
process, all sizes of data in the benchmark database were successfully encrypted though there are 0.00 figures reported in the graph. These zero figures are due to the encryption CPU time exceeding the one-hour limit except DBE1 (P13) where the encryption time recorded manually is well below the hour limit but their CPU time were not recorded in their respective trace files. In other words within the one-hour constraint, TSSC-4LFSR (P1), TSBC (P5), and the DBE2 scheme (P15 to P19) managed to encrypt the all the benchmark database files. The TSSC-8LFSR and DBV-DES were not successful in encrypting the large database. Both the DBV- 3DES 2-key and 3 key only managed to encrypt the small data file, and the DBE1 scheme is more appropriate to be said as "information not available". Based on the successful program runs, we may also extrapolate the fact that the encryption CPU time has a downward trend in processing larger data files, if given this type of data distribution. We may notice this trend by looking at the time-line for the encryption of specific data file sizes in our scheme and yet very obvious in DBE2 scheme.

The pair-wise activity in the benchmarking is the decryption, and this is shown in Figure 8.12 with even program numbers. As mentioned above, encryption of all data in the three data sizes were successful. Therefore, we can conclude that their decryption is either exceeding the one-hour limit or the cipher text data cannot be encrypted. In the figure, we make the observation that not all decryption was successful. The unsuccessful decryptions were TSSC-8LFSR (P4), and DBV-3DES- 3Key (P12) which exceeded the time limit, and DBV-DES (P8), DBV-3DES-2Key (P10), and DBE1 (P14) which their reasons were unknown. In decryption, the proposed scheme managed to pass the benchmark except TSSC-8LFSR which is expected. In using the proposed scheme with DBV algorithms (P8 to P12), they pass the benchmark only for the small and medium file sizes but took very much longer time than expected. We may interpret this as a failure if we compare it to the proposed scheme since the proposed scheme can decrypt the large file size in very much less time than what the DBV can decrypt for the medium file size.

In statistical term, TSSC-4LFSR (P2) decrypted the large data file in 21.78 seconds while the DBV algorithm (P8, P10 or P12) decrypted in more than 1,400 seconds or more than 23 minutes.

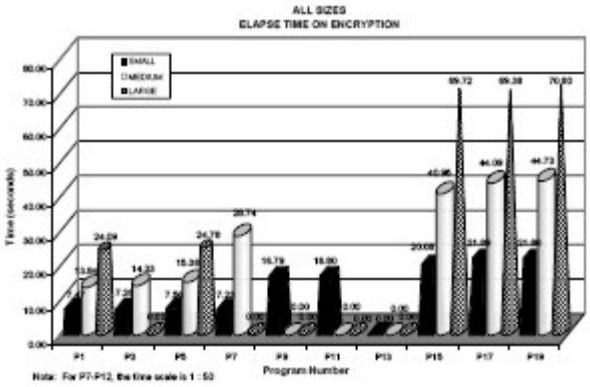


Program Performance in combine-sizes Graph (E-CPU Time)

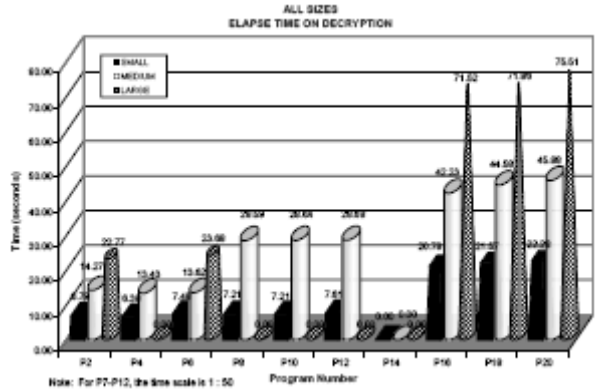


Program Performance in combine-sizes Graph (D-CPU Time)

Not taking into account the DBV time data, since we may interpret it as a failure, the elapse time data shows a very interesting outcome. We observed a big difference in behavior between the proposed scheme and the DBE2 scheme. The elapse time in Figures 8.13 and 8.14 for both the encryption and decryption processes, we noticed that the data timing for the proposed scheme remains almost the same with their respective CPU time. Where else in the DBE2 scheme, the elapse time for both the encryption and decryption processes are almost double compared to their CPU time, respectively. The elapse time for encryption and the elapse time for decryption for both the schemes mentioned above remain the same.



Program Performance in combine-sizes Graph (E-ElapseTime)



Program Performance in combine-sizes Graph (D-Elapse Time)

2.15 Performance Evaluation

The performance evaluation for the proposed scheme, based on the benchmark methodology recommended in the research, can be viewed from two perspectives. First, is within the proposed database encryption scheme per se. The second is to compare it with other available schemes and or algorithms. In the first perspective, we are of the opinion that the three implementations of the scheme were a success. All three implementations were able to encrypt and decrypt data in a reasonable amount of time.

In the second perspective, we would like to know where our scheme stands. To do this, we judged ours with two other commercially available database encryption schemes and also subjected our scheme to use the encryption algorithms available in a commercial RDBMS. The outcome of these comparisons was very encouraging and is in favor of our encryption scheme. Our scheme did extremely well when compared to the

results obtained by subjecting the same scheme to using a DBMS vendor's encryption algorithms. With these explanations and prove, we are of the opinion that the proposed database encryption scheme has a performance advantage and should perform well in a real world situation.

2.16 Summary

This chapter has introduced us to the subject matter of performance and security evaluations for the processing of multilevel secure DBMSs through benchmarking. Specifically, it has presented a new benchmarking methodology applicable in the testing and evaluation of the performance and security of multilevel secure databases consisting of a number of different database encryption algorithms. Since a survey of the current state-of-the-art in benchmarking did not reveal any particular benchmark that can be used directly for the performance and security evaluation of a database encryption scheme, we have proposed a new design for benchmarking database encryption schemes. The proposed benchmark methodology provides a model for the development of a full fledged methodology which we aim to achieve in the long run. However, the current work here served the purpose as an initial step towards the stated objective. The benchmark results and observations proved that the proposed database encryption scheme has achieved a satisfactory level of performance based on the two performance metrics on the premise that the encryption and decryption process were all successful and provides a certain level of security which it is designed for. The two performance metrics are the CPU time and the elapse time measurements. This is of course within the given environment setup, the use of a specific benchmark database, and the benchmarking procedures followed strictly. The implementation of the algorithms is currently software based. Even though the TSBC and TSSC algorithms developed were not implemented in hardware, based on their design it is possible to implement them at the hardware level encryption.

CHAPTER 4

PROJECT DISCUSSION

4.1 Introduction

The OraCrypt™ development process describes the life cycle model for database security projects. This project followed the unified process model, which is a generic process framework that can be specialized for database security software systems. In this chapter, the researcher discusses the results obtained from the development of OraCrypt™ product. It focuses on the analysis of the system development and other deliverables such as documentation produced. Besides that, the constraints faced during the software development and some recommendations in improving the software development will be covered.

4.2 Output Analysis

As mentioned earlier in the previous chapter, the OraCrypt™ product was divided into two (2) main modules, which are key generation, management and distribution module, and encryption and decryption module. Hence, the integration between all modules was very important in order to ensure that the OraCrypt™ product will be capable of handling encryption and decryption process. The development fulfills its functional and nonfunctional requirements that are increased performance, flexibility,

and reliability. As mentioned before, the researcher has implemented RUP software process during the entire development of OraCrypt™ product. The following are the outputs expected for this project:

- An initial version of a commercial database security product based on local expertise and technology.
- An implementation of the newly designed database encryption scheme for Oracle 8i RDBMS systems.
- A subsystem for the generation, management, and distribution of cryptographic keys.
- A benchmarking technique for the performance and evaluation of database security products.

4.2.1 Project Phases And Milestones

The development of the OraCrypt™ project was conducted using a phase approach where multiple iterations occur within a phase. *Table 4.1* describes each phase and the major milestone that marks the completion of the phase.

Table 4.1 Project Phases and Major Milestones

Phase	Description	Milestone
Inception Phase	The inception phase developed the product requirements for the OraCrypt™ project. The major use cases were developed as well as the high level Software Development Plan (SDP). At the completion of the inception phase, the researcher decided to proceed with the project based upon the estimates time and budget.	*Business Case Review Milestone* at the end of the phase marks the Go/No decision for the project budget and time as planned.
Elaboration Phase	The elaboration phase defined and baselined the Software Architecture Document (SAD). This phase analyzed the requirements and developed the architectural prototype (based on the architecturally-significant use cases). At the completion of the elaboration phase, all use cases selected for Release 1.0 will have completed analysis and design.	The *Architectural Prototype Milestone* marks the end of the elaboration phase. This prototype signifies verification of the major architectural components that comprise the R1.0 Release.

Table 4.1 Project Phases and Major Milestones (continue)

Phase	Description	Milestone
	In addition, the high-risk use cases for Release 2.0 have been analyzed and designed. The architectural prototype was tested the feasibility and performance of the architecture that is required for Release 1.0.	
Construction Phase	During the construction phase, remaining use cases were analyzed and designed. The Beta version for Release 1.0 was developed and distributed for evaluation. The implementation and test activities to support the R1.0 and R2.0 releases was completed.	The *R2.0 Operational Capability Milestone* marks the end of the construction phase. Release 2.0 Software is ready for packaging.
Transition Phase	The transition phase prepared the R1.0 and R2.0 releases for distribution. It provided the required support to ensure a smooth installation including user training.	The *R2.0 Release Milestone* marks the end of the transition phase. At this point all capabilities are installed and available for the users.

The following table describes the iterations along with associated milestones and addressed risks.

Table 4.2 Project Iteration with the Milestones and Risks

Phase	Iteration	Description	Associated Milestones	Risks Addressed
Inception Phase	Preliminary Iteration	Defined product requirements, and Software Development Plan.	Business Case Review	Clarified user requirements up front. Developed realistic Software Development Plans and scope. Determined feasibility of project from a business point of view.
Elaboration Phase	E1 Iteration – Develop Architectural Prototype	Completes analysis and design for all R1 use cases. Developed the architectural prototype for R1. Completed analysis & design for all high-risk R2 use cases.	Architectural Prototype	Architectural issues clarified. Technical risks mitigated. Early prototype for user review.

Table 4.2: Project Iteration with the Milestones and Risks (continue)

Phase	Iteration	Description	Associated Milestones	Risks Addressed
Construction Phase	C1 Iteration – Develop R1 Beta	Implemented and tested R1 use cases to provide the R1 Beta Version.	R1 Beta	All key features from a user and architectural prospective implemented in the Beta. User feedback prior to release of R1.
	C2 Iteration – Develop R1 Release	Implemented and tested remaining R1 use cases, fixed defects from Beta, and incorporated feedback from Beta. Developed the R1 system.	R1 Software	R1 fully reviewed by user community. Product quality should be high. Defects minimized. Cost of quality reduced.
	C3 Iteration – Develop R2 Release	Designed, implemented, and tested R2 use cases. Incorporated enhancements and defects from R1. Develops the R2 system.	R2 Software	Quick release of R2 addresses customer satisfaction. All key functionality provided in System by R2 Release.
Transition phase	T1 Iteration – R1 Release	Packaged, distributed, and installed R1 Release.	R1 Release	Two-stage release minimizes defects. Two-stage release provides easier transition for users.
	T2 Iteration – R2 Release	Package, distribute, and install R2 Release.	R2 Release	Two-stage release minimizes defects. Two-stage release provides easier transition for users.

4.2.2 Project Releases

The Software Development Plan addressed the first 2 releases of the OraCrypt™ product. All features critical to encryption and decryption are planned for the first release (R1.0). The planned content of the releases is expected to change as the project progresses. This may be due to a number of business and technical factors. To

accommodate the changes, Rational Requisite Pro was used to manage the product requirements and to keep track of release content.

Release 1 contained the basic functionality as listed below:

- Generate Key
- Encrypt Data
- Decrypt Data

Release 2 included:

- Manage User Information
- Manage Key Structures
- Login
- Initialize System
- Manage Report

4.2.3 Project Deliverable(s)

Besides the OraCrypt™ product, there are several deliverables, which were completed during the development of OraCrypt™ product. The deliverable items are:

- Project Glossary
- Iteration Plan
- Software Development Plan
- Software Architecture Document
- Software Requirements Specification
- Use Case Specification
- Supplementary Specification
- Software Design Description
- Software Test Description / Report
- Software Development File
- Software User Manual

4.3 Project Constraint(s)

During the development of OraCrypt™ product, the researcher faced several problems and constraints in carry out the task given. The researcher had listed the difficulties as following:

i. Time constraints

In the OraCrypt™ product development, it is important for researcher to study in details on technology and concepts that is being used. For the limited time available to carry out the project, the researcher has to consider in time factor and effort so that the system developed within the schedule and achieve all the system objectives.

ii. Lack of experience and knowledge

As the researcher is new in the cryptography technology and database security development, this contributes to small problem in OraCrypt™ product development. The researcher had to take much time for research in development techniques, instead of the time can be used to focus on upgrade the system performance and reliability.

iii. Absence of previous product documentation

The OraCrypt™ product involved specialized cryptography technology and focus on database security application. This is one major constraint because no software document can be used as a reference. As a result, OraCrypt™ product involved had a difficulty in understand the software requirements.

4.4 Project Advantages / Uniqueness

The product is designed with the capability to be incorporated into existing database application system. Therefore, the benefits to the industry namely to all users of Oracle RDBMS without regards to the application system they are using, will be it allows the users to use their existing application system with added capability i.e. encryption and decryption of data.

There are a few competing products namely DBEncrypt and Secure.Data from the USA. The evaluation and comparison with them showed that OraCrypt™ can encrypt and decrypt data at three levels i.e. data item, row and column. None of the other two can and theirs limited to whole column encryption. One other advantage of OraCrypt™ product is that it does not store any crypto keys in the DBBMS. Instead all our crypto keys are external to the security system and will be fetched from secure smartcard systems.

In terms of efficiency, the norm in cryptography is that the data size will increase almost double after encryption. This happens to OraCrypt™ also but the researcher had designed a new 64-bit storage format that will reduced the storage need from double to about 1.33 percent only.

4.5 Project Innovation

This product is a homemade product i.e. a product from the outcome of a PhD research done in Malaysia. The design of the encryption system is an original research and utilized indigenous cryptographic algorithm that was also a product from previous PhD research effort. Therefore, there is no question on the originality of concept and its implementation. It was successfully implemented in Oracle 8i RDBMS using quite a

substantial amount of data obtained from a library information system. There are many outstanding features and is summarized as follows:

i. Able to encrypt and decrypt data at all level i.e. data element, row and column.
ii. Data can be wholly or selectively encrypted at those three levels.
iii. No crypto keys are to be found anywhere in the RDBMS.
iv. All crypto keys are securely stored in smartcards that are using proprietary smartcard operating system.

The software development methodology is back by the software engineers from the Centre for Advanced Software Engineering (CASE), UTM. The researcher was using a component based system development methodology based on object orientation, which will render the system ease of maintenance and adaptability.

4.6 Project Potential

The target markets are all users of RDBMS with or without application system embedded in their DBMS. This includes the military, police, governments, banking and finance, and the private sector at large. This product has international potential especially with import restriction on encryption algorithms and security products like from the USA. With the current development of the product, there are great potential to create employment for its marketing, support and training services since the product is aimed at international market.

One important strategy to be ahead of competitors especially in the international arena is that the researcher designed and developed the product based on sound software engineering practices that allows for easy maintenance, upgrades and adaptability to many types of RDBMS.

4.7 Project Contribution

In this report, the researcher had discussed the issues of multilevel database encryption, encryption in relational database management systems, direct and hierarchical access controls to encrypted databases, cryptographic key management, encryption schemes, and database security in general. Through the development of this project, the researcher had made numerous progress and achievements particularly in the design of a multilevel encryption scheme and issues relevant to it. In the following paragraphs, the researcher present the views of the advancements made thus far and its contributions towards the body of knowledge mentioned earlier.

- A new encryption scheme

  The researcher employed and implemented various concepts and tools from the fields of cryptography, data security, and encryption algorithm design. The new database encryption scheme is able to encrypt data at all levels of the relational database model, not only at the row and column levels but also at the finest level of granularity that is at the data element level. A most significant characteristic about the new scheme is its high security feature contributed by the fact that no cryptographic keys are stored in the scheme, and no security information needed to be stored anywhere in the database as has been the tradition.

- A new approach in accessing and processing of encrypted data

  The new approach controlled access and processing of encrypted data uses Initialization Vectors (IV) for data. In this approach, including the use of IV in key generation, data IV are never secret and do not have any security significance. Because of this, the researcher is able to store them together with the encrypted data. In this scheme, once data are encrypted in the database their corresponding data IV are captured for later decryption and retrievals. In short, this approach is a new solution to the problem of access controls in a multilevel secure RDBMS and it solves the problem of discretionary and mandatory access controls.

- A new key management technique

  The new key management technique allows the generation, management, and distribution of system and user keys in a very secure fashion. For higher security requirements, key management is better done off-line, in a stand-alone mode, and stored securely in tamper proof smartcards. The key management technique based on user IV results in a hierarchical access control to encrypted data. Discretionary and direct access to encrypted information is made possible with the production of shared keys.

- A new and innovative stream cipher design

  Database applications are normally involved in online transaction processing besides the processing of huge amount of data. With the above experience and knowledge, the researcher was able to make a major modification to the basic design of TSSC stream cipher. Because of this great modification, the researcher considered it a new TSSC. The major modifications made were to its basic design in using four (4) linear feedback shift registers instead of eight. The combination of these four (4) registers produces a total key length of 128 bits instead of 189 bits (Part I algorithm) or 349 bits (Part 2 algorithm). The new TSSC algorithm is a very much faster and is yet secure cipher system for the processing of large amount of data in databases at greater speed.

- A local database encryption package

  This invention is a local database security product, with the implementation of local made cryptographic algorithm. This research contributes to the development of local software industry in the fields of database programming, software engineering, and information security in general.

- An independent and secure database security mechanism

  Findings from this research have allowed the building of an independent and secure solution for commercial RDBMS including Oracle. This security mechanism can be added to the existing security features offered by a DBMS as an independent module. With proper design considerations, the security mechanism can be easily

implemented in applications to become as part of its application security procedure and transparent to users.

4.8 Future Work

This research has opened up a new avenue for further research in the role cryptography can play in database security, and in the wider scope of Information Technology Security (ITSEC). Research in database encryption schemes also has evolved beginning from pure cryptographic study for database security to the integration of modern cryptographic technology into database management systems thus providing the required cryptographic supports. The following list provides some insights into the possible future work that may be carried out in this area. The list however is not meant to be exhaustive.

- Security of the scheme

  In the scheme's implementation, user keys are communicated between the smartcard and the encryption module. Creating a secure channel between the client (smartcard reader) and the back-end database machines will render a more secure system. The current implementation of the scheme also assumed the underlying operating system and database system to be at a certain level of security.

- Speed and efficiency of the scheme

  The redesign of the cipher algorithms has helped to increase the speed in the encryption process. Another major factor in implementation is to ensure that the encryption scheme is implemented efficiently and effectively. Further research should be conducted to find out the best implementation methodology to gain maximum performance from the scheme.

- Scheme Implementations

  The current scheme implementation is specific to Oracle8i RDBMS running on Windows 2000 server in a stand-alone single-user mode. Future work can be conducted to implement the scheme on different database platforms running on different operating systems and using different hardware platforms. Besides this, the scheme can be implemented in a networking environment and possibly in a multi-user mode. Testing for effectiveness and efficiency in Internet access to the secure RDBMS should also be conducted.

- Security of the cryptographic keys

  All cryptographic keys are stored in smartcards acquired from the market place. To be more secure, our own smartcard technology needs to be developed to be independent, full control, and free from possible Trojan horses. Further study on the full life cycle of a key definitely can increase the security including the methods for key storage, distribution, and destruction. Proper procedures on issues related to cryptographic keys need to be resolved and standardized.

- Issues in encrypted data

  A better way of storing encrypted data than in Base64 format as proposed in the research should be seek. Most encryption schemes evaluated by the researcher were using the hexadecimal format for their cryptograms. More research should be conducted in the processing of encrypted data including query processing that allows a wider range of operations to be performed on the cryptograms.

- Adaptation of software engineering practices

  To adapt the software engineering practices on working environment, it is suggested that some study has to be done on the software process concepts to find the best solution for the project development in future. Even training can be conducted as to get the benefits of software engineering practices.

CHAPTER 5

CONCLUSION

5.1 Introduction

This chapter summarized the whole research work done in this project emphasized on the project perspective and software engineering perspective. These include a review of the objective and purpose of the research as whole, the methodology used, various decision made, lesson learnt, experience accumulated, the limitations, and most important of all its contributions.

5.2 Conclusion

The primary mission of this research was to give an opportunity to researcher to gain as much knowledge and experience in a real time software development environment in the industry. The research project gave the researcher the opportunity to understand and learn the development lifecycle of the OraCrypt™ product. As stated before, the OraCrypt™ product is developed based on the Window 2000 Server and Oracle platform. This gave an opportunity for the researcher to gain experience in using the utilities on Oracle for compiling and executing a program, SQL script and other related development utilities. In terms of programming skill, the researcher was able to improve the OO programming skills with better understanding by implemented them

during the entire development of OraCrypt™ product. Besides that, the researcher also gained some knowledge in testing activity where the researcher conduct a database benchmarking, test for performance on huge databases.

From perspective of software engineering, this project has adopted a professional software engineering culture in development of OraCrypt™ where it follows a proper software development process. The researcher realizes that to produce a good and quality software, it is important to choose and applied suitable software process. All decision on software development that refers to software methodology makes the decision easy and regular because it considered on all aspects in development.

Instead of technical knowledge is emphasize on, a good development team organization is a factor in identifying the successful of software development. A development environment, make individual more exuberant in doing the task assign to them. From the review or meeting carry out, team member can share their opinion, rise out the problems faced and altogether get a better solution for the problems. Sharing knowledge among team members also helps to save time in development process.

With a proper planning, any projects can complete in expected schedule and budget. The proper planning includes assessment of tasks among development team so that each member can complete the tasks given. From research, it discovers out that software engineering practices still not fully implemented in the software development. Most company in Malaysia, still do not have a standard and does not follow proper development process. Therefore, most project having problem with user requirements and took along time to be completed.

Documentation is very important in communicating ideas as well as to bridge a gap on certain issues either among developers or between developers and the client. An approach to documentation needs some changing so that the effort of producing it is not wasted. This is also to reflect the way the software being engineered. MIL Std-498 is an improved military standard that focused on a tight discipline in software engineering.

All this effort was an indication to a software community that there should be a paradigm shift in looking at the whole software process in general an in particular documentation itself.

Based on this indication, the researcher had tried to figure out a way towards an improvement of documentation that reflects the reality of a software development. To ensure the successful software reuse one need to ensure the success on the reusability of component documentation itself. An appropriate document to start with is a design document due to the reason of its underlying technicality and its usability. Regardless of the documentation standard used, there is a need in revising the total approach to communicate the software engineering. This should be done since the lifecycle of the software need to be prolonged.

In conclusion, the researcher would like to state that this research endeavor has resulted in the establishment of a new database encryption scheme utilizing local cryptographic tools. The new cryptosystem can be well adapted in many database application systems to benefit local researcher and business entities. With due considerations, the research effort can be transformed into a commercial product providing security solutions to organizations in protecting their precious information in databases.

REFFERENCES

Abbey, M., Corey, M. J., and Abramson, I. (1999). *Oracle8i: A Beginner's Guide.* Oracle Press Edition, Berkeley, C.A.: Osborne McGraw-Hill.

Abrams, M.D., Jajodia, S. and Podell, H.J. eds. (1995). *Information Security: An Integrated Collection of Essays.* Los Alamitos, California: IEEE Computer Society Press.

Al-Salqan, Y.Y., Jagannathan, V.J., Davis, T., Zhang, N. and Keddy, Y.V.R. (1996). Security and Confidentiality in Health Care Informatics. *Proceedings of the First ACM Workshop on Role-based Access Control.*

Assoc Prof. Zailani Mohamed Sidek. (2003). Universiti Teknologi Malaysia. *The Development of a Commercially Viable Database Encryption Tool for Oracle8i RDBMS*. PhD Thesis in Computer Science.

Barry W. Boehmn. (1996). IEEE Software. *Anchoring the Software Process*. July 4, 13. 73-82.

Barry W. Boehmn. (1988). TRW Defense Systems Group. *A Spiral Model of Software Development and Enhancement*. May 5. 61-72

Bertino, E., Jajodia, S., and Samarati, P. (1995). *Database Security: Research and Practice*. Information Systems. 20(7): 537-56.

Blakley, B. (1996). The Emporer's Old Armor. *ACM New Security Paradigm Workshop*. Lake Arrowhead, CA, 2-1.

Booch, G., Jacobson, I., and Rumbaugh, J. (1999). *The Unified Software Development Process.* Addison-Wesley.

Booch, G., Rumbaugh, J., and Jacobson, I. (1998). *The Unified Modeling Language User Guide*. Addison-Wesley.

Castano, S., Fugini, M. G., Martella, G., and Samarati, P. (1995). *Database Security*. Wokingham, England: Addison-Wesley Publishing Company.

Ciechanowicz, C. (2001). *Database Security.* Royal Holloway University of London: Lecture Notes. Not Published.

Dastjerdi, A. B., Pieprzyk, J., and Naini, R. S. (1996). *Security in Database: A Survey Study*. Communications of the ACM. 44(2): 38-44.

Denning, D. E. (1983). Field Encryption and Authentication in Advances in Cryptology. *Proceedings of CRYPTO 83 (D. Chaum, ed.), (Santa Barbara, CA)*. Plenum Press, New York. 231-247.

Humphrey, Watt (1990). *Managing The Software Process*. Addison-Wesley.

Jacobson, J., Griss, M., and Jonsson, P. (1997). *Software Reuse – Architecture, Process and Organization for Business Success*. Harlow, England: Addison Wesley Longman.

Jacobson, I., Christerson, M., Jonsson, P., and Overgaard, G. (1992). *Object-Oriented Software Engineering - A Use Case Driven Approach.* Wokingham, England: Addison-Wesley. 582.

Kruchten, P. (2000). *The Rational Unified Process—An Introduction*. 2nd ed. Addison Wesley Longman.

Kruchten, P. (1995). IEEE Software. *The 4+1 View Model of Architecture*. 12 (6): November. IEEE, 42-50.

Menezes, A. J., Oorschot, P. C. V., and Vanstone, S. A. (1997). *Handbook of Applied Cryptograph.* Boca Raton, U.S.A.: CRC Press.

Michael, A., Michael J. C., and Abramson, I. (1999). *Oracle 8i, A Beginner's Guide*. Osborne McGraw-Hill.

Pfleeger, S. L. (2000). *Software Engineering Theory and Practice*. 2nd ed. Prentice Hall.

Pernul, G. (1994). *Information Systems Security: Scope, State-of-the-art, and Evaluation of Techniques*. Int. Journal of Information Management, Butterworth-Heinemann. 15(3): 105-121.

Rumbaugh, J.; Booch, G.; Jacobson, I. (1999). *UML Reference Manual*. Addison Wesley Longman.

Roger, S. P. (2001). *Software Engineering A Practitioner's Approach*. 5th ed. McGraw-Hill International Edition.

Royce, W. (1998). *Software Project Management: A Unified Framework*. Addison Wesley Longman.

Sandhu, R. and Jajodia, S. (1990). Integrity mechanisms in database management systems. *Proc. 13th National Computer Security Conference,* 526-540.

Sandhu, R. and Samarati, P. (1996). Authentication, Access Control and Audit. *ACM Computing Surveys.* 28(1): 241-243.

Schneier, B. (1998). IEEE Computer. *Cryptographic Design Vulnerabilities*. 29-33.

Seberry, J. and Pieprzyk, J. (1989). *Cryptography: An Introduction to Computer Security.* Victoria, Australia: Prentice Hall of Australia Pty. Ltd.

Smith, G. W. (1989). Multilevel Secure Database Design: A Practical Application. *Proc. 5th IEEE Annual Computer Security Application Conference*. IEEE

Computer Society Press. 314-321.

Theriault, M. and Heney, W. (1998). *Oracle Security. 1st. ed.* Sebastopal, CA: O'Reilly & Associates.

Tuan Sabri bin Tuan Mat @ Tuan Muhammad. (2000). *Design of New Block and Stream Cipher Encryption Algorithms for Data Security.* Universiti Teknologi Malaysia: Ph.D. Thesis.

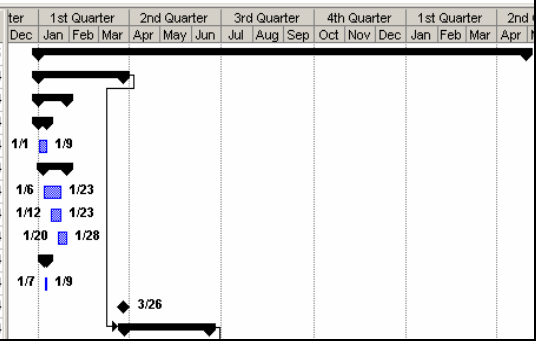
Universiti Teknologi Malaysia. (2003). *Panduan Menulis Tesis*. Universiti Teknologi Malaysia.

APPENDICES

APPENDIX A

OraCrypt Development Plan

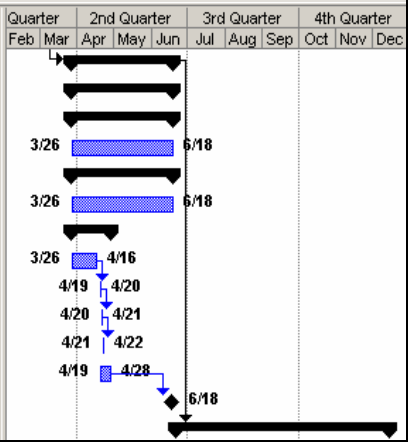
Task Name	Duration	Start	Finish
OraCrypt Development Plan	**261.25 days**	**Thu 1/1/04**	**Sat 4/30/05**
Inception Phase	**45.75 days**	**Thu 1/1/04**	**Fri 3/26/04**
Iteration I1 - Preliminary Iteration	**15 days**	**Thu 1/1/04**	**Wed 1/28/04**
Project Management	**5 days**	**Thu 1/1/04**	**Fri 1/9/04**
Develop Software Development Plan	5 days	Thu 1/1/04	Fri 1/9/04
Requirement Management	**12.25 days**	**Tue 1/6/04**	**Wed 1/28/04**
Analyze the Problem	10 days	Tue 1/6/04	Fri 1/23/04
Define the System	7 days	Mon 1/12/04	Fri 1/23/04
Manage Scope of the System	5 days	Tue 1/20/04	Wed 1/28/04
Environment	**2 days**	**Wed 1/7/04**	**Fri 1/9/04**
Prepare Environment of the Project	2 days	Wed 1/7/04	Fri 1/9/04
Business Case Review Milestone	0 days	Fri 3/26/04	Fri 3/26/04
Elaboration Phase	**45.75 days**	**Fri 3/26/04**	**Fri 6/18/04**



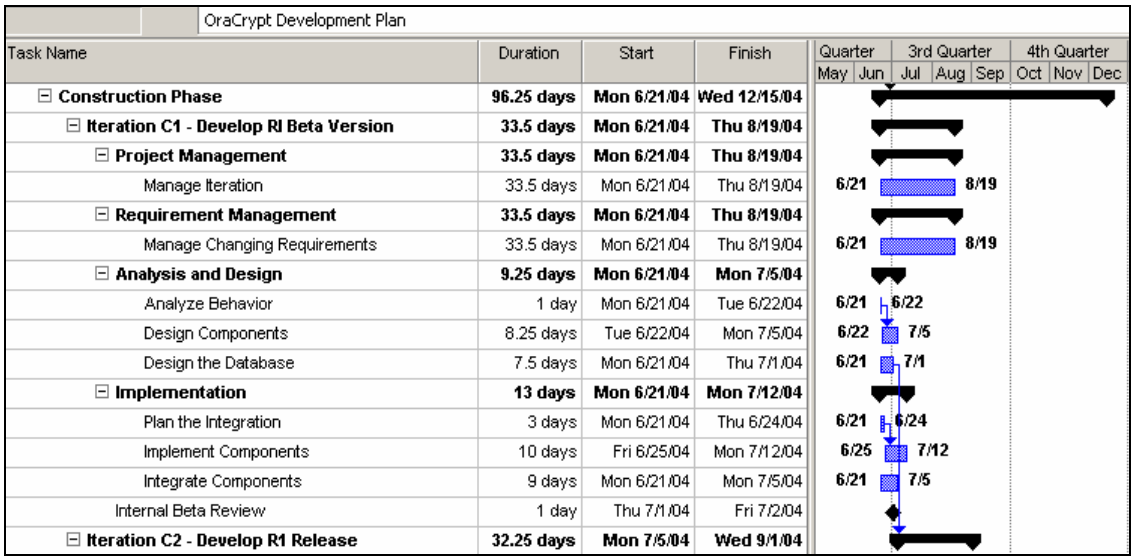
Appendix A-1 Project Schedule for Inception Phase

OraCrypt Development Plan

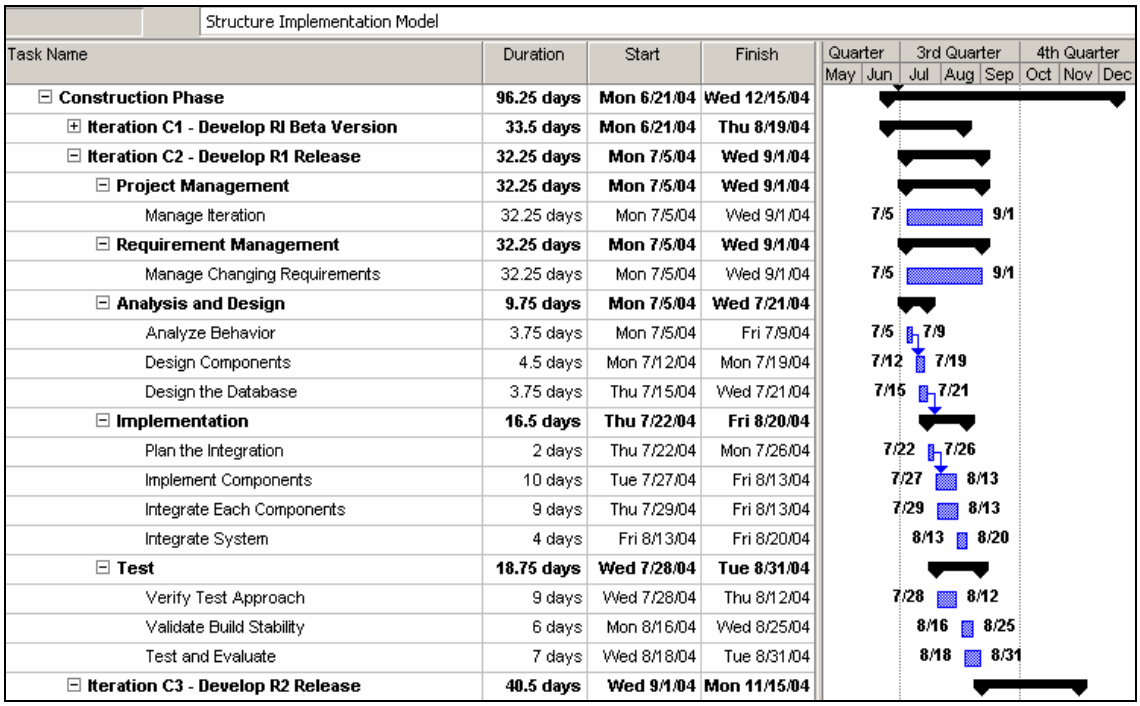
Task Name	Duration	Start	Finish
Elaboration Phase	**45.75 days**	**Fri 3/26/04**	**Fri 6/18/04**
Iteration E1 - Develop Architectural Prototype	**45.75 days**	**Fri 3/26/04**	**Fri 6/18/04**
Project Management	**45.75 days**	**Fri 3/26/04**	**Fri 6/18/04**
Manage Iteration	45.75 days	Fri 3/26/04	Fri 6/18/04
Requirement Management	**45.75 days**	**Fri 3/26/04**	**Fri 6/18/04**
Manage Changing Requirements	45.75 days	Fri 3/26/04	Fri 6/18/04
Analysis and Design	**18 days**	**Fri 3/26/04**	**Wed 4/28/04**
Define Candidate Architecture	12 days	Fri 3/26/04	Fri 4/16/04
Analyze Behavior	1 day	Mon 4/19/04	Tue 4/20/04
Design Components	1 day	Tue 4/20/04	Wed 4/21/04
Design the Database	1 day	Wed 4/21/04	Thu 4/22/04
Structure Implementation Model	6 days	Mon 4/19/04	Wed 4/28/04
Lifecycle Architecture (LCA) Milestone	0 days	Fri 6/18/04	Fri 6/18/04
Construction Phase	**96.25 days**	**Mon 6/21/04**	**Wed 12/15/04**



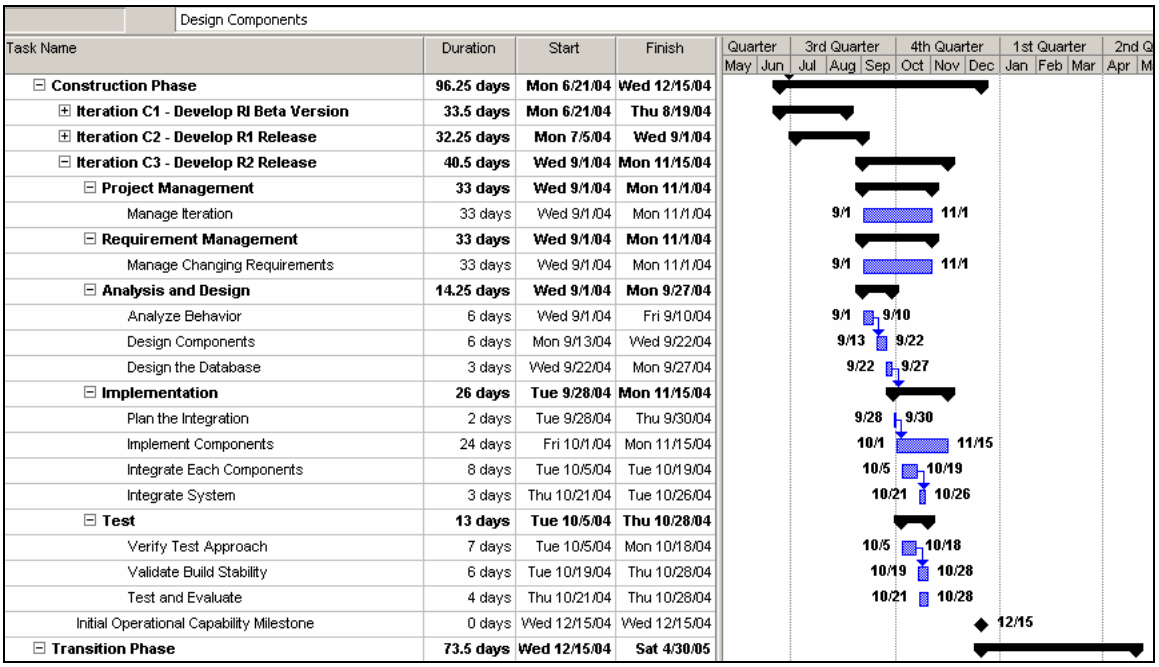
Appendix A-2 Project Schedule for Elaboration Phase



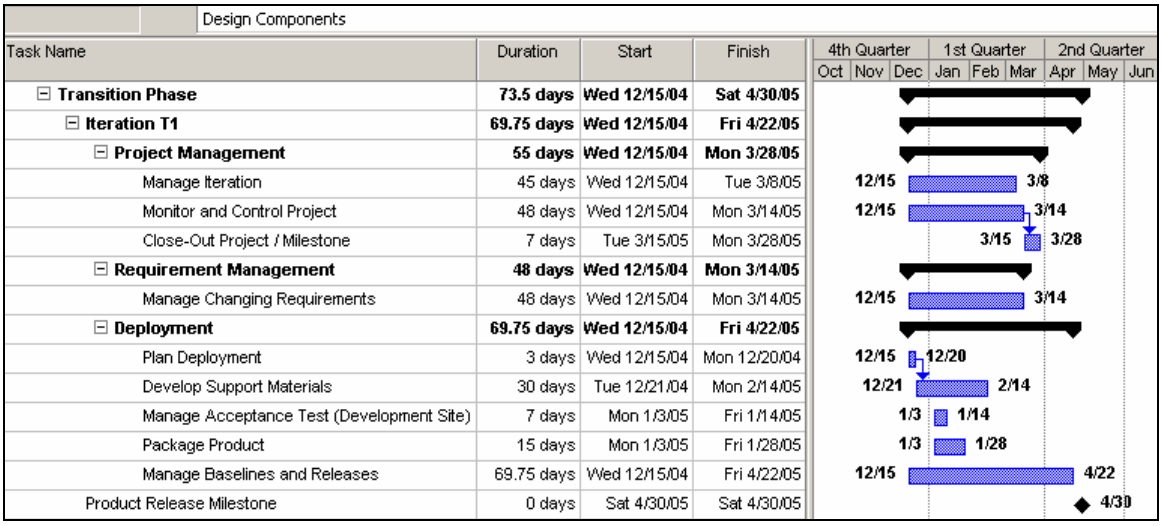
Appendix A-3 Project Schedule for Construction Phase – Iteration C1



Appendix A-4 Project Schedule for Construction Phase – Iteration C2

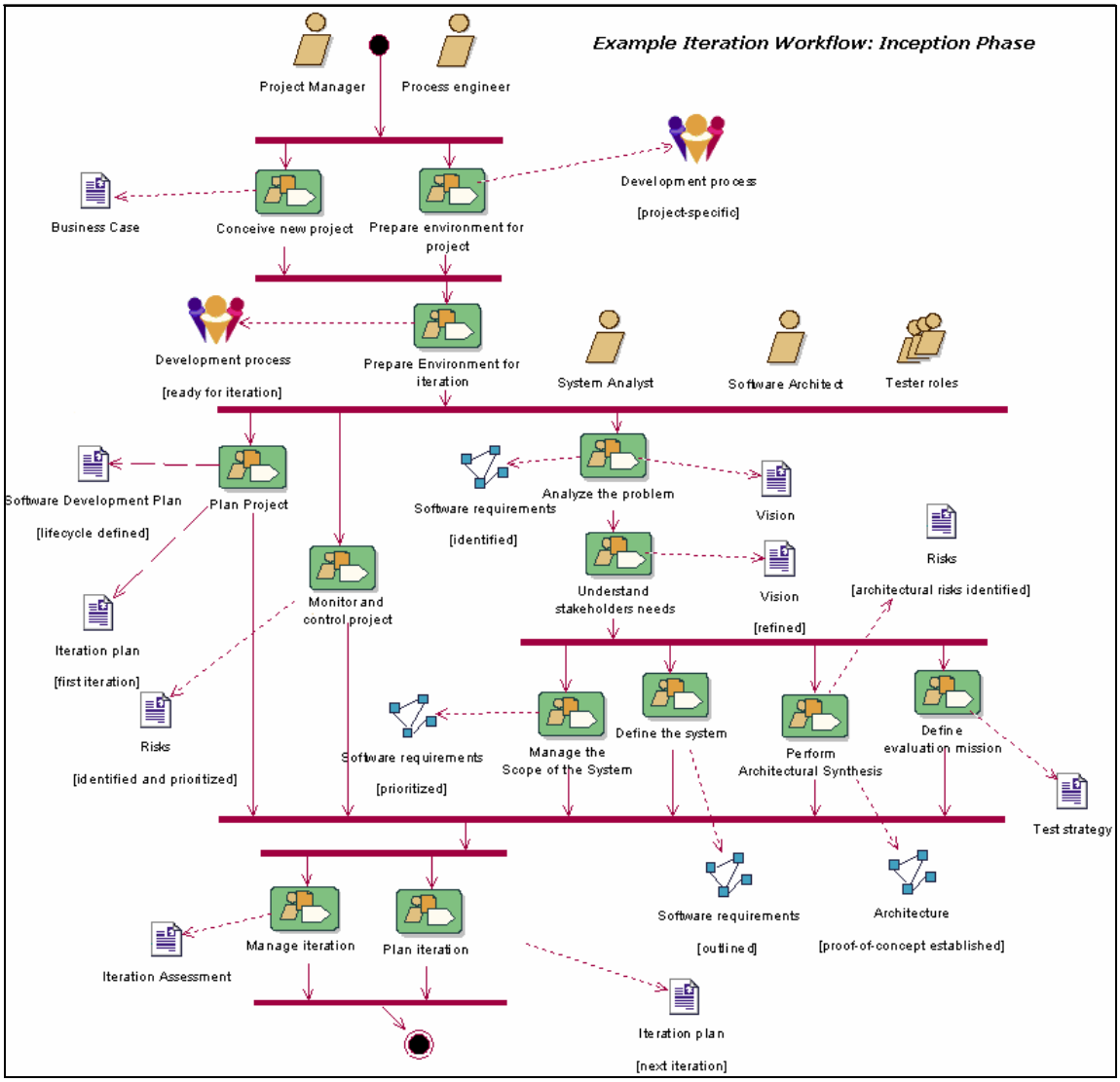


Appendix A-5 Project Schedule for Construction Phase – Iteration C3

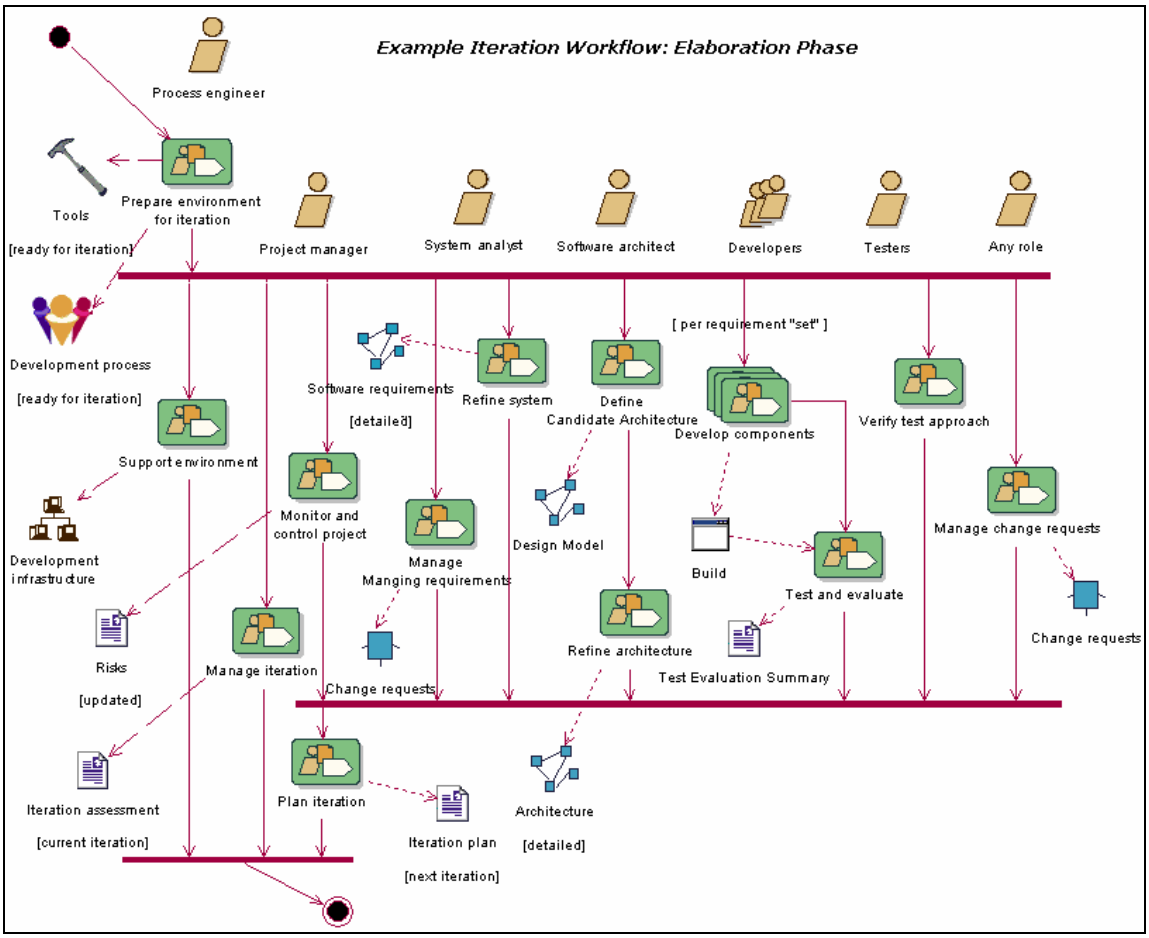


Appendix A-6 Project Schedule for Transition Phase

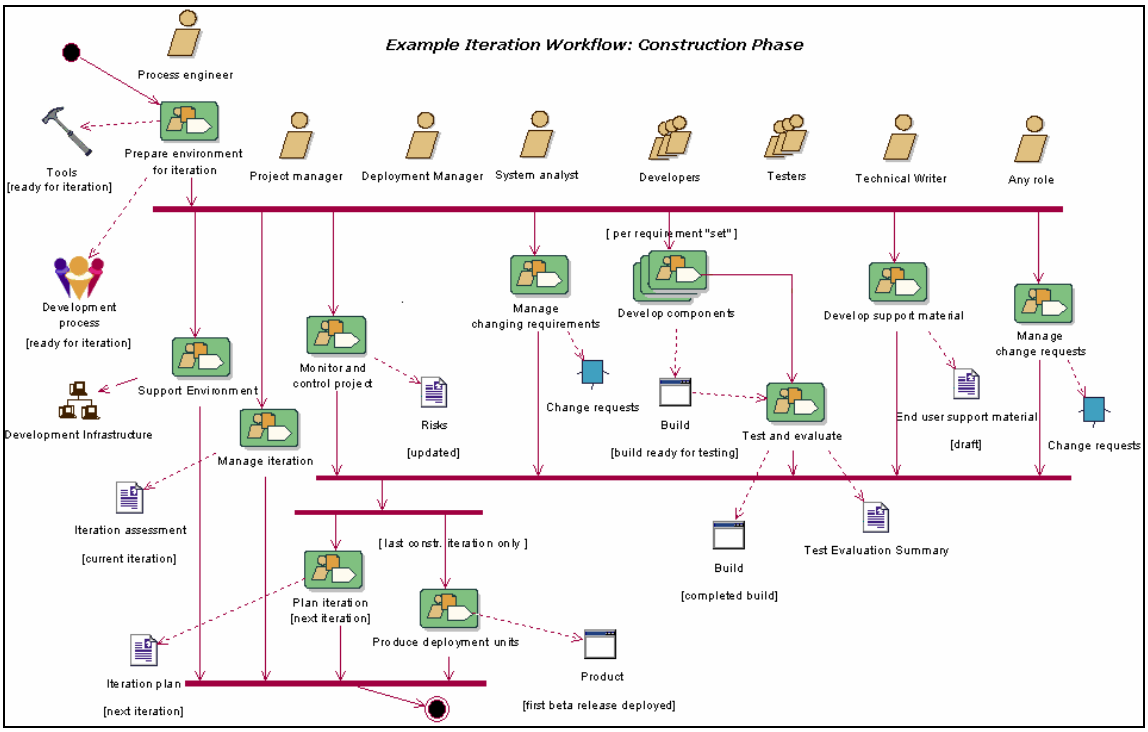
APPENDIX B



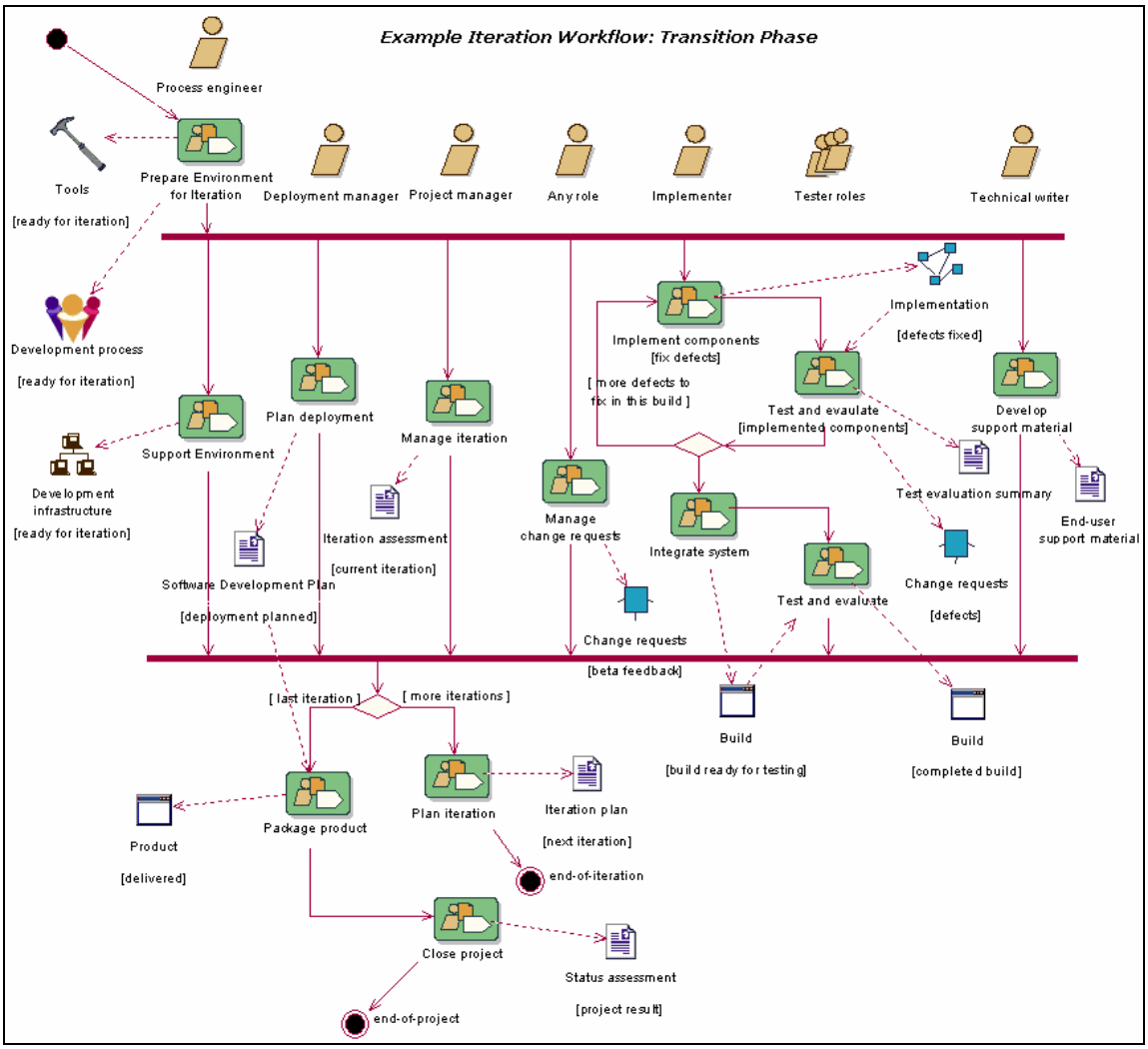
Appendix B-1 Iteration Workflow In Inception Phase



Appendix B-2 Iteration Workflow In Elaboration Phase

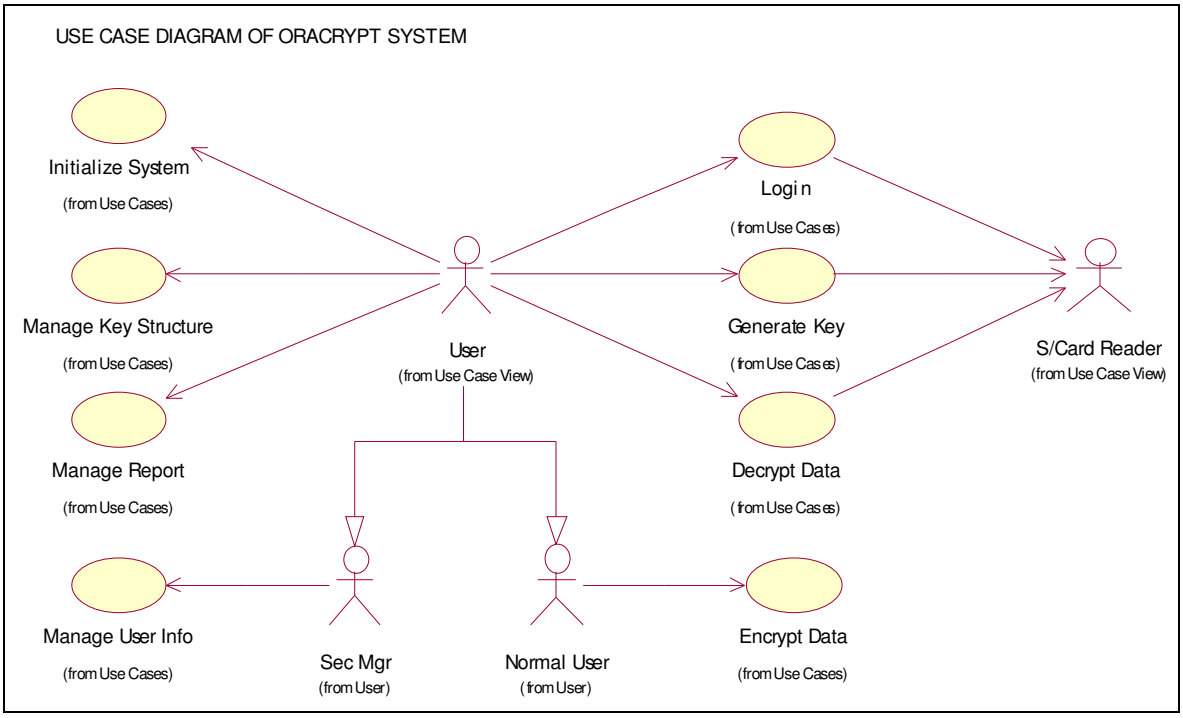


Appendix B-3 Iteration Workflow In Construction Phase



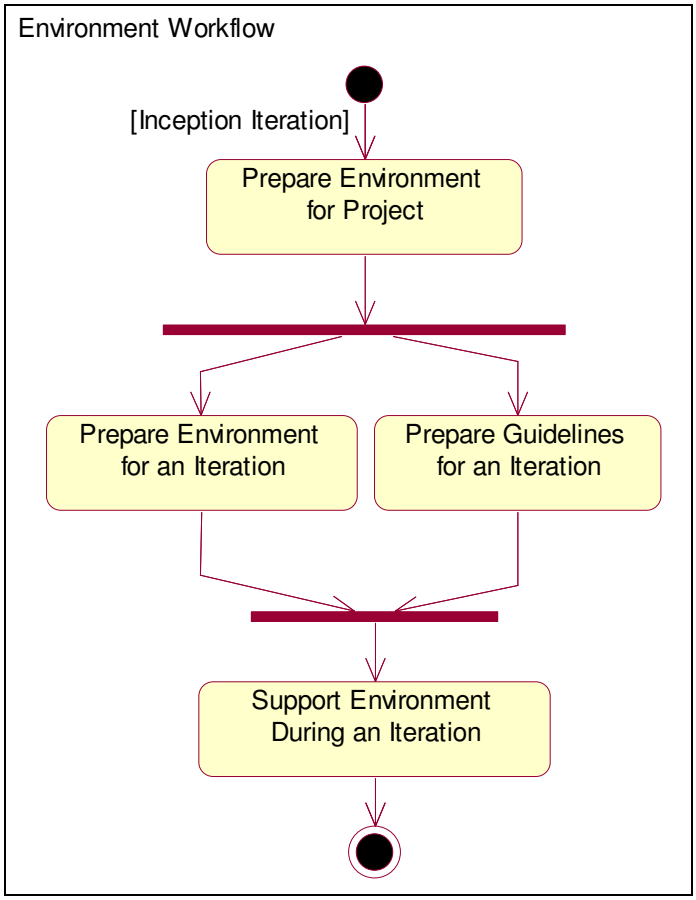
Appendix B-4 Iteration Workflow In Transition Phase

APPENDIX C

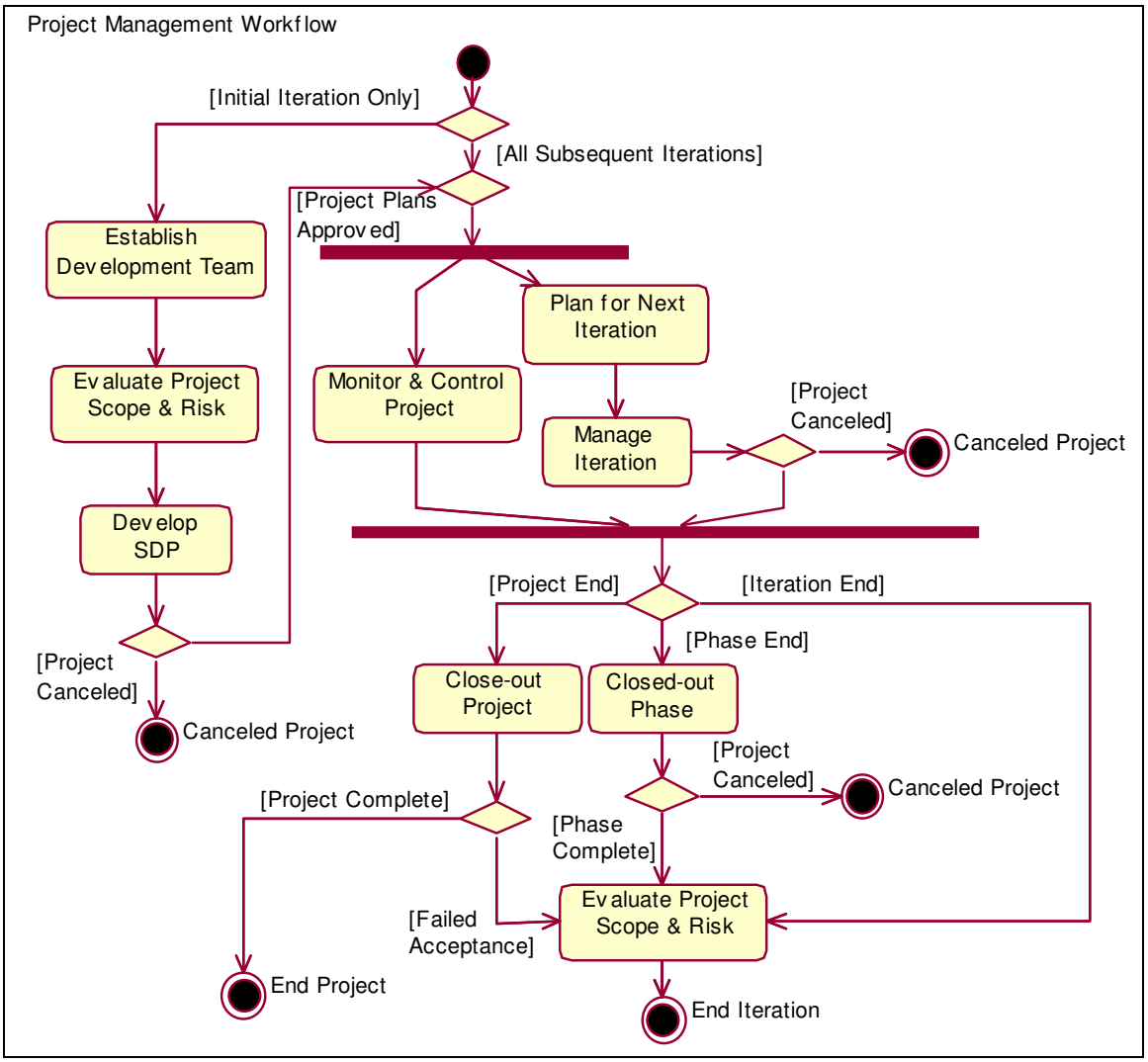


Appendix C Use Case Diagram of OraCrypt™ product

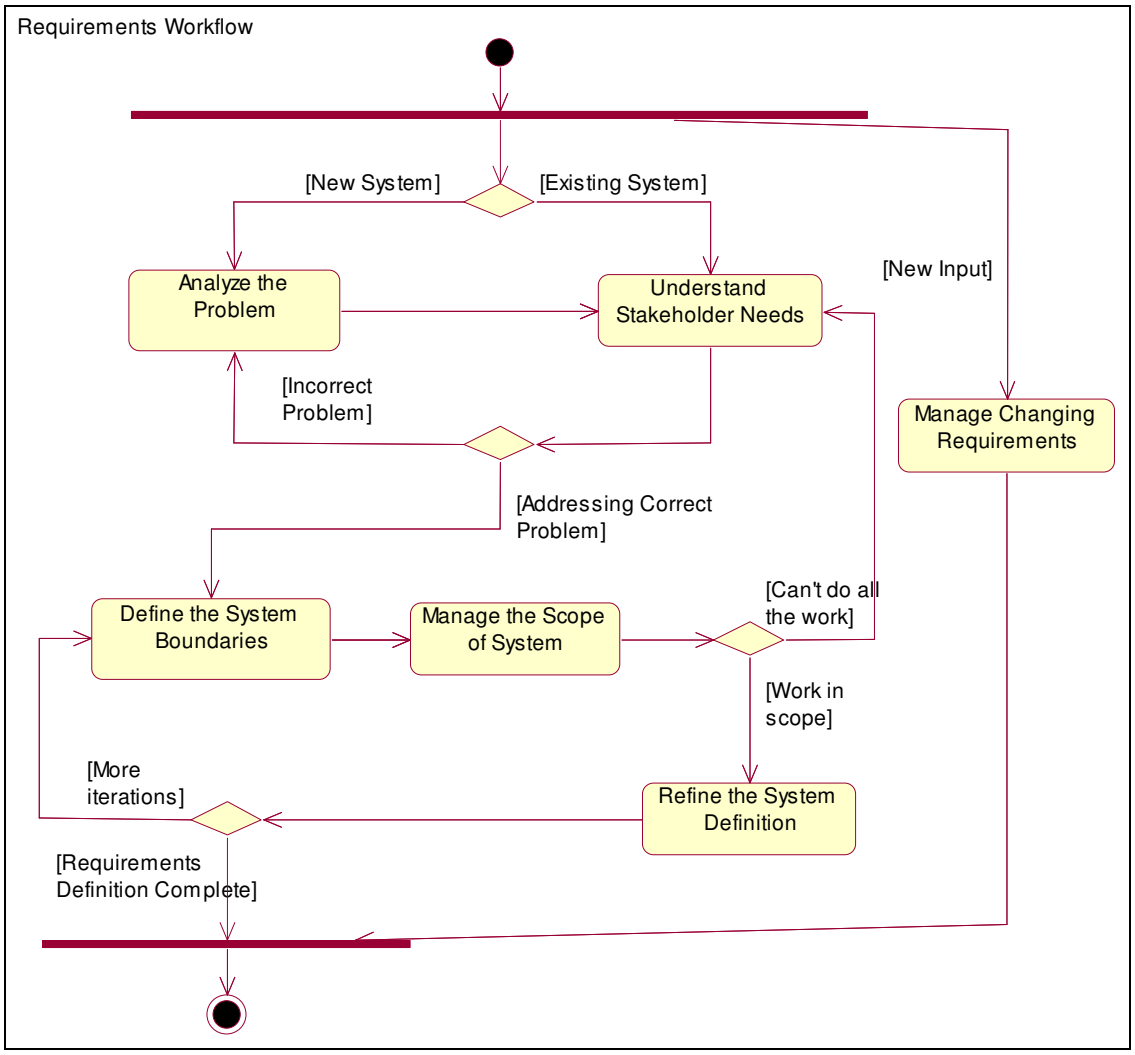
APPENDIX D



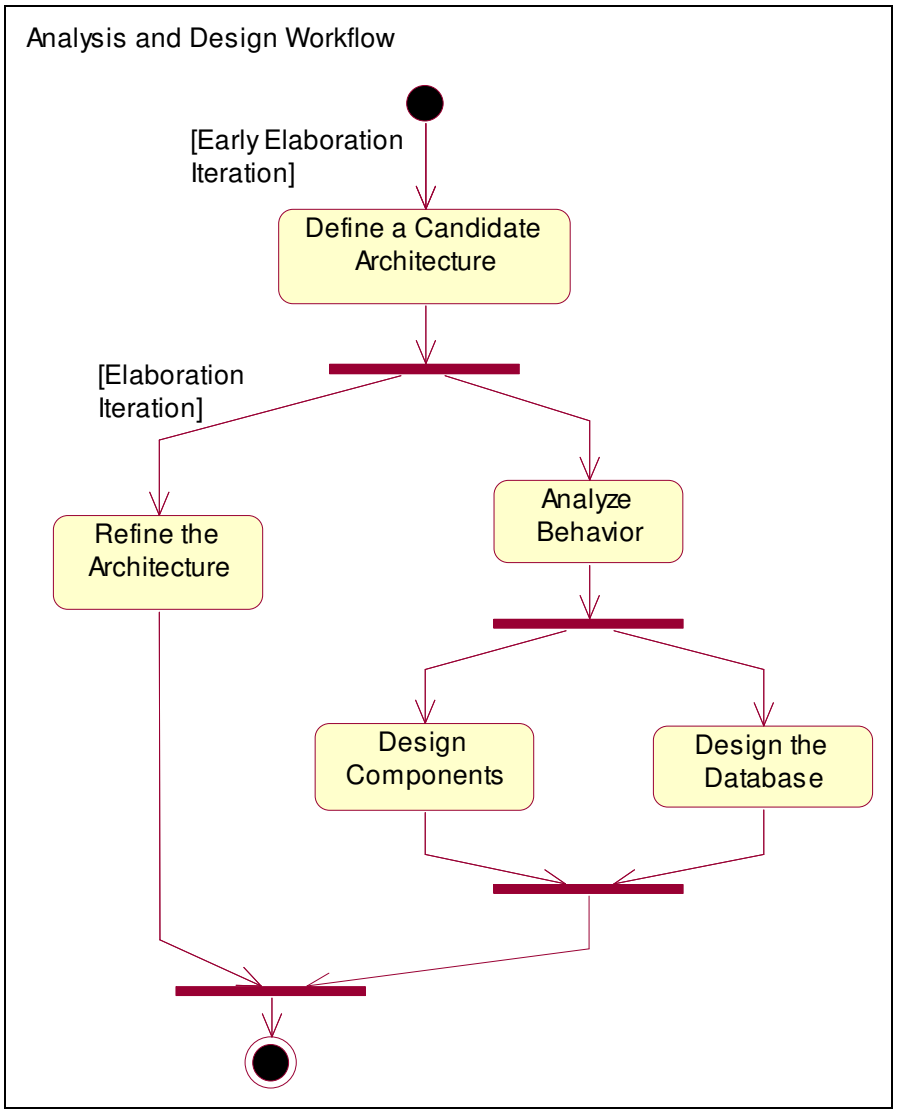
Appendix D-1 Environment Workflow



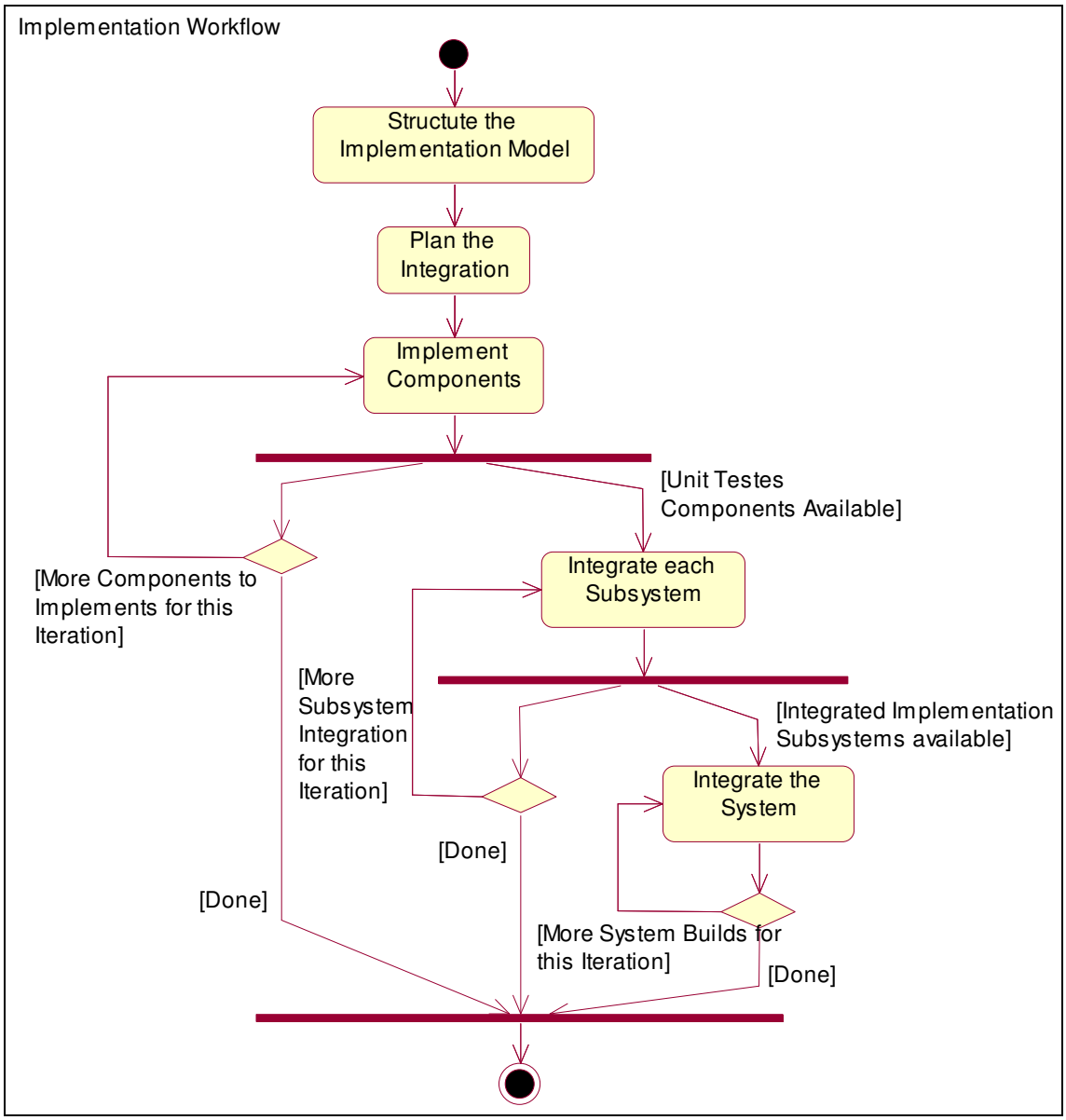
Appendix D-2 Project Management Workflow



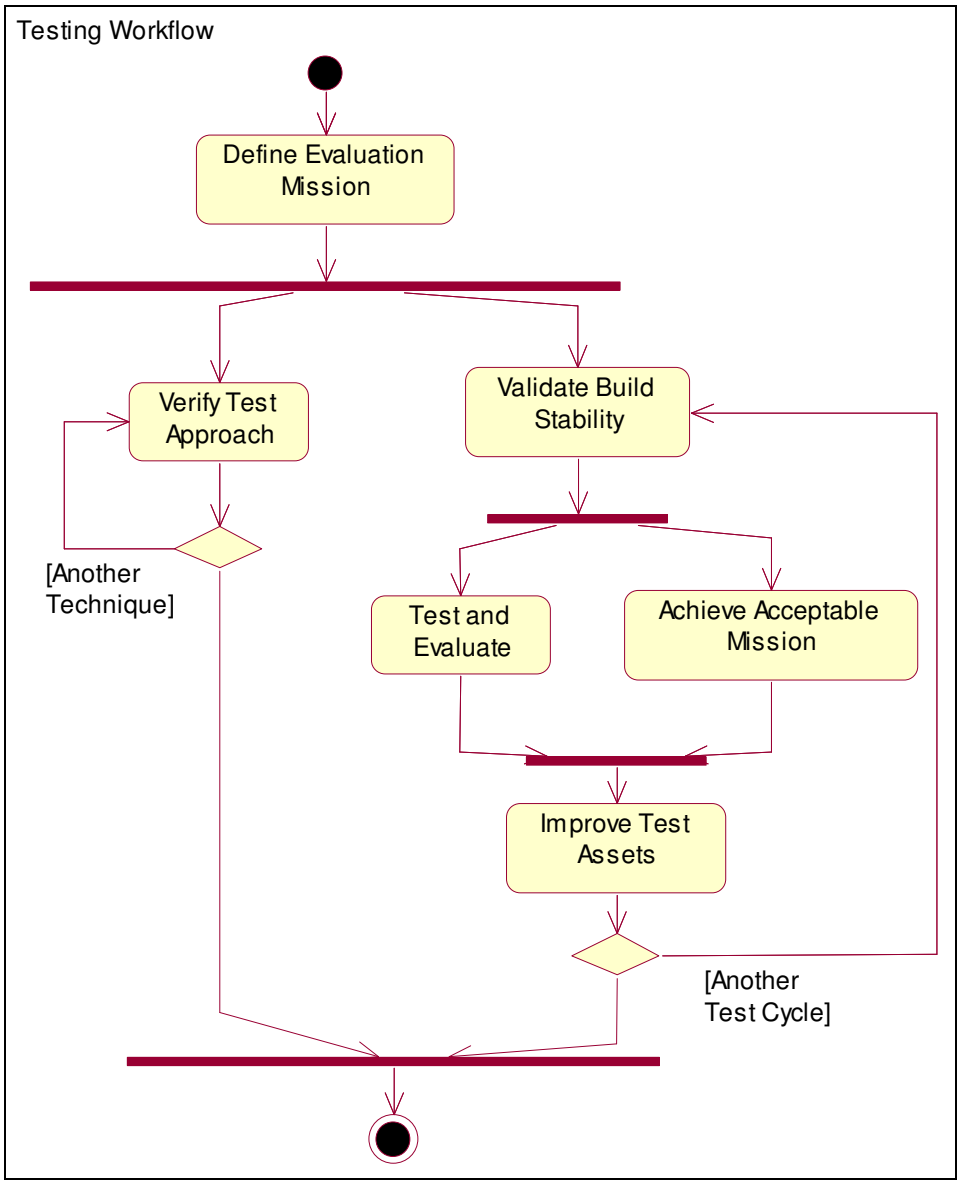
Appendix D-3 Requirements Workflow



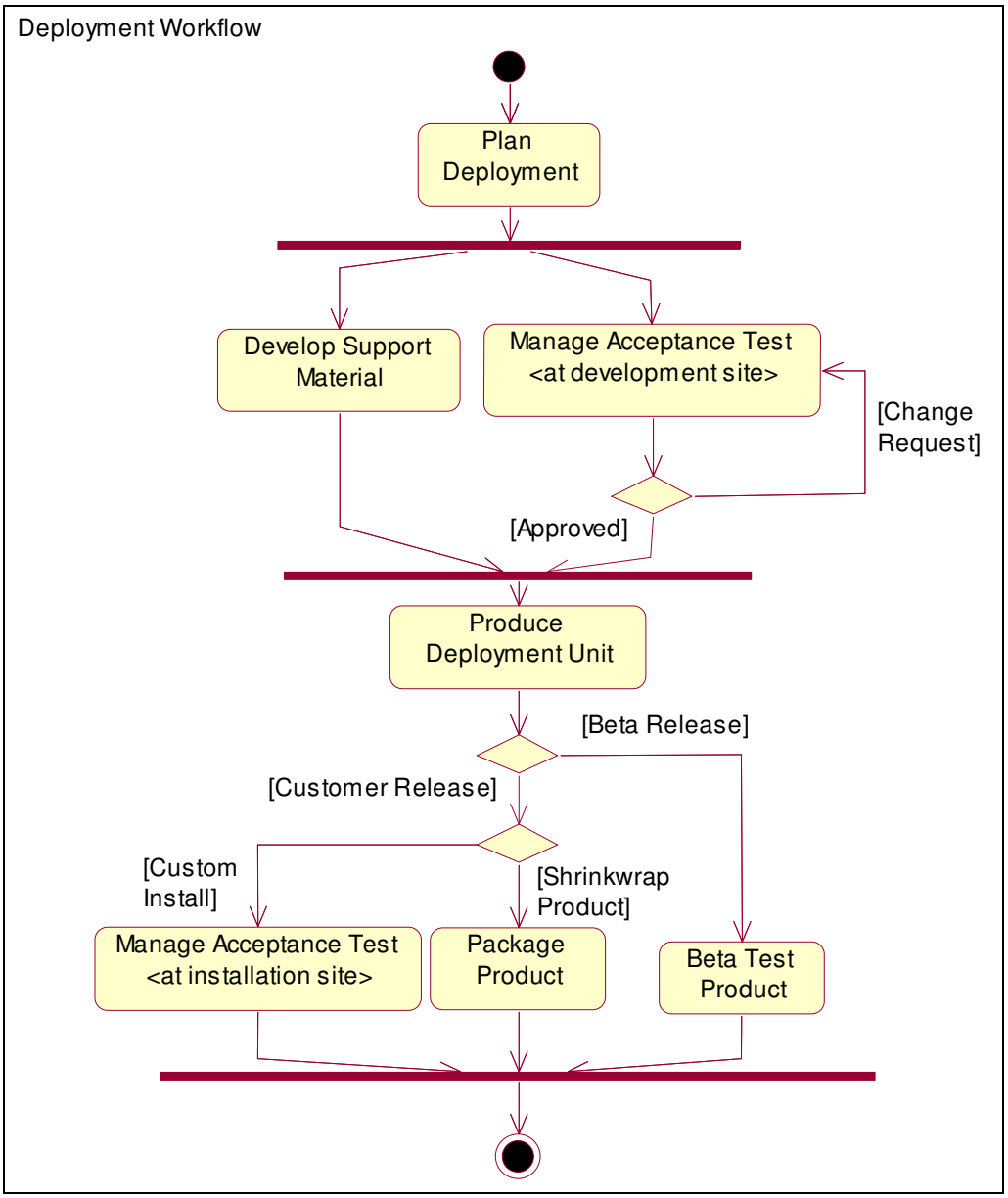
Appendix D-4 Analysis and Design Workflow



Appendix D-5 Implementation Workflow



Appendix D-6 Testing Workflow



Appendix D-7 Deployment Workflow

ABSTRACT

In database security, access control is a major research issue. Discretionary access controls have been handled well by many database management systems through user roles and privileges. Mandatory access controls, on the other hand, remains a big problem when users with lower security clearance accessing data of higher security class. Data with classifications and users have clearances developed multilevel access controls, thus the problem of multilevel security. Many researches have been conducted using methods like object labeling, trusted systems, security filters, database views and etc. Many a times the problem remains unsolved due to either too theoretical or not practical to be implemented. Recent developments in research showed cryptography to be the promising solution to the multilevel security problem. With appropriate key management and good multilevel security scheme design, the problem can be solved in both theory and implemented in practice. This research endeavor is one such effort. It presents an investigation into the applications of modern cryptography for the security of databases. The investigation yields a new multilevel security scheme based on indigenous cryptographic primitives and supported by a new key management technique. The cryptographic primitives include enhanced block cipher and a new stream cipher design successfully implemented in a commercial database. The system yields a new approach in accessing and processing encrypted data using Initialization Vectors and provides solutions for hierarchical and direct access controls. The novel scheme allows the encryption of data at the tuple, attribute, and data element levels of a relation. The security of the scheme is guaranteed with no keys present in the system but stored securely in smartcards. The outcome from this research is realized in OraCrypt application which is implemented by usign Oracle 8i RDBMS.

Key researchers:

Mr. Mohd Nazri Kama

Assc. Prof. Dr. Zailani Mohamed Sidek

E-mail : nazrikama@citycampus.utm.my

Tel No : 03-26154344

Vot No : 74228

ABSTRAK

Dalam keselamatan pangkalan data, kawalan capaian merupakan satu isu penyelidikan yang besar. Kawalan capaian "budi bicara" telah ditangani dengan baik oleh sistem pengurusan pangkalan data menggunakan peranan dan hak istimewa pengguna. Kawalan capaian mandatori pula masih memberi masalah besar apabila pengguna yang mempunyai kebenaran keselamatan lebih rendah cuba mencapai data yang mempunyai kelas keselamtan lebih tinggi. Data mempunyai kelas dan pengguna mempunyai kebenaran tertentu menyebabkan kawalan capaian menjadi berbilang aras dan masalah keselamatan berbilang aras. Banyak penyelidikan dijalankan untuk menyelesaikan masalah seperti menggunakan kaedah melabel objek, sistem beramanah, penapis keselamatan, pandangan pangkalan data, dan lain-lain. Dalam banyak keadaan, masalah ini tidak dapat diselesaikan samada kerana terlalu teori atau tidak praktik untuk dilaksanakan. Perkembangan terkini penyelidikan menunjukkan kriptografi adalah satu penyelesaian yang baik kepada masalah ini. Dengan sistem pengurusan kunci yang sesuai dan skim keselamatan berbilang aras yang baik, masalah ini dapat diselesaikan secara teori dan praktik. Penyelidikan ini merupakan satu usaha ke arah ini menggunakan kriptografi bagi menyelesaikan masalah keselamatan dalam pangkalan data. Penyiasatan ini menghasilkan satu skim keselamatan berbilang aras yang baru menggunakan primitif kriptografi asli dan disokong oleh satu teknik pengurusan kunci baru. Primitif kriptografi ini termasuk sifer blok dipertingkatkan dan satu rekabentuk baru strim sifer yang telah dilaksanakan dengan jayanya dalam satu pangkalan data perdagangan. Sistem memberi satu pendekatan pencapaian dan pemprosesan data tersulit menggunakan Vektor Peng-awalan data. Ia mampu memberi penyelesaian kawalan capaian data secara terus dan berhirarki pada tiga peringkat iaitu baris, lajur, dan unsur data. Keselamatan skim ini dijamin kerana kunci disimpan dalam kad pintar.

Hasil kajian penyelidikan ini direalisasikan di dalam satu aplikasi yang iaitu OraCrypt yang menggunakan pangkalan data Oracle8i RDBMS.

Penyelidik Utama:

Mr. Mohd Nazri Kama

Assc. Prof. Dr. Zailani Mohamed Sidek

E-mail : nazrikama@citycampus.utm.my

Tel No : 03-26154344

Vot No : 74228

TABLE OF CONTENTS

LIST OF TABLES

TABLE NO TITLE PAGE

LIST OF FIGURES

FIGURE NO TITLE PAGE

LIST OF APPENDICES

APPENDIX NO TITLE PAGE

LIST OF ABBREVIATIONS

CASE	-	Center for Advanced Software Engineering
DBMS	-	Database Management Systems
DL	-	Digital Library
DLL	-	Dynamic Link Library
GUI	-	Graphical User Interface
HLIV	-	Hashed Left Data IV
HRIV	-	Hashed Right Data IV
HUIV	-	Hashed User IV
IRPA	-	Intensity Research and Priority Area
ITK	-	Institut Teknologi Kebangsaan
IV	-	Initialization Vector
I/O	-	Input/Output
LIV and RIV	-	Left and Right IVs
MINDEF	-	Ministry of Defense
OO	-	Object-Oriented
OS	-	Operating System
RDBMS	-	Relational Database Management System
RUP	-	Rational Unified Process
SAD	-	Software Architecture Document
SDK	-	Software Development Kit
SDP	-	Software Development Plan
SRS	-	Software Requirements Specification
TS	-	Tuan Sabri

TSBC	-	TS Block Cipher
TSSC	-	TS Stream Cipher
UML	-	Unified Modeling Language
UTM	-	University of Technology Malaysia

LIST OF TERMINOLOGY

Actor	Someone or something, outside the system that interacts with the system.
Authentication	The process of verifying the identity of a user, device, or other entity in a computer system, often as a prerequisite to allowing access to resources in a system.
Authorization	Permission given to a user, program, or process to access an object or set of objects. In Oracle, authorization is done through the role mechanism. A single person or a group of people can be granted a role or a group of roles. A role, in turn, can be granted other roles.
Availability (of Systems)	Preventing, detecting and/or deterring improper denial of access to services provided by the system.
Beta Testing	Pre-release testing in which a sampling of the intended customer base tries out the product.
Codes	A system for hiding the meaning of a message by replacing each word or phrase in the original message with another character or set of characters.
Computer Security	Protection of information processed by a computer against unauthorized observation, unauthorized or improper modification, and denial of service (ensuring no authorized use of the information is denied).
Construction Phase	The third phase of the Unified Process, in which the software is brought from an executable architectural baseline to the point at which it is ready to be transitioned to the user community.
Cryptography	Mathematical manipulation of data, which the purposes are for reversible or irreversible transformation. The act of writing and deciphering secret code resulting in secure messages.

Data Encryption	The encoding of data so that it cannot be easily deciphered without the use of specialized decryption mechanisms or procedures.
Database	A collection of data, arranged in some meaningful way.
Database Security (DB Security)	A set of measures, policies and mechanisms to provide secrecy, integrity and availability of data and to combat possible attacks on the system (threats) from insiders and outsiders, both malicious and accidental.
Database Management System (DBMS)	A software/program that organizes and operates on the database and auxiliary control information to implement the decisions of the access policy and gives users the means to retrieve information.
Decryption	The process of converting the contents of an encrypted message (ciphertext) back into its original readable format (plaintext).
Dynamic Link Library	A file containing executable code and data bound to a program at run time rather than at link time. The C++ Access Builder generates beans and C++ wrappers that let your Java programs access C++ DLLs.
Elaboration Phase	The second phase of the process where the product vision and its architecture are defined.
Encryption	The process of disguising a message rendering it unreadable to anybody but the intended recipient.
Inception Phase	The first phase of the Unified Process, in which the seed idea, request for proposal, for the previous generation is brought to the point of being (at least internally) funded to enter the elaboration phase.
Phase	The time between two major project milestones, during which a well-defined set of objectives is met, artifacts are completed, and decisions are made to move or not move into the next phase.
Security Threats	A hostile agent that, either casually or by using a specialized technique, can disclose or modify the information managed by a security system.

Smartcard	A plastic card (like a credit card) with an embedded integrated circuit for storing information, including such information as user names and passwords. A smartcard is read by a hardware device at any client or server.
Transition Phase	The fourth phase of the process in which the software is turned over to the user community. A relationship between two states indicating that an object in the first state will perform certain specified actions and enters the second state when a specified event occurs and specified conditions are satisfied. On such a change of state, the transition is said to execute.
TS Block Cipher	Block cipher can process data in blocks of 16 characters at a time.
TS Stream Cipher	Stream cipher can process 255 characters of data at a time.
Unified Modeling Language	A language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system.
User Authentication	The authorization mechanism that uniquely identify the database users before allowing them to access the data.

CHAPTER 1

INTRODUCTION

1.1 Introduction

This chapter describes the introduction of the project in which the researcher has been involved in order to fulfill the requirements for the new technology found for the Research Management Centre (RMC). The project has been operated in eighteen (18) months duration time starting from 14th December 2003 until 14th June 2005.

1.2 Company Background

This section describes the background of the institution that the researcher has involved during the project research and development.

1.2.1 Universiti Teknologi Malaysia

Universiti Teknologi Malaysia (UTM) was established on 14th March 1972 under the name Institut Teknologi Kebangsaan (ITK). On 1st April 1975, this technical school had undergone many reformations and officially known as UTM. UTM is a prestigious university at the frontier of technology with a vision to lead in the development of

creative human resource and technology in line with the aspirations of the nation. It is one of Malaysia's leading universities for engineering and technology, which has:

i. A reputation for innovative education and leading-edge research, educating the technologies and professional
ii. More than 25,000 students at campus in Johor, more than 4,500 students in Kuala Lumpur campus, and more than 5,000 students on distance learning/part-time programmes
iii. More then 2,500 postgraduates students in various fields of specialization
iv. More than 20 specialist institutes and centre, in addition to academic departments to service the technology, education and research needs.

UTM strives for academic excellence through creative learning and state-of-the-art technology. UTM has 2 campuses, the 1,222-hectare main campus in Skudai, Johor and 18-hectare branch campus situated at Jalan Semarak, Kuala Lumpur.

1.2.2 Centre For Advanced Software Engineering (CASE)

The Centre for Advanced Software Engineering (CASE) was established in 1996 as a joint-venture programme between Universiti Teknologi Malaysia (UTM) and Universite Thales, France for enabling the technology transfer into Malaysian industry, especially the software industry. CASE is committed in providing opportunities for advanced studies and professional development for the current and future needs of technology based industries. At CASE, the aim is to create opportunities for local industries or individuals to be trained in important aspects of Information Security and Software Development via the following programmes:

i. Masters and Doctor of Philosophy in Information Security
ii. Master and Doctor of Philosophy in Software Engineering
iii. Continuing Education/Short Courses

In meeting these goals, CASE promotes its expertise and technical capabilities in its effort to become a centre of excellent, majoring in the following specific areas:

i. Research and Development
ii. Software Consultancy and Mentoring
iii. Partnership in Software Development Software engineering project and software methodology
iv. Software engineering techniques and tools

1.2.3 Project Team

OraCrypt™ project team consists of four (4) staff working closely with each other in research and development activities at CASE. The project was funded by Intensification of Research Priority Areas (IRPA) and managed by the Research Management Centre (RMC). RMC is responsible to manage the research and development activities conducted by UTM researchers that sometimes collaborate with the industries outside the UTM. It also regulates the grant for the research project conducted by UTM researchers.

The project team consists of one (1) Project Advisor, one (1) Project Leader, and two (2) System Analysts. *Figure 1.1* illustrates the project organization.

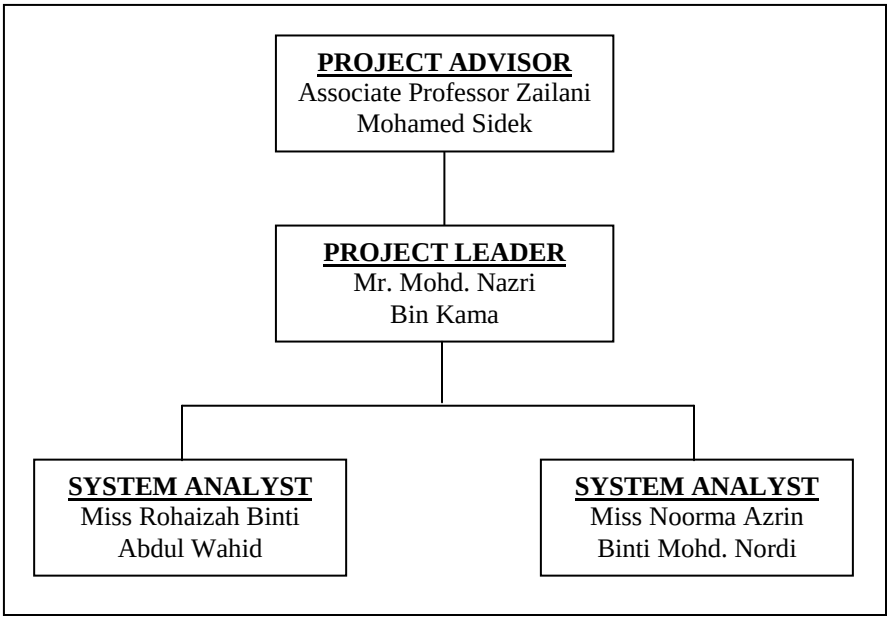


Figure 1.1 OraCrypt^{TM} Project Organizations

Table 1.1 describes the role represented in the project organization diagram above and their primary responsibilities.

Table 1.1 Lists Of Roles Respective Responsibilities

Role	List of Responsibilities
Project Advisor	▪ Advise on the overall project contents and the products that should be delivered on a regular period of time. ▪ Motivate the team members to perform their tasks. ▪ Help the team in allocating the tasks and resolving issues.
Project Leader	▪ Plan and regulate the tasks. ▪ Approve the process development. ▪ Schedule and check the milestone for review and evaluation of the product, tasks and software process. ▪ Maintain a project plan. ▪ Lead the team in producing the development strategy. ▪ Lead the team in producing the preliminary size and time estimates for the products to be produced. ▪ Lead the team in producing the high-level design and design specification. ▪ Motivate the team members to perform their tasks.

Role	List of Responsibilities
	▪ Help the team in allocating the tasks and resolving issues.
System Analyst	▪ Define the responsibilities, operations, attributes, and relationship of one or several classes, and determine how they will be adjusted to the implementation environment. ▪ Devise a coding standard to be used during the implementation process. ▪ Implement the design defined in the SDD. ▪ Strive to produce code that is readable and bug-free. ▪ Develop and test components, in accordance with the projects adopted standards, for integration purposes. ▪ Establish and maintain the team's development standards.

1.3 Project Background

The project is being done under the Intensification of Research and Priority Area (IRPA) project at UTM. The project was initially a work of a doctorate student Associate Professor Zailani Bin Mohamed Sidek of CASE. The research conducted is mainly on database security specifically in the area of cryptography. He argued that authentication, access control, and audit together provide the foundation for information and system security. The database security mechanism should be based on cryptographic technology, which is independent, possibly developed locally, and is based on strong theoretic foundation. With this in mind, efforts are made to serve the purpose of being independent, secure and proactive. OraCrypt™ product is one of the solutions. In this project, the researcher take the approach of building a prototype with highly secure, independent, and implementable database security mechanism for a commercially, widely-used relational database management system.

1.3.1 Project Overview

OraCrypt™ is a product for the encryption and decryption of data in Oracle8i RDBMSs. The product consists of two (2) main modules that are Key Generation, Management and Distribution module, and Encryption and Decryption module. The key

generation, management and distribution module allows the generation of Master and Shared keys. Master keys are for the hierarchical access control to encrypt data while Shared keys are useful for both direct access and hierarchical access controls. The encryption and decryption module is designed with no keys (Master or Shared keys) to be found or stored in the database Management System (DBMS) concern. Instead all keys are stored on smartcard. In this module, it allows users to encrypt and decrypt data at all levels (row, column, and data element). And in each level, users may provide condition for the selective encryption and decryption of data at those three levels.

1.3.2 Project Vision

OraCrypt™ product is expected to become a commercial data encryption software package within a Relational Database Management System (RDBMS). In this project, the implementation is focused on the Oracle Database Management System version 8i only.

1.3.3 Project Objective(s)

The objective of this effort is to develop a prototype software security system that:

i. Design and implement a database security mechanism using independent and local cryptographic technology that will enhance the security of the database for Oracle8i RDBMS.
ii. Implement the new design for IV-based multilevel database encryption scheme in Oracle8i RDBMS environment.
iii. Develop and implement cryptographic key generation, management and distribution system.
iv. Implement the OraCrypt™ product evaluation and validation.

1.3.4 Project Scope(s)

The scopes of this project are as follows:

i. Identify and examine locally developed cryptographic algorithms that are available and suitable in the proposed database security application.
ii. Analyze and implement the new IV-based multilevel database for encryption and decryption.
iii. Implement the cryptographic key generation, management and distribution scheme.
iv. Analyze, modify, and implement the cryptographic algorithms in column encryption-decryption module.
v. Analyze, design and formulate working models and solution procedures for database encryption and decryption. The working models and solution procedures are expected to be better in terms of its security and performance when compared to the existing schemes available.

1.3.5 Project Schedule

This project was scheduled for eighteen (18) months. Refer to *Appendix A* for the details of the project's schedule.

1.4 Problem Background

This section looks into some key definitions on the issue of computer security and database security. The researcher describes the security goals and its design decisions/principles, the specific threats to database security, and the requirements for database security.

1.4.1 Computer Security Definition

Computer security [Schaefer (1993)] has come to a stage where its definition needs to be rethinking due to changes in the environment and the advancement of technologies. Blakley (1996) argues that the traditional model of computer security is no longer viable, and the new definitions of the security problem are needed before the industry can begin to work toward effective security in the new environment. In addition, Schaefer (1993) agrees that much of computer security has been reduced to practice but it is also important to understand that there are no general closed form solutions to computer security problems. This means there is no generally applicable technique that applies to securing arbitrary system environments.

They are few definitions of computer security for the sake of discussions. Castano, *et. al*. (1995), defines computer security as protection of information processed by a computer against unauthorized observation, unauthorized or improper modification, and denial of service vis-à-vis ensuring no authorized use of the information is denied. Gollmann (1999) provides a simpler definition of computer security that deals with the prevention and detection of unauthorized actions by users of a computer system. Schneier (1998) describes security as about risk management that detection and response are just as important as prevention, and that reducing the *window of exposure* for an enterprise is security's real purpose.

1.4.2 Security Goals

The three goals of secure computing are confidentiality, integrity, and availability [Pfleeger (2000), Gollmann (1999), Abrams *et. al*. (1995), Pernul (1994)]. Confidentiality means that assets of a computing system are accessible only by authorized parties. It is also called secrecy or privacy. Integrity means that assets can be modified only by authorized parties or only in authorized ways. Availability means

assets are accessible to authorized parties. It is sometimes known by its opposite, denial of service. These three goals can overlap, and they can even be mutually exclusive.

1.4.3 Database Security Definition

Database security comprises a set of measures, policies, and mechanisms to provide secrecy, integrity and availability of data and to combat possible attacks on the system from insiders and outsiders, both malicious and accidental [Castano *et. al*. 1995), Ciechanowicz (2001), Al-Salqan *et. al*. (1996)]. Database security encompasses physical, logical and organizational issues. Physical database security focuses on tools, devices, and hardware or software techniques able to prevent or detect unauthorized physical accesses to data storage facilities, and to provide database backup and/or recovery. Logical database security consists of control measures, models and techniques to prevent, detect, or deter unauthorized logical accesses to data. Organizational database security concentrates on management constraints, operational procedures, and supplementary controls established to provide database protection.

Secrecy has the objective of keeping information unreadable for outsiders while making it available for authorized users. It also means preventing, detecting, or deterring the improper disclosure of information and is very much relevant to protection of data in highly protected environments, such as the military and commercial environments. Related to this is privacy and confidentiality. Privacy reflects information about individuals or legal entities and is normally protected by laws and rules in many countries. Confidentiality is a service used to keep the content of information from all but those authorized to have it [Menezes *et. al*. (1997)]. Data integrity covers methods and techniques, which addresses the unauthorized alteration of data. To ensure data integrity, Smith (1989) specify that one or more fields of a DBMS table are designated as a unique, non null primary key, thereby ensuring that duplicate rows do not occur within one table. While Sandhu and Jajodia (1990) describes ensuring integrity of information as means of preventing/detecting/deterring the

improper modification of information. Availability of data means to ensure authorized users can have access to data when they require.

1.4.4 Threats To Database Security

A threat can be defined as a hostile agent who intentionally or unintentionally gains an unauthorized access to the protected database resources and able to disclose or modify the information managed by a system [Castano *et. al*. (1995)]. Dastjerdi, et.al (1996) classify threats into physical and logical threats. Physical threats are violations to databases in the form of a forced disclosure of passwords, a theft, a destruction of physical storage devices to a power failure, and any form of natural disaster. The methods and techniques for this type of database protection include restriction of physical access to database storage facilities and the database backup and recovery. Logical threats involve unauthorized logical access to information in databases via software. They may result in information disclosure, integrity violation, and inaccessibility to an authorized individual.

These threats may occur intentionally or by accident. Accidental threats include natural disasters like earthquakes, flood and fire, or by human error, and hardware and software failures. Malicious attacks are always intentionally carried out by disgruntled employees or abused of privileges by authorized users. These hostile users execute their acts through some hidden codes within some legitimate functions in order to violate the security of the system. Examples of such codes are Trojan Horses, trapdoors, and viruses.

1.4.5 Security Requirements For Database System

In the last section, the researcher looked at some possible threats to database security. With this understanding, the researcher selects or develops appropriate security policies and mechanisms for controlling or eliminating those threats or malicious attacks

on protected data. Security policies are policies, which outline an organization's position on security issues [Theriault and Heney (1998)]. The security policy is made more concrete with the mechanism necessary to implement the requirements of the policy [Jajodia (1996)]. Security mechanisms define how the security system should achieve the security goals. Following is a list of requirements for security of database systems [Pfleeger (2000), Castano *et. al*. (1995)]:

- Physical database integrity – the data in the database is immune to physical problems such as power failures and it can be reconstructed if destroyed through a catastrophe.
- Logical database integrity – the structure of the database is preserved. With logical integrity of a database, a modification to the value of one field does not affect other fields.
- Element integrity – the data contained in each element is accurate.
- Auditability – able to track who has accessed the elements in the database.
- Access control – a user is allowed to access only authorized data and different users can be restricted to different modes of access (read or write).
- User authentication – ensure that every user is positively identified, both for the audit trail and for permission to access certain data.
- Availability – users can access the database in general and all the data for which they are authorized.
- Encryption – ensures that only the intended user can decipher any data processed.

1.5 Problem Statement

According to Sandhu and Samarati (1996), authentication, access control, and audit together provide the foundation for information and system security. The basic premise is to have some form of protection for the precious computer and information resources. With the advent of the Internet, access to data and information has been greatly enhanced and this caused a major concern for safety and security of the

resources. The Internet has brought a borderless world of today and the need for adequate security mechanisms is becoming increasingly critical especially with the ever-rising rate of security breaches. Firewall seems to be a popular solution. However, firewalls are not immune to penetrations; once an outsider is successful in penetrating a system, they typically do not provide any protection to internal resources. We may only view Firewalls as our first line of defense since they do not protect against illegal actions by insiders, an organization's authorized users. Many security experts are of the opinion that most of the computer related crimes are done by insiders.

Protection of internal resources, though not a trivial task, needs to be supported by appropriate tools and techniques. Though there are many security measures, better protection of computer information can be obtained if several of these measures are applied simultaneously whenever possible. Such an approach is especially necessary when protection of databases is concerned. We have been introduced to the kinds of threats to database security and the security requirements of database systems against these threats. This research then is an attempt to contribute to this effect. The main statement of the research problem is:

> What is a "simple" cryptographic database security mechanism, with due respect to all security requirements and constraints, that is applicable and implemental in commercial database management products popularly in use today?

What constitutes a simple security mechanism, in our context, is that the database security mechanism needs to be a practical and workable solution, general in nature, and that suits variety of commercial DBMS products. The database security mechanism should be based on cryptographic technology which is independent, possibly developed locally, and is based on strong theoretic foundation.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This literature review will focus on knowledge representation, research on existing product in the market, comparison between the existing product and the process modeling standards that was used during the development of OraCrypt™ product.

2.2 Cryptographic Application In Database Security

Database management systems are a part of computer systems, as such protection of information in databases are very much dependent on the operating systems and the DBMSs. Traditionally, physical security and operating system security have been used to provide security for the databases. Physical security has the disadvantages of becoming very expensive, prevents remote access, and cannot address the requirement of a wide variety of legitimate users with disparate types of access privileges to information stored in the database. The operating system supervises user access to raw data in the operating memory. The DBMSs, though is the primary line of defense in controlling user access to databases have the weakness of not able to guarantee that software errors. This database weakness is mainly due to the not advance

enough of the current state of software certification, database vendor policies or governmental politics.

Using cryptographic controls in database security, results in data sets being stored in databases in the form of suitable cryptograms. Cryptograms are non-readable to anyone without the appropriate cryptographic key to apply. Illegal leaking of such data will not compromise protection of the database. As cryptographic operations are usually carried out under the supervision of the operating system, transmission of information between the computer and the auxiliary memory is also protected [Seberry and Pieprzyk (1989)]. Gudes *et. al* (1976) and Seberry and Pieprzyk (1989) introduced cryptographic protection into a database system. They presented some basic cryptographic transformation techniques that preserve the database structure.

These are ten (10) characteristics of a database encryption system or ten (10) constraints imposed by typical database organizations and their application environments.

i. The encryption system should either be theoretically secure or require an extremely high work factor to break.
ii. Encryption and decryption should be fast enough not to unacceptably degrade system performance.
iii. The encrypted data should not have significantly greater volume than the unencrypted data.
iv. The encipherment must be record oriented, thus most stream ciphers are effectively eliminated from consideration.
v. The encryption scheme should support the logical subschema approach to databases (each field should be accessible under a different key(s)).
vi. An encrypted record must not consist of a series of individually encrypted fields. Rather, an encrypted record should be a single encrypted value, which is a function of all fields.

vii. The encryption scheme should be as flexible as possible with regard to the combinations of read and write privileges which can be granted and the conditions under which reads and writes must take place.

viii. The DBMS should be able to detect and reject records, which are created or updated under a false encryption key without being able to determine what the data are.

ix. The DBMS should not be forced by the encryption system to keep duplicate copies of data items in order to allow subschema to be represented.

x. Since records are likely to be built over a period, it must be possible to recover information from partially completed records in the same way it is recovered from full records.

Database encryption and authentication at the field level is not usually recommended for security reasons. The reasons are using encryption to hide individual data elements is vulnerable to cipher text searching; and using cryptographic checksums to authenticate individual data elements is vulnerable to plaintext or cipher text substitution. However, field based protection is advantageous as it allows projections to be performed and individual data elements decrypted or authenticated. To take advantage of this and also solving the above two (2) weaknesses, Denning (1983) proposed techniques for secure encryption and authentication at the field level. The principle idea behind her techniques is to generate a different key to encrypt (or authenticate) each data element in the database.

2.3 Cryptographic Capabilities Of Commercial DBMS

Research in the application of cryptography in databases has made some progress compared to its implementation in commercial database management systems. The current commercial database management systems have some form of cryptographic controls. The next paragraphs provide a brief preview of the cryptographic features pertaining to database encryption in commercial databases.

DB2 Universal Database is IBM's proprietary DBMS. IBM® DB2 Version 7.2 has the capability for encryption and decryption of string data through user-defined functions. With the built-in encryption and decryption functions provided, users can encrypt data to add an additional layer of security. The ENCRYPT function encrypts data using a password-based encryption method. The encryption function also allows for a password hint to be stored and another function is provided to get the hint without using the password. The DECRYPT functions decrypt data using a password-based decryption method.

Microsoft® SQL Server™2000 does not have any direct capability to encrypt data in its database. However, it uses Kerberos to support mutual authentication between the client and the server, as well as the ability to pass the security credentials of a client between computers. So that work on a remote server can proceed using the credentials of the impersonated client. Microsoft® SQL Server™ (Microsoft, 2001) has the capability to encrypt:

- Login and application role passwords stored in SQL Server.
- Any data sent between the client and the server as network packets.
- Stored procedure definitions.
- User-defined function definitions.
- View definitions.
- Trigger definitions.
- Default definitions.

- Rule definitions.

Oracle™ Corporation is undoubtedly the largest database software vendor and the largest provider of software for e-business today [Oracle Press (2000)]. Being the leader in secure, highly available database software Oracle first introduced Trusted Oracle7 MLS RDBMS in 1992. The product supports strict multilevel security policy on trusted operating system. In 1998, the Secure Access Tier 3 product was merged to produce Oracle Label Security in 1999, which is built on the Oracle8i Virtual Private Database (VPD). Oracle8i has the integration capability with external services like Distributed Computing Environment (DCE). Early 2001, with the release of Oracle9i, the database software now provides a wide range of solution ranging from data encryption across the network and within the database, consolidated user access across multiple applications, user authentication, server-defined access control policies, and extended auditing capabilities [Oracle (2001)].

Oracle has supported encryption of network data through Oracle Advanced Security, since Oracle7. Starting with Oracle8i release 2, Oracle has allowed selective data to be protected via encryption within the database as well. To address the need for selective data encryption, Oracle9i provides the *DBMS_OBFUSCATION_TOOLKIT PL/SQL* package to encrypt and decrypt stored data. The package supports bulk data encryption using DES algorithm, and includes procedures to encrypt and decrypt using DES. It also supports triple DES (3DES) encryption, in both two (2) and three (3) key modes. Oracle9i supports two (2) methods of encrypting data prior to storage in the database, which are a PL/SQL interface and a Java Cryptographic Extension (JCE) provider. In this proprietary package, users are not able to plug in a different encryption algorithm, nor make multiple passes of the function.

2.4 Smartcard Usage

The main purpose of using smartcard in the OraCrypt™ product is to store keys of the database owners. The keys cannot be stored in the database itself because of security reason. Some techniques were proposed to solve this problem either by storing the keys in diskettes, CD-ROMs or a file to store all the owner's keys. All the proposed ideas were not suitable due to the criteria of media size, storage space, processor needed and security features. Then the smartcard idea comes out. After doing some research, the project found that storing the keys in smartcard is more appropriate and secrecy than storing the keys in the database.

A smartcard is a portable, tamper-resistant computer with a programmable data store. Smartcards are different with magnetic stripe cards. The advantages of using smartcard are security, economical, convenience, customization and multi-functional. A smartcard has two (2) types of fundamental software; there are host software (runs on a computer connected to a smartcard and referred to as reader-side software) and card software (runs on the smartcard itself and referred to as card-side software). Host software is the most smartcard software used and will be written against existing smartcards, either commodity off-the-shelf smartcard available from manufacturers or created by major smartcard issuers such as bank associations, retailers, and national governments.

One of the first tasks of a host software programmer is to choose the smartcard that will be included in the system. There are a lot of smartcards developers and manufacturers in the market nowadays such as Atmel, EDS, Gemplus, IBM, NTRU Cryptosystems Inc, Philips Electronics, SC Solutions, SchlumbergerSema**,** SCM Microsystems, Turtle Mountain Communications, and Allied Solution. The current principles of smartcard standards include PKCS#11, which are Cryptographic Token Interface Standard, PC/SC, OpenCard, Java Card, Common Data Security Architecture, Microsoft Cryptographic API, Embossed and Difference between OpenCard and PC/SC.

Other industry standard groups are GSM, CEPS, EMV, Global Platform, and Open Platform.

2.5 Research On Existing Product

This section covers issues on several existing product that related to encryption and decryption process, and database security.

2.5.1 DBEncrypt

DBEncrypt is a flexible solution providing a means of encrypting rows and columns in a database. DBEncrypt provides a complete database encryption solution including a variety of strong encryption algorithms to pick from, templates to build own encryption procedures from, as well as a point-and-click user interface for installing and managing the encryption.

2.5.1.1 Turn-Key Solution

DBEncrypt transparently encrypts data on a per column basis. The user picked the columns for encryption and decryption and DBEncrypt does the rest in seconds. As an encryption solution, DBEncrypt not only saves thousands of hours of research and development but also provides a secure solution that has been reviewed by the security community. DBEncrypt allows database administrators and developers to encrypt data develop an entire key management system themselves.

2.5.1.2 Low Level Application Programming Interface (API)

DBEncrypt provides a direct interface to encryption functions that allow database administrators and developers to design and build their own encryption features. From PL/SQL, developers can access a rich set of encryption features including hashing functions, public key ciphers, block and stream ciphers, ciphers to sign and verify data, and random numbers generators. This low- level API provides the developer the flexibility to choose details such as the padding mode or the key size. As a low- level interface, DBEncrypt provides developers with the ability to utilize encryption as he sees fit. These developers are empowered with the ability to extend the current DBEncrypt features as well as design entirely independent solutions.

2.5.1.3 Features Of DBEncrypt

DBEncrypt features are:

i. Database column and row level encryption
   DBEncrypt provides database column and row-level encryption callable from SQL and PL/SQL. It was designed to be incredibly efficient for PL/SQL programmers and lucidly simple for the business user. Effective simplicity in design directly translates into the organization to saving time and money.

ii. Access to world class security resources that facilitate effective data protection
   DBEncrypt™ provides enterprise with strength protection for the database by giving user full access to a wide variety of block and stream ciphers, public key algorithms, message authentication codes, and one-way hash functions, as below:

   - Symmetric Algorithms (Block) - AES, DES, DESX, Triple DES (2 and 3 key), Blowfish, Twofish, RC2, Serpent, CAST128, CAST256 and Skipjack
   - Symmetric Algorithms (Stream) - RC4_compatible
   - Public Key Algorithms - RSA

- Hashing Algorithms - SHA1, MD5
- Message Authentication Code - HMAC_SHA1

iii. Mechanism in providing authentication, encryption and data integrity

Tools and templates are provided to create strong authentication, encryption, and data integrity of the information within the database. DBEncrypt™ provides an interface to a group of both low- level and high- level encryption functions. It is also provided an interface to generate secure random numbers, strong encryption keys, and initialization vectors.

2.5.1.4 DBEncrypt Pros And Cons

DBEncrypt security solution provides a way to meet security and privacy in the database. This is because DBEncrypt is transparent to encrypt data on a per column basis. With DBEncrypt solution, it allows database administrators and developers to encrypt data without having to invent and develop an entire key management system themselves. In DBEncrypt, tools and templates are provided to create strong authentication, encryption, and data integrity of the information within the database. It also provides an interface to generate secure random numbers, strong encryption keys, and initialization vectors.

DBEncrypt provides a direct interface to encryption functions that allow DBA and developers to design and build their own encryption features. From PL/SQL, developers can access a rich set of encryption features including hashing functions, public-key ciphers, block and stream ciphers, and random numbers generators. As a low-level interface, DBEncrypt provides developers with the ability to utilize encryption as he sees fit. DBEncrypt also provides database column and row-level encryption callable from SQL and PL/SQL.

2.5.2 Protegrity Secure.Data™

Secure.Data™ is a granular privacy solution for sensitive information and records in databases. It released by Protegrity, an information security company, specializing in the encryption of databases. The product includes Secure.Data Manager and Secure.Data Server modules. Secure.Data Manager is the security official that defines user, controls data-access privileges, and specifies data to be encrypted and other security parameters. Secure.Data Server is the active component, performs real-time security processing of encryption and decryption. Secure.Data™ provides encryption protection for information at a data item level. It allows organizations to manage and control all access to information. Secure.Data provides dynamic and real-time enterprise-wide database protection across major platforms and operating systems.

2.5.2.1 Unique Approach

The Protegrity Secure.Data program suite takes a granular approach to information security. Instead of building walls around servers or hard drives, Secure.Data builds a protective layer of encryption around individual data items or objects. This concept protects the data wherever it is stored or processed instead of protecting the server that stores or protects the data (*Figure 2.1*).

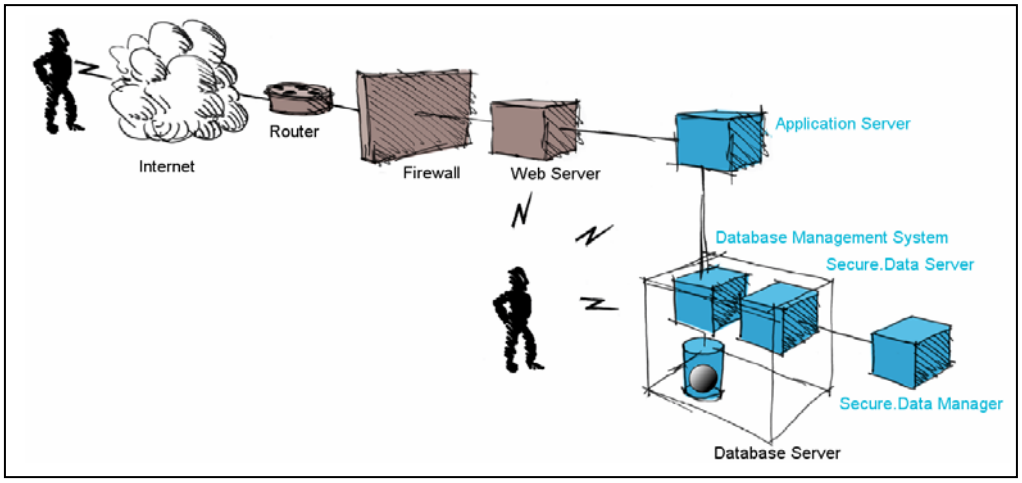


Figure 2.1 Secure.Data Unique Approaches

The Protegrity solution adds a critical and final checkpoint for users accessing sensitive information. Secure.Data remains in place protecting the data at the closest and most effective point. This prevents outside attacks as well as infiltration from within the server itself.

2.5.2.2 Data Item Protection

Secure.Data allows an organization manager to define which data stored in databases that is sensitive and which individuals or groups that should have the authority to access the information. This selection is done on a data item level, thereby focusing protection only on the sensitive data. The sensitive data is then encrypted and stored in the database. The selectivity of Protegrity's data item protection technique not only prevents intruders from gaining access to sensitive data, but also minimizes the delays or burdens on the system that may occur from other bulk encryption methods. The Protegrity approach is transparent to applications and does not affect non-sensitive data.

2.5.2.3 Role Based Access

The role based access model is the cornerstone of the Secure.Data work-enabling design. This method allows managers to decide the levels of user access that are appropriate for various types of information. In Secure.Data, roles are created and determined as well as defined the unique data access privileges for each user. These roles are used to authenticate and authorize any application requests made from user queries. Roles can be assigned to users on a specific, individualized basis or general role types can be created to embody the functions commonly performed by other individuals who need secure access to sensitive data. All roles enterprise-wide are managed from the Secure.Data Manager and enforced through the Secure.Data Server. Secure.Data can import information from the organizations' preexisting access control facilitates to minimize data entry and management.

2.5.2.4 Encryption

Secure.Data utilizes encryption to transform sensitive data into a form that is unreadable to anyone who does not have proper authorization. To achieve maximum protection of data and maintain the best system performance, Protegrity offers its crypto-conductor process. This system allows implementation of encryption down to the column level within databases. While other security technologies encrypt whole files, tables or databases, Secure.Data lets users specify and secure only the sensitive data items that need protection. The crypto-conductor process supports encryption for up to 168 DES, strong (CBC block cipher) encryption with IV. Data encryption is the only way to absolutely protect sensitive data, eliminate potential data sharing issues, ensure that privacy is maintained, minimize the potential for identity theft, and ultimately meet data privacy compliance requirements. There are two (2) basic approaches to encryption of stored data:

i. The entire database can be encrypted. There are two (2) problems with this approach. First, it allows anyone with access to the database to decrypt all data stored therein. Second, it exacts a substantial performance penalty because all requests, whether for sensitive data or not, it requires data to be decrypted and re-encrypted.

ii. Individual database applications can be modified to encrypt sensitive data. This requires the use of crypto toolkits to reengineer in-house applications, a costly and inflexible application-by-application proposition that does not offer a scalable and uniform approach to data privacy. In addition, this approach does not provide essential key management functions and may actually increase the number of people requiring access to databases for development and maintenance purposes.

2.5.2.5 Protegrity Secure.Data™ Pros And Cons

Secure.Data provides a way to meet security and privacy requirements without having to alter the code in the applications. This is called application transparency. With the Protegrity solution, users can no longer bypass security policies embedded in applications because security policies are associated directly with data. This solution unifies access controls and protection of information into a central protected database repository, and at the same time closes the security loopholes that inevitably occur when adding, deleting, removing, and changing user access rights to sensitive information stored in a database.

The Protegrity solution allows the company to predetermine who is authorized to have access to sensitive information within corporate databases and to apply the strongest mechanisms of encryption based security to this select information. Above this core functionality, the Protegrity solution then provides automatic encryption key management. The following are the benefits of the product that ensure protection of corporate information assets, protect its customers, and consumers.

- Protection

  Secure.Data Manager operates in its own self-secure encrypted environment, protected by strong authentication that ensures the confidentiality and integrity of all Secure.Data security parameters. Secure.Data Manager runs on a workstation separate from both application and database servers.

- Simple user interface

  Secure.Data Manager's graphical user interface incorporates the latest Windows techniques, such as *drag and drop* and *browse and pick*. The easy-to-use GUI greatly simplifies implementation and management procedures, such as assigning access privileges and choosing data protection methods.

- Separation of duties

  It allows organizations to separate database security control from the duties of the database and system administrators. The database or system administrator can still perform his duties, but an appointed Security Manager maintains responsibility for the security of the sensitive information through the Secure.Data Manager module, which is operated separately from the database.

- Advanced prevention of internal attacks

  Secure.Data provides a sophisticated system of self-protection that protects the encryption mechanism in a way that it cannot be manipulated by a DBA or System Administrator. Through this self-protection mechanism, no user, even with root/administrator rights, can undetected access the encrypted data.

- High granularity

  By using role-based access control and multilevel security, Secure.Data Manager allows operators to define user roles, workgroups, access rights and encryption requirements down to the data item level.

- Single point of control

  Secure.Data does not require security operators to update security parameters for various servers individually. Secure.Data Manager's centralized administration allows operators to maintain the integrity of all Secure.Data policies and configurations from a single location.

- Auditing and reporting

  Encrypted logs are produced to track both the Secure.Data Manager administrator's activities and Secure.Data Server activities around protected data, such as records of changes to the security policy and unauthorized access attempts. Real-time auditing ensures non-repudiation of transactions on sensitive information and is independent of the DBMS audit mechanism. Secure.Data Manager also includes a report generator with various templates that can be tailored by the security operator and exported for use. All logs are of common format regardless of the Transaction

Protocols and Database Managers in use. Secure.Data encrypts all logs and audits to ensure the fullest protection and security.

2.6 Existing Product Comparative Study

The comparative study was done on the security features and characteristics of four (4) existing database security systems, which uses cryptography to secure data in databases. The existing systems are DBEncrypt™, Secure.Data™, Microsoft SQL Server 2000, and Oracle Database Security.

2.6.1 DBEncrypt vs. Secure.Data Features

Table 2.1 shows the comparative studies on characteristics and features between DBEncrypt and Secure.Data.

Table 2.1 DBEncrypt vs. Secure.Data

Characteristics	Features	DBEncrypt	Secure.Data
Installation and un-installation	Easiness	Fast and easy for non-expert user.	Fast but tedious for non-expert user.
	Modules	Install client-side module, then install server side module.	Install Secure.Data Manager, then install Secure.Data Server.
	Setup and configuration	Provide sample database for tutorial.	Provide sample database for tutorial.
	Platforms	Server-side: ▪ Database - Oracle 8i, MySQL ▪ OS - Sun Solaris, HP-UX, Linux, MsWindows NT/2000 Console: ▪ OS - MsWindows NT 4.0 SP3 / 2000	Server-side: ▪ Database - Oracle 8i, MySQL ▪ OS - Sun Solaris, HP-UX, Linux, MsWindows NT/2000 Console: ▪ OS - MsWindows NT 4.0 SP3 / 2000

Characteristics	Features	DBEncrypt	Secure.Data
		▪ HDD - 10MB ▪ RAM - >32MB	▪ HDD - 10MB ▪ RAM - >32MB ▪ VPN-1/FireWall-1 v4.1 SP1 or higher (gateway)
User Interface		▪ Simple interface for selecting column to be encrypted, and user-key generation. ▪ Provide space to execute PL/SQL commands. ▪ User-friendly GUI interface with *point-and-click*.	▪ Provide Security Policy interface for script generation. ▪ Generate script for user to use for encrypt and decrypt. ▪ GUI Interface with Multiple Document Interface (MDI) appearance. ▪ GUI incorporates the latest Windows techniques, such as *drag and drop* and *browse and pick*.
Capability	Encryption / Decryption	Encrypt and decrypt process on column only.	Encrypt and decrypt process on column, row and data element.
	Data type supported	▪ VARCHAR2 ▪ NUMBER (without precision or scale)	▪ VARCHAR ▪ Real, Float, Integer
	Algorithms	Provide 14-encryption algorithm - AES, DES, DESX (120-bits), Triple DES, Blowfish, Twofish, RC2 compatible, RC4 compatible, Serpent, CAST128, CAST256, SKIPJACK, RSA, SHA1.	168 DES, strong (CBC block cipher) encryption with IV (Initialization Vector).
	Help	▪ Good coverage on the package capability. ▪ Provide examples. ▪ Wizard supported.	▪ Good coverage on the package capability.
	Key management	Secret keys are kept in 3 ways: i. Encrypted with password ii. Operating system iii. Table within database	Using dynamic 2 factor RSA SecureID authentication: i. User knows ii. User possess
	Key generation	Private and Public key model.	Role based model.
	Clustering	Have to install in each database server.	Have to send the encryption and decryption process to database server .
	Impact on	▪ Minimum degradation on	▪ No need to encrypt all data.

Characteristics	Features	DBEncrypt	Secure.Data
	database	encrypted data. ▪ No need to encrypt all data.	
General	Availability	Trial version license provided for 2 weeks.	General.
	Technical documentations	Provided in full.	Provided in full.

2.6.2 Microsoft SQL Server 2000 vs. Secure.Data Features

Table 2.2 shows the comparative studies on features between Microsoft SQL Server 2000 and Secure.Data.

Table 2.2 Microsoft SQL Server 2000 vs. Secure.Data

Microsoft SQL Server 2000	Secure.Data
User authentication	**Maintain the existing database authentication**
User authenticate once with single username/password.	Secure.Data sit on top of the DBMS and it fully utilized the default authentication of the DBMS.
Predefined server and database roles for application users	**Enhanced role, row and time based access control**
Set the access control by predefined roles in the database, allow DBA to assign the relevant user to that role.	Access is enhanced by the roles and workgroups; additional security levels can also be defined for workgroups to further enhance row-level access control. Further enhanced by time based access controls, which allow user to login to the database in specific time range in a day or a period.
Data protection (in transit)	**Data protection (encryption for data at rest)**
Secure Socket Layer (SSL) encryption of the data and other network traffic as it travels between the client and server system.	Information is encrypted during transit and at rest stage. Information is unreadable to anyone who is not authorized, it will only be decrypted back if the user is authorized.
Server-based encrypted data, passwords and stored procedures	**Advances and automatic encryption key management**
Further enhanced to use the Windows Crypto API.	Unified protection across major database platforms. Allow implementation of strong cryptographic algorithms, 3DES based on CBC and IV.

2.6.3 Oracle vs. Secure.Data Features

Table 2.3 shows the comparative studies on features between Oracle Database Security and Secure.Data.

Table 2.3 Oracle vs. Secure.Data

Oracle 8i/9i	Secure.Data
User authentication to identify users throughout all tiers of the network	**Maintain the existing database authentication**
User authenticate once with single username/password. Centralization of user access information in oracle internet directory. PKI integration functions for easier deployment and management.	Secure.Data sit on top of the DBMS and it fully utilized the default authentication of the Oracle DBMS. Secure.Data can maintain all DB username and password.
Role and row based access control	**Enhanced role, row and time based access control**
Set the access control by predefined roles in the organization, and assigned the relevant user to that role. Oracle Virtual Private Database (VPD) offers fine-grained access control or row level securities can only access rows of data that belong to him/her by labeling data at the row level.	Access is enhanced by the roles and workgroups; additional security levels can also be defined for workgroups to further enhance row-level access control. Further enhanced by time based access controls, which allow user to login to the database in specific time range in a day or a period.
Data protection (in transit)	**Data protection (encryption for data at rest)**
Oracle advanced security provide SSL RC4 and 3DES encryption which ensures data confidentially and integrity as data travels across the network.	Information is encrypted during transit and at rest stage. Information is unreadable to anyone who is not authorized, it will only be decrypted back if the user is authorized.
Stored data protection	**Advances and automatic encryption key management**
Oracle Pl/SQL encryption toolkit allows for development of native database encryption capabilities.	Unified protection across major database platforms. Allow implementation of strong cryptographic algorithms, 3DES based on CBC and IV
Auditing for accountability	**Selective and secured auditing**
Audit track users database activity.	Monitoring the activity around protected sensitive information. All audit information is encrypted and stored.

2.7 Software

Humphrey (1990) formally defines software as a program and all the associated information and materials needed to support its installation, operation, repair, and enhancement. This definition has some implications on software projects. Firstly, it widens the scope of software to include documentations (all of them throughout the lifecycle), support, training, and maintenance. Secondly, it extends the notion of software lifecycle: the lifecycle does not end when the product is shipped, but seems rather un-ending instead. These implications stress the potential largeness of software projects, which in turn implies the need for some kind of processes.

2.8 Process

Process is defined as a series of actions and operations by the Webster dictionary. Process thus implies a structured, ordered set of activities. We can also think of process as workflow in the context of product development.

2.9 Software Process

The software development process is the central process of any software development effort. Humphrey (1990) defined software process as a set of activities, methods, and practices that are used in the production and evolution of software. In short, a software process defines who is doing what, when and how to reach a certain goal. Thus, software process could be thought as an ordered set of activities, methods, and practices that are used in anything that is related to software development. In other words, all activities that are related to software projects could be put within some software processes, or software processes are applicable to all activities in a software project. The purpose of a software development process is to produce high quality and timely results without imposing a large overhead on the project. The process is depicted as a model that determines the order of stages involved in development and evolution

establishes transition criteria for progressing between stages by specifying the entrance and exit criteria.

2.10 Rational Unified Process (RUP)

The Rational Unified Process (RUP) is based on the integrated work of three (3) primary methodologists, Ivar Jacobson, Grady Booch and James Rumbaugh. These methodologists, aided by a large and extended methodologist community, were assembled by Rational Corporation to form a unified, cohesive and comprehensive methodology framework for the development of software systems. There work, occurring over several years and based on existing, previously tested methodologies, have lead to significant standards in the development community, including the general acceptance of use cases and the Unified Modeling Language™ (UML).

2.10.1 RUP Definition

Ivar Jacobson, Grady Booch and James Rumbaugh (1999), and Kruchten (1999) state that RUP software process is a generic business process for object oriented software engineering. It describes a family of related software engineering processes sharing a common structure and common process architecture. It provides a disciplined approach to assigning tasks and responsibilities within a development organization. The exactly definition of RUP can be derived from different perspectives, as below:

i. Software engineering process

RUP is a software engineering process that provides a disciplined approach to assigning tasks and responsibilities within a development organization. Its goal is to ensure the production of high-quality software that meets the needs of its end-users, within a predictable schedule and budget [Ivar Jacobson, Grady Booch and James Rumbaugh (1999), Kruchten (1999)].

ii. Process product

RUP is a process product, developed and maintained by Rational® software. The development team for the RUP is working closely with customers, partners, Rational's product groups as well as Rational's consultant organization, to ensure that the process is continuously updated and improved upon to reflect recent experiences and evolving and proven best practices.

iii. Unified Modeling Language (UML)

RUP is a guide for how to effectively use the UML. The UML is an industry-standard language that allows us to clearly communicate requirements, architectures and designs. The UML was originally created by Rational® software, and is now maintained by the standards organization Object Management Group (OMG) [Booch *et. al.* (1988)].

iv. Tools

The RUP is supported by tools, which automate large parts of the process. They are used to create and maintain the various artifacts of the software engineering process, which are visual modeling, programming, and testing. They are invaluable in supporting all the bookkeeping associated with the change management as well as the configuration management that accompanies each of iteration.

v. Best practices

RUP captures many of the best practices in modern software development in a form that is suitable for a wide range of projects and organizations. Deploying these best practices by using the RUP will offers development teams a number of key advantages.

2.10.2 Phases In RUP

RUP consists of cycles that may repeat over the long-term life of a system. A cycle consists of four (4) phases: Inception, Elaboration, Construction, and Transition [Kruchten (1995)]. The software lifecycle is broken into cycles, each cycle working on a new generation of the product. Each phase is concluded with a well-defined milestone (*Figure 2.3*), which means a point in time at which certain critical decisions must be made and therefore key goals must have been achieved [Barry W. Boehm (1996)].

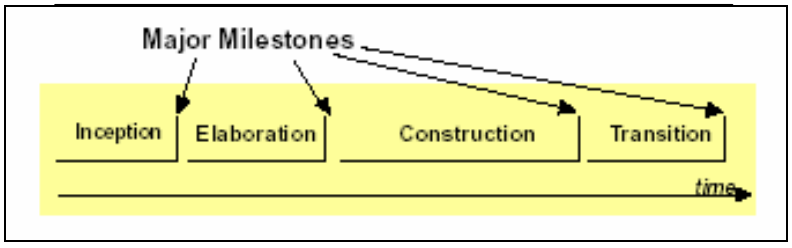


Figure 2.3 The Phases and Major Milestones in the RUP

2.10.2.1 Inception Phase

During the inception phase (*Appendix B-1*) the core idea is developed into a product vision. In this phase, we review and confirm our understanding of the core business drivers. We want to understand the business case for why the project should be attempted. The inception phase establishes the product feasibility and delimits the project scope. To accomplish this you must identify all external entities with which the system will interact and define the nature of this interaction at a high-level. This involves identifying all use cases and describing a few significant ones. The business case includes success criteria, risk assessment, and estimate of their sources needed, and a phase plan showing dates of major milestones [Kruchten (1995), Walker Royce (1998)].

At the end of the inception phase is the Lifecycle Objectives Milestone. The evaluation criteria for the inception phase are:

- Stakeholder concurrence on scope definition and cost/schedule estimates.
- Requirements understanding as evidenced by the fidelity of the primary use cases.
- Credibility of the cost/schedule estimates, priorities, risks, and development process.
- Depth and breadth of any architectural prototype that was developed.
- Actual expenditures versus planned expenditures.

2.10.2.2 Elaboration Phase

The purpose of this phase (*Appendix B-2*) is to analyze the problem domain, establish a sound architectural foundation, develop the project plan, and eliminate the highest risk elements of the project. During the elaboration phase, the majority of the use cases are specified in detail and the system architecture is designed. Architectural decisions have to be made with an understanding of the whole system, which are scope, major functionality and nonfunctional requirements such as performance requirements. At the end of this phase, the hard engineering is considered complete and the decision on whether or not to commit to the construction and transition phases. For most projects, this also corresponds to the transition from a mobile, light and nimble, low-risk operation to a high-cost, high-risk operation with substantial inertia. While the process must always accommodate changes, the elaboration phase activities ensure that the architecture, requirements and plans are stable enough, and the risks are sufficiently mitigated, so you can predictably determine the cost and schedule for the completion of the development.

In the elaboration phase, an executable architecture prototype is built in one or more iterations, depending on the scope, size, risk, and novelty of the project. This effort should at least address the critical use cases identified in the inception phase, which typically expose the major technical risks of the project. While an evolutionary prototype of a production-quality component is always the goal, this does not exclude the development of one or more exploratory, throwaway prototypes to mitigate specific risks such as design/requirements trade-offs, component feasibility study, or demonstrations to investors, customers, and end-users.

At the end of the elaboration phase is the Lifecycle Architecture Milestone. At this point, you examine the detailed system objectives and scope, the choice of architecture, and the resolution of the major risks. The main evaluation criteria for the elaboration phase involve the answers to these questions:

- Is the vision of the product stable?
- Is the architecture stable?
- Does the executable demonstration show that the major risk elements have been addressed and credibly resolved?
- Is the plan for the construction phase sufficiently detailed and accurate? Is it backed up with a credible basis of estimates?
- Do all stakeholders agree that the current vision can be achieved if the current plan is executed to develop the complete system, in the context of the current architecture?
- Is the actual resource expenditure versus planned expenditure acceptable?

2.10.2.3 Construction Phase

During the construction phase (*Appendix B-3*), all remaining components and application features are developed and integrated into the product, and all features are thoroughly tested. The construction phase is, in one sense, a manufacturing process where emphasis is placed on managing resources and controlling operations to optimize costs, schedules, and quality. In this sense, the management mindset undergoes a transition from the development of intellectual property during inception and elaboration, to the development of deployable products during construction and transition. Many projects are large enough that parallel construction increments can be spawned. These parallel activities can significantly accelerate the availability of deployable releases; they can also increase the complexity of resource management and workflow synchronization. A robust architecture and an understandable plan are highly correlated. In other words, one of the critical qualities of the architecture is its ease of construction.

At the end of the construction phase, you decide if the software, the sites, and the users are ready to go operational, without exposing the project to high risks. This release is often called a *beta* release. The evaluation criteria for the construction phase involve answering these questions:

- Is this product release stable and mature enough to be deployed in the user community?
- Are all stakeholders ready for the transition into the user community?
- Are the actual resource expenditures versus planned expenditures still acceptable?

2.10.2.4 Transition Phase

The purpose of the transition phase (*Appendix B-4*) is to ensure that the requirements have been met to the satisfaction of the stakeholders and transit the software product to the user community. Once the product has been given to the end user, issues usually arise that require developing new releases, correcting some problems, or finish the features that were postponed. The transition phase is entered when a baseline is mature enough to be deployed in the end user domain. This typically requires that some usable subset of the system has been completed to an acceptable level of quality and that user documentation is available so that the transition to the user will provide positive results for all parties.

The transition phase focuses on the activities required to place the software into the hands of the users. Typically, this phase includes several iterations, including beta releases, general availability releases, as well as bug fix and enhancement releases. Considerable effort is expended in developing user-oriented documentation, training users, supporting users in their initial product use, and reacting to user feedback. At this point in the lifecycle, user feedback should be confined primarily to product tuning, configuring, installation, and usability issues. The primary objectives of the transition phase include:

- Achieving user self-supportability.
- Achieving stakeholder concurrence that deployment baselines are complete and consistent with the evaluation criteria of the vision.
- Achieving final product baseline as rapidly and cost effectively as practical.

At the end of the transition phase is the Product Release Milestone. At this point, we decide if the objectives were met, and if we should start another development cycle. In some cases, this milestone may coincide with the end of the inception phase for the next cycle. The primary evaluation criteria for the transition phase involve the answers to these questions:

- Is the user satisfied?
- Are the actual resources expenditures versus planned expenditures still acceptable?

CHAPTER 3

PROJECT METHODOLOGY

3.1 Introduction

Almost all operations need some kind of processes. Software development is no exception. Software development is a complex task. The researcher handled complexity by working with different models of the system to develop and each focusing on certain aspects of it. The development process is based on architecture, method, process, and tools. The organizations that developed security software and implemented well software engineering process have improved the quality of the software product and process itself. Therefore, software process is instrumental in the success of software management. To aim this goal, the researcher implements the software engineering process to support the development of OraCrypt™ product.

3.2 Software Development Methodology

Software engineering is known as an engineering discipline frequently characterized the approach as a practical, orderly and measured development of software. The goals of software engineering focus on the development of software that supports the needs of the user and helps the maintainer to adapt the software so that it will continue to support the evolving needs of the user.

In this chapter, the software development methodology focuses on software process, software methods, software techniques and tools used for this project. It comprises of well-defined steps and techniques to built a right system with the effective software development strategy. The choice of the software development methodology is one of the major contributors to the success or failure of software development projects because it defines the way a system is built and organized.

3.3 RUP As A Software Development Process

In RUP based project, the overall architecture of the software development process goes through four (4) phases, which are Inception, Elaboration, Construction, and Transition phase. In this section, the researcher details the objectives, purposes and activities involved for each phases.

3.3.1 Inception Phase

The goal of this phase is to help the project team decided what the objectives of the project would be. It tells everyone what the system is, and may also tell who will use it, why it will be used, what features must be present, and what constraints exist. The researcher established the scope of OraCrypt™ product, what is included, and what is not, and its boundary conditions. The researcher wanted to understand the business case for why the project should be attempted. To accomplish this, all external entities are identified with which the product will interact and defined the nature of this

interaction at a high-level. This involved the identification of actors and use cases for this system. And then, the researcher drafted the most essential use case that will drive the major design trade-off. Businesses and requirements risks were also addressed before the project can proceed. During the inception phase, the researcher has identified an iteration that called *Preliminary Iteration*.

3.3.1.1 Preliminary Iteration

The preliminary iteration involved project management, software requirement analysis, and environment workflow. In this section, the researcher details the activities for each workflow involved in this iteration during the inception phase.

i. Environment Workflow

This workflow describes on the engineering part of the OraCrypt™ product development. The environment discipline focuses on the activities necessary to configure the process for the project. It describes the activities required to develop the guidelines in support of the project. The purpose of the environment activities is to provide the software development organization with the software development environment, with both processes and tools that will support the development team. Tools include those for modeling, testing, requirements, management, coding, planning, and tracking. Practically, all the works and activities in the environment workflow are done during the inception phase. Many of the tasks associated with the process can benefit from the use of tools. The following activities were carried out during the environment workflow (*Appendix D-1*).

- Prepared the Development Case

  The researcher created a first version of the Development Case of the project. It will be developed in increments, covering more and more of the disciplines in each of iteration. The Development Case includes phases and milestones;

artifacts to produce, activities to perform, the way to work in each discipline, and iteration plan descriptions for this project.

- Prepared the software specification

  The minimum software specifications needed to carry out development of OraCryptTM product are stated as follow.

  a. Project Management Workflow

Table 3.1 Lists of Tools in Project Management Workflow

Software	Purpose
Microsoft Project 2000	Project tracking and planning.
Microsoft Word 2000	Documentation tools.

  b. Requirements Workflow

Table 3.2 Lists of Tools in Requirements Workflow

Software	Purpose
Microsoft Word 2000	Documentation tools.
Rational Rose Tool Evaluation Version	Software requirements tools.
Rational Requisite Pro	Requirements management tools.

  c. Analysis and Design Workflow

Table 3.3 Lists of Tools in Analysis and Design Workflow

Software	Purpose
Microsoft Word 2000	Documentation tools.
Rational Rose Tool Evaluation Version	Analysis and design tools.

  d. Implementation Workflow

Table 3.4 Lists of Tools in Implementation Workflow

Software	Purpose
Microsoft Visual Basic 6.0	Software development tools.

Oracle 8i Enterprise Edition Release 8.1.7.0.	Database management system.
Microsoft Word 2000	Documentation tools.

- Prepare the hardware specification

  The minimum hardware specification that is required for the development OraCrypt™ product is stated as follow.

Table 3.3 Hardware Specifications

No	Requirements
1	Operating system – Microsoft Windows 2000 Server
2	Processor – Intel Pentium IV
3	Hard disc – 30 GB
4	RAM – 512 MB
5	Monitor – 14 inch
6	Mouse – serial port

ii. Project Management Workflow

Project management is conducted to provide a framework for managing and understand the structure and the dynamics of the organization in which an OraCrypt™ product is to be deployed. In this workflow, the researcher identified the process and improvement potentials for the organization to ensure everyone have a common understanding of the target organization. The system requirements are derived to support the target organization. Therefore, the researcher provided the guidelines for planning, staffing, managing risk, executing, and monitoring the project. The following activities were carried out during the project management workflow (*Appendix D-2*).

- Established the development team

  It is important to have at least an experience team member to ensure the objectives of software development are catered.

- Formulated the scope of project

The purpose of this activity is to reappraise the project's intended capabilities and characteristics. Therefore, the researcher captured the context and most important requirements in enough detail to derive acceptance criteria for the product to support target organization. The development needs to know what the scope of the project is and how it will satisfy the user of the organization.

- Estimated the potential risks

  Many decisions in an iterative lifecycle are driven by risks. In order to achieve this, the researcher need to get a good grip on the risks the project is faced with, and have clear strategies on how to mitigate or deal with them. So, the researcher estimated all the possible risks, planned for the risk mitigation to reduce the probability or impact, planned for contingencies and monitored the risks.

- Developed the Software Development Plan

  The researcher developed the Software Development Plan that captures the overall envelope of the project, for one cycle and the following cycles, and all of the information necessary to control the project. The researcher planned the project schedule and resource needs, and to track progress against the schedule. In SDP, the researcher also described the approach to the development of the software, and the top-level plan for directing the development effort. To get a good plan and estimation, the researcher getting the technical estimation from the developers who will implement an OraCrypt™ product features.

- Developed the Iteration Plan

The researcher developed the iteration plan that contains set of activities and tasks, with assigned resources, containing task dependencies for the iteration. In this project, there are two (2) iteration plans active at any given point, which are current iteration plan and the next iteration plan. The current iteration plan is used to track the project progress, while the next iteration plan is built towards the second half of the current iteration, and it is ready at the end of the current iteration.

- Closed-out the inception phase

  The researcher brings the phase to closure by ensuring that all major issues from the previous iteration are resolved, and the state of all artifacts is known.

iii. Software Requirements Workflow

Requirement is perhaps the most important process of the software process model. Without the correct requirements, subsequent design and implementation of the application will be practically meaningless. Hence, software requirements analysis is conducted to find out as much requirements related to the OraCrypt™ product. The purpose of requirements analysis is to establish a common understanding between the user and the software development team concerning all requirements for the defined software development project. All the functional and non-functional requirements were gathered using the following techniques:

- Internal discussion

  Discussions were conducted among team members and the supervisor to obtain a deeper understanding of the current system and the additional features required of OraCrypt™ product.

- Study of previous system

  Several documents of previous system were studied and referred in order to understand some features of OraCrypt™ product.

Finally, all requirements are managed and organized in efficient ways to build a common understanding of the project needs and commitments among the user. The following activities were carried out during the software requirements analysis workflow (*Appendix D-3*).

- Analyzed the problem

  Analysis of the problem involved identified the user, defined the system boundary, and identified the constraints imposed on the system to ensure that everyone agreed on what the problem is that needs to be solved. In order to fully understand the problem that need to be addressed, the researcher identified the high-level user view of the system to be built that will be represented in the use case model. It is also important to agree on common terminology, which will be used throughout the project by defined the project terms in a glossary.

- Outlined and clarified the user needs

  The researcher collected and elicited information from the user of the project in order to understand what their needs really are. The collected user request is used as primary input to defining the high-level features of OraCrypt™ product. This activity was done using various techniques such as storyboard and brainstorming.

- Defined the boundaries of the system

  The researcher defined the system boundaries that describe an envelope in which the solution system is contained. All the interactions with the system occur via interfaces between the system and the external world were defined. Therefore, we used Use Case Diagram as a concise summary of information contained actors and use cases to show how the actor interacts with the system and what the system does. The use case has a set of sequence actions and performs observable results to a particular actor, who interacts with the system. A use case diagram for OraCrypt™ product is shown in *Appendix C*. In the earlier phase, there were three (3) actors and six (6) use cases defined in the

OraCrypt™ product. The following are the descriptions of each actor and use case involved.

a. Actor: Security Manager (Sec Mgr)
   Security Manager is the person who responsible to ensure the system is in secured. He manages the system and database users to the proper organization structure. He also responsible to generate, manage and distribute the user key to all users. The system provides the different authentication for the Security Manager to access the OraCrypt™ product.

b. Actor: Normal User
   Normal user has the right access to use the encryption and decryption module, while user with authorized access can also manages his structure.

c. Actor: Smartcard Reader (S/Card Reader)
   Smartcard reader is the external actor for the OraCrypt™ product. It provides the communication between the smartcard and the OraCrypt™ product.

d. Use Case: Login
   This use case shows the process of verification and authentication user to the OraCrypt™ product. It enables the user to establish connection to the database.

e. Use Case: Encrypt Data
   This use case shows the encryption process that allows the user in the database to access his table for encryption purpose. The system provides three (3) levels of encryption, which are data element, column or row. In each level, user may provide condition for the selective encryption of data at those three (3) levels.

f. Use Case: Decrypt Data

This use case shows the decryption process that allows the user in the database to access the appropriate table for decryption purpose. The system provides three (3) levels of decryption, which are data element, column or row. In each level, user may provide condition for the selective decryption of data at those three (3) levels.

g. Use Case: Generate Key

This use case shows the generation of key in the OraCrypt™ product. It produced three (3) types of key, which are System Key, Master Key and Shared Key (with Control and without Control). The generated key will be managed and distributed to the user.

h. Use Case: Manage User Info

This use case is used by Security Manager to manage the user information that related to the system and the smartcard. He can imports the user information from the database to the system, add the user or user information, deletes the user or user information, and manages the smartcard services.

i. Use Case: Manage Key Structure

User used this use case to manage the hierarchical structure. The hierarchy structure can be build for organization, project, compartments, or group purpose.

- Established the scope of the system

The purpose of this activity is to make the scope of the system being developed as explicit as possible, and focused on a manageable body of requirements work for the iteration. The researcher prioritized and refined the input to the selection of requirements that are to be included in the current iteration. The researcher also determined the set of scenarios and use cases that represent some significant central functionality, and have a substantial architectural

coverage. These use cases or scenarios drive the development of the architecture and should be addressed in the current iteration.

The inception phase ends at the *Business Case Review Milestone* that marks the Go/No decision for the project budget and time as planned. At this point, the researcher examined the lifecycle objectives of the project, and decided to proceed with the project. The deliverables produce in this phase are listed as following:

- Glossary
- Development Case
- Software Development Plan
- Preliminary Plan for Inception Phase
- Iteration Plan for Elaboration Phase
- Use Case Specifications
- Supplementary Specifications

3.3.2 Elaboration Phase

The elaboration phase is where the foundation of software architecture is laid. The problem domain is analyzed, bearing in mind the entire system that is being built. The goals of this phase are to refine the support environment, defined, validated and baseline the architecture as rapidly as is practical. During this phase, a detailed plan for the construction phase will be baseline to provide a stable basis for the bulk of the design and implementation workflow. The system architecture evolves out of a consideration of the most significant requirements and an assessment of risk. The stability of the architecture was evaluated through the architectural prototypes. To ensure that the architecture, requirements and plans are stable enough, and the risks sufficiently mitigated, the researcher determined the schedule for the completion of the development. The architecture derived from the significant scenarios, which typically

expose the risks of the project, is established. In this phase, the researcher defined an iteration called *E1 Iteration* that purposely to develop architectural prototype.

3.3.2.1 E1 Iteration (Develop Architectural Prototype)

In this iteration, the workflows involved are environment, project management, software requirement analysis, and analysis and design workflow. In this section, the researcher details the activities for each workflow.

i. Environment Workflow

The purpose of this workflow is to ensure that the project environment is ready for the upcoming iteration. This includes process and tools. The following activities were carried out during the environment workflow (*Appendix D-1*).

- Prepared the environment for the iteration

  The researcher prepared and customized the tools to use within the iteration. The researcher verified that the tools have been correctly configured and installed. The researcher also prepared a set of project-specific templates and guideline to support the project development within the iteration. All the changes made to the project environment are ensuring properly communicated to the project team.

- Supported the environment during the iteration

  The researcher supported the developers in their use of tools and process during iteration. Supporting the project environment is an ongoing task to allow the project team to do their job efficiently without being slowed down due to issues with the development environment. This includes installation of required software, ensuring that the hardware is functioning properly and that potential network issues are resolved without delays.

ii. Project Management Workflow

In this workflow, we focused on refining the Software Development Plan, monitoring and controlling the OraCrypt™ project, and managing the risks. The following activities were carried out during the project management workflow (*Appendix D-2*).

- Refined the Software Development Plan

  The researcher refined the SDP, and mapping out a set of iterations using the prioritized use cases and associated risks. The project plans developed at this point are refined after each subsequent of iteration and become more accurate as iterations are completed.

- Refined the iteration plan, risks and architectural objectives

  The researcher constructed the iteration plan after the previous iteration was assessed and the project scope and risk reevaluated. The evaluation criteria for the architecture are outlined in the Software Architecture Document, taking into consideration the architectural risks that are to be mitigated. The system architecture assists with the planning. Therefore, we concerned on the consistent architecture and development guidelines to ensure it followed the project planning, and adherence to project standards.

- Monitored and controlled the project

  The researcher monitored the project in terms of active risks and objective measurements of progress and quality. The researcher regular reporting the project status, and any change requests are scheduled for the current or future iterations. The researcher also controlled and configured the changes during each of activities, iteration, and phases to ensure it followed the project plan.

- Closed-out the elaboration phase

  The researcher brings the phase to closure by ensuring that all major issues from the previous iteration are resolved, and the state of all artifacts is known.

iii. Software Requirements Workflow

In this workflow, all the requirements of OraCrypt™ product were managed and established to achieve the organization target. The following activities were carried out during the software requirements analysis workflow (*Appendix D-3*).

- Managed changing requirements

  The researcher assessed the impact of requested changes to the requirements, and managed the downstream impact of the changes approved to be action. The researcher evaluated requested changes and determined their impact on the existing requirement set. The researcher has to restructure the use case model based on the changes to requirements.

- Refined the system definition

  The researcher refined the requirements in order to capture the consensus understanding of the system definition by describing the use case flow of events in details, detailing the Supplementary Specifications, and developing a Software Requirements Specification. This work is done by reviewing the existing actor definitions and, then continues with detailing the use cases that have been previously outlined for each actor.

iv. Analysis And Design Workflow

Software analysis and design is conducted to transform the requirements to a design of the system-to-be. It evolves a robust architecture for the system. During this workflow, the design will be adapted to match the implementation environment for designing it for performance. The following activities were carried out during the analysis and design workflow (*Appendix D-4*).

- Defined candidate architecture

  The researcher allocated time to investigate potential candidate architecture. The researcher spent time early in the process to evaluate selecting technologies, and perhaps developing an initial prototype can reduce some major risks for the project. Early elimination of architectural in the project concerns of relationship between the architecture and organizational structures; separation of development concerns, which will provide a basis for separation work, and adherence to standards. The researcher defined the use-case realizations to be addressed in the current iteration

- Analyzed the behavior

  The researcher transformed the behavioral descriptions provided by the requirements into a set of elements upon which the design can be based. The researcher identified the analysis classes that satisfied the required behavior. And then, the researcher determined how these analysis classes fit into the logical architecture of the system.

- Designed the components

  The purpose of this activity is to refine the design of the system. The researcher refined the definitions of design elements by working out the details of how the design elements realize the behavior required of them. The researcher fleshed out the design details for the elements contained in the package by focusing on the recursive decomposition of functionality in the system in terms of classes. The use case realizations must be refined to reflect the evolving responsibilities of the design elements.

- Designed the database

  The researcher identified the design classes to be persisted in a database, and designed the appropriate database structures to store the persistent classes. The researcher defined the mechanisms and strategies for storing and retrieving persistent data in such a way that the performance criteria for the system are met.

The *Lifecycle Architecture* milestone marks the end of the elaboration phase. At this point, the researcher examined the detailed system objectives, scope, the choice of architecture, and the resolution of the major risks. The researcher has tested and evaluated the executable prototypes (R1.0 Release), and it signifies that the major risk elements have been addressed and credibly resolved. The deliverables that considered take into account during the elaboration phase are listed as following:

- Iteration Plan for Construction Phase
- Software Architecture Document
- Architectural Prototype

3.3.3 Construction Phase

The goal of this phase is to decide if the software is ready to be deployed. In this phase, the researcher managed the resources and controlled the operations to optimize the schedules and emphasized the quality of the product. To achieve the degree of parallelism in the development of OraCrypt^{TM} product, all components and software features are implemented and integrated into the software product. The following are the details of general activities for each workflow involved in iterations defined in the construction phase that are environment, project management, software requirements, and analysis and design workflows.

i. Environment Workflow

The purpose of this workflow (*Appendix D-1*) is to ensure the project environment is ready for the upcoming iteration. The researcher prepared and customized the tools to use within the iteration. The researcher verified that the tools have been correctly configured and installed. The researcher also prepared a set of project-specific templates and guideline to support the development of project within the iteration. All the changes made to the project environment are ensuring properly communicated to the project team.

ii. Project Management Workflow

In this workflow, the researcher focused on refining the Software Development Plan, monitoring and controlling the OraCrypt™ project, and managing the risks. The following activities were carried out during the project management workflow (*Appendix D-2*).

- Refined the SDP

  The researcher refined the SDP, and mapping out a set of iterations using the prioritized use cases and associated risks. The project plans developed at this point are refined after each subsequent of iteration and become more accurate as iterations are completed.

- Updated the iteration plan and risks

  The researcher updated the iteration plan after the previous iteration was assessed, and the project scope and risk reevaluated. The researcher updated the iteration plan based on the new functionality added during the new iteration. The researcher also concerned on the risks that need to be mitigated in the upcoming iteration.

- Monitored and controlled the project

  The researcher monitored the project in terms of risks and objective. The researcher regular reporting the project status, and any change requests are scheduled for the current and future iterations. The researcher also controlled and configured the changes during each of activities, iteration, and phases to ensure it followed the project plan.

- Closed-out the construction phase

  The researcher brings the phase to closure by ensuring that all major issues from the previous iteration are resolved, and the state of all artifacts is known.

iii. Software Requirements Workflow

In this workflow, the requirements of OraCrypt™ product were managed and established to achieve the organization goals. This is the process of deriving the system's requirements through observations of existing system. By doing this, the researcher determined the features that are absolutely necessary for the application. The following activities were carried out during the software requirements workflow (*Appendix D-3*).

- Updated the boundaries of the system

  During the third iteration, which is *C3 Iteration*, the researcher defined the new system boundaries that describe an envelope in which the solution system is contained. The researcher have defined two (2) new use cases that are represented in use case diagram as shows in *Appendix C*. The following are the description of each new use case involved.

  a. Use Case: Initialize System

     This use case shows the system initialization during the installation process of the OraCrypt™ product. It decomposed into two categories based on the user authorization, which are initialize system for Security Manager and authorized user. This process shall be done only once during system installation either on the server or client. Overall process shall be done by the system and only a few part need for user interaction.

  b. Use Case: Manage Report

     This use case shows the generating of system reports that can be categorized as encryption, decryption, key generation, user information, and system log report. User can view the report to get the better overview of the process selected.

- Managed changing requirements

  During the system development, the requirements frequently evolve. The researcher managed and controlled the requirements change include activities such as established a baseline, determined which dependencies are important to trace, and established traceability between related items.

iv. Analysis And Design Workflow

Once the necessary requirements have been collected, then the researcher proceeds to the next step in the model. This step is the analysis and design of the application. The design model represents the intent of the implementation, and is the primary input to the implementation workflow. The following activities were carried out during the analysis and design workflow (*Appendix D-4*).

- Analyzed the behavior

  The researcher transformed the behavioral descriptions provided by the requirements into a set of elements upon which the design can be based. The model elements are analyzed and refined by allocating responsibilities to specific elements (classes), and updated their relationships and attributes. New elements are added to support possible design and implementation constraints.

- Refined use case realization

  The researcher identified the analysis classes from the architecturally significant use cases. And then, the researcher updated the use-case realizations with analysis class interactions.

- Identified the design scheme

  When designing the application, it is important to be aware of the fact that the design process encompasses many aspects of the application. The researcher ensured that the design should correctly implement the specifications and requirements. The researcher broke down the application into user interface design and architectural design parts but the researcher rarely focus on the

design of the entire application. Dividing them into smaller modules make them easier to manage.

a. User Interface Design

The design for the user interface can be done independently. It is absolutely critical that the user interface is intuitive and easy to use. It should include the most basic and fundamental functions.

b. Architectural Design

This is the design of the overall architecture of the application. It consists of various sub-components. The integration of these sub-components together makes up the complete application.

- Designed the components

  The researcher decomposed the application into a set of interacting components. The researcher designed the packages, subsystems and components that contains of design elements in terms of functionality and classes. The use case realizations are refined to reflect the evolving responsibilities of the design elements.

- Designed the database

  The researcher identified the design classes to be persisted in a database, and designed the database structures to store the persistent classes. The researcher defined the mechanisms for storing and retrieving persistent data in such a way that the performance criteria for the system are met.

In the construction phase, the researcher divided the activities into three (3) iterations, which are iteration for developing R1 Beta Version, R1 Release, and R2 Release. The new use case is implemented during each of iteration. In the following subsection, the researcher details all activities involved in each iteration.

3.3.3.1 C1 Iteration

The *C1 Iteration* was implemented in developing R1 Beta Version. This iteration involved environment, project management, software requirement, analysis and design, and implementation workflow. In the implementation workflow, the researcher defined the organization of the code. All the classes and object will be implemented in terms of components and were make as simple as possible. The researcher will test the developed components as units. And then, the researcher integrated the results into an executable system. The following activities were carried out during the implementation workflow (*Appendix D-5*).

- Structured the Implementation Model

  The researcher structured the implementation model to ensure a smooth implementation and integration process for R1 Beta Version. The researcher is moving from the design to the implementation space started by mirroring the structure of the design model into the implementation model. The mapping from the design model to the implementation model may change as each implementation subsystem is allocated to a specific layer in the architecture. Structuring the implementation model generally results in a set of implementation subsystems that can be developed relatively independently.

- Planned the integration

  The purpose of this activity is to plan the integration of the system for the current iteration. Planning the integration is focused on which implementation subsystems should be implemented, and the order in which the implementation subsystems should be integrated in the current iteration. In the iteration plan, it specifies all use cases and scenarios that should be implemented in this iteration. So, the researcher identified which implementation subsystems participate in the use cases and scenarios for the current iteration. The researcher also identified which other implementation subsystems are needed and to make it possible to compile.

- Implemented the components

  The purpose of this activity is to complete a part of the implementation so that it can be delivered for integration. The researcher wrote source code, adapted existing source code, compiled, linked and performed unit tests, as they implement the elements in the design model. The researcher also fixed code defects and performed unit tests to verify any changes. Then the code is reviewed to evaluate quality and compliance with the Programming Guidelines.

- Integrated the components

  The researcher integrated changes from developer to create a new consistent version of an implementation subsystem. The researcher tested the components to ensure that the components perform to its specification. And then, the researcher delivered the implementation subsystem for component integration.

3.3.3.2 C2 Iteration

The *C2 Iteration* was implemented in developing R1 Release. This iteration involved environment, project management, software requirement, analysis and design, implementation, and testing workflow. In this section, the researcher details the activities involved in implementation and testing workflow.

i. Implementation Workflow

After the researcher has completed the designs for the application, the next step is to use that design scheme to implement the purpose of each design component. This workflow consists of writing, compiling, and executing source code. All the classes and object will be implemented in terms of components. The researcher tested the developed components as units. And then, the researcher integrated the results into an executable system. The following activities were carried out during the implementation workflow (*Appendix D-5*).

- Structured the Implementation Model

  The researcher structured the implementation model to ensure a smooth implementation and integration process of R1 Release. The researcher moved from the design to the implementation space by mirroring the structure of the design model into the implementation model to produce a set of subsystems that can be developed relatively independently.

- Planned the integration

  The researcher planned the integration of the system (R1 Release) for the current iteration. The integration plan specified all use cases and scenarios that should be implemented in this iteration. The researcher also identified other implementation subsystems are needed in developing R1 Release.

- Implemented the components

  The researcher wrote the source code, adapted the existing source code, and compiled it. The researcher also fixed the code defects and performed unit tests to verify any changes. Then, the code is reviewed to evaluate quality and compliance with the Programming Guidelines.

- Integrated the components

  The researcher tested the components to ensure that the components perform to its specification. Then, the tested components are delivered for component integration.

- Integrated the system

  The purpose of this activity is to integrate the implementation subsystems to create a new consistent version of the overall system (R1 Release). The researcher combined the components and subsystems into an executable build set and delivered them incrementally into the system integration workspace. Hence, the researcher integrated the system by implement the bottom-up based with respect to the layered structure, to ensure that the versions of the subsystems are consistent.

ii. Testing Workflow

This workflow acts as a service provider to the other workflows in many respects. The researcher focused on the evaluating and assessing product quality. The researcher validated and proved the assumptions made in the design and requirement specifications through concrete demonstration. The researcher proved that the software product works as designed and all the requirements are implemented appropriately. The following activities were carried out during the testing workflow (*Appendix D-6*).

- Defined evaluation mission

  The researcher identified the test effort for the iteration, and gained the agreement with user on goals and scopes that are direct to the test effort. Then, the researcher outlined the approach that will be used for the testing.

- Verified test approach

  The researcher demonstrated that the techniques outlined in the test approach would facilitate the planned test effort, produced accurate results, and is appropriate for the available resources. The researcher established the basic infrastructure to enable and support the test strategy by identified the scope, boundaries, limitations and constraints of each technique.

- Tested and evaluated

  The researcher achieved the test efforts that enable a sufficient evaluation of the items being targeted by the tests. The researcher provided ongoing evaluation and assessment of the target test, recorded the information necessary to diagnose, and resolved any identified issues. The researcher also provided the feedback on the most likely areas of potential quality risk.

- Achieved acceptable mission

  The researcher delivered a useful evaluation result to the user of the test effort. The researcher prioritized the set of tests that must be conducted, and the test results are evaluated against testing objectives to achieve the evaluation

mission. The researcher also advocated the resolution of important issues that have a significant negative impact on the evaluation mission.

3.3.3.3 C3 Iteration

The *C3 Iteration* was implemented in developing R2 Release. This iteration involved environment, project management, software requirement, analysis and design, implementation, and testing workflow. In this section, the researcher details the activities involved in implementation and testing workflow.

i. Implementation Workflow

This workflow consists of writing, compiling, and executing source code. In this workflow, the researcher wrote the code to allow the components to execute it task. The following activities were carried out during the implementation workflow (*Appendix D-5*) at the last iteration.

- Structured the Implementation Model

  The researcher structured the implementation model to ensure a smooth implementation and integration process of R2 Release. The researcher mirrored the structure of the design model into the implementation model. By structuring the implementation model, the researcher produced a set of implementation subsystems that can be developed relatively independently.

- Planned the integration

  The researcher planned the integration of the system (R2 Release) that specified all use cases and scenarios that should be implemented in this iteration. The researcher also identified other subsystems are needed in developing R2 Release. The results of this planning should be reflected in the SDP.

- Implemented the components

  The researcher wrote the source code, adapted the existing source code, compiled, linked it to allow the component to execute it task. The researcher also fixed the code defects and performed unit tests to verify any changes in the implementation model. Then, the code is reviewed to evaluate quality and compliance with the Programming Guidelines.

- Integrated the components

  The researcher tested the components to ensure that the components perform to its specification, and then the researcher released the tested implementation component for component integration.

- Integrated the system

  The purpose of this activity is to integrate the implementation subsystems to create a new consistent version of R2 Release. The researcher combined the components and subsystems into an executable build set and delivered them incrementally into the system integration workspace. In the process of resolved any merge conflicts, the researcher ensured that the versions of R2 Release are consistent and performed to its specification.

ii. Testing Workflow

After implementing the application, the researcher should have a program that is executable. Before the researcher rushed into publishing the product to the mass market, it is absolutely essential to subject the program to various testing. There are two (2) types of testing that the application can undergo. First type is debugging. Debugging simply means that the researcher subject the application to various testing scenarios to determine if there exists any kind of logic or programming errors. The second type is validation of the requirements. This means that the researcher created test scenarios with a different purpose in mind. This process is designed to determine whether the software application is meeting its requirements and services originally stated. There are evident in the testing that the services are indeed present and functional as well. This also determined

the inaccuracy of that functionality. The following activities were carried out during the testing workflow (*Appendix D-6*).

- Defined the evaluation mission

  The researcher setup the system-testing environment, identified the test effort, and gained the agreement with user on the goals and scopes that are direct to the test effort. The researcher outlined the approach that will be used for the testing.

- Verified the test approach

  The researcher demonstrated that the techniques outlined in the test approach would facilitate the planned test effort, produced accurate results, and is appropriate for the available resources. The researcher implemented the test infrastructure to verify that the test approach will work.

- Validated the build stability

  The researcher validated that the build is stable enough for the detailed test and evaluation effort to begin. The researcher referred this work as acceptance into testing that helps to prevent the test resources being wasted on a futile and fruitless testing effort.

- Tested and evaluated

  The researcher executed the test cases, provided ongoing evaluation and assessment of the target test, recorded the information necessary to diagnose, and resolved any identified issues. The researcher also provided the feedback on the most likely areas of potential quality risk.

- Achieve acceptable mission

  The researcher determined if the software source code satisfied system requirements allocated to software and software requirements specifications documents, based on testing result. The test results are evaluated against testing objectives to achieve the evaluation mission. The researcher advocated

the resolution of important issues that have a significant negative impact on the evaluation mission.

The *Initial Operational Capability* milestone marks the end of the construction phase. At this point, the product is ready to be handed over to the transition phase. All functionality has been developed. The researcher have tested and evaluated the executable prototypes, and it signifies that the major risk elements have been addressed and have been credibly resolved. In addition to the software, a user manual has been developed. The deliverables that considered take into account during the construction phase are listed as following.

- Iteration Plan for Transition Phase
- Software Design Description
- R1.0 (Beta Release)
- Release 1.0
- Release 2.0
- Test Cases

3.3.4 Transition Phase

The transition phase ensured that the software is available for its end users. The transition phase includes testing the product in preparation for release and making minor adjustments based on user feedback. At this point in the lifecycle, user feedback needs to focus mainly on fine-tuning the product, configuring, installing, and usability issues. This phase focuses on the required activities to lace the software into hands of the user. In this phase, the researcher defined an iteration called *T1 Iteration*.

3.3.4.1 T1 Iteration

T1 Iteration was implemented in deployment of R1 Release and R2 Release. This iteration involved project management, software requirement analysis, and

deployment workflow. In this subsection, the researcher details all activities involved in this iteration.

i. Project Management Workflow

In this workflow, the researcher monitored and controlled the OraCrypt™ project, and managed the risks occur in the development of this system. The following activities were carried out during the project management workflow (*Appendix D-2*).

- Monitored and controlled the project

  The researcher monitored the project in terms of active risks and objective measurements of progress and quality. The researcher regular reported the project status, and any change requests are scheduled for the particular iteration. The researcher also controlled and configured the changes during each of activities, iteration, and phases to ensure it followed the project plan.

- Closed-out the transition phase

  The researcher brings the phase to closure by ensuring that all major issues from the previous iteration are resolved, and the state of all artifacts is known. Any deployment problems are addressed.

- Closed-out the project

  The researcher ensured that the project is formally accepted. The researcher archived all project documentation and records. The researcher prepared the final status assessment, which if successful, the user formally accepts ownership of the software product. In the end, the researcher signed the agreement with user that all contracted deliveries have been made, meet the contracted requirements and are accepted into ownership by the user. The Project Manager then completed the closeout of the project by disposed of the remaining assets and reassigned the remaining staff.

ii. Software Requirements Workflow

In this workflow (*Appendix D-3*), all the system requirements were managed and established to achieve the project goals. Because of the requirements frequently evolve, the researcher managed and controlled the requirements change include activities such as established a baseline, determined which dependencies are important to trace, and established traceability between related items.

iii. Deployment Workflow

The deployment workflow describes the activities associated with ensuring that the software product is available for its end users. There are two (2) modes of product deployment that have been implemented in this project, which are custom install and shrink wrap product. There is an emphasis on testing the product at the development site, followed by beta testing before the product is finally released to the user. The following activities were carried out during the deployment workflow (*Appendix D-7*).

- Planned the deployment

  The researcher planned the product deployment that needs to take into account how and when the product will be available to the end user. Deployment planning requires a high degree of user collaboration and preparation. It is concerned with establishing the schedule and resources for development of end-user support material, acceptance testing, and production, packaging and distribution of software deployment units to ensure that end users can successfully use the delivered software product.

- Developed the support material

  The researcher produced the collateral needed to effectively deploy the product to the user. The researcher developed the support material that covers the full range of information required by the end-user to install, operate, use, and maintain the delivered system. It includes the training material for all of the various positions to effectively use the new system.

- Managed Beta testing

  This activity solicited feedback on the product from the users while it is still under active development. The beta test begins partway into the iteration, and may continue until the end of the iteration. The researcher runs the beta test and, in the case of shrink-wrap products, the researcher deal with the manufacturers to ensure that adequate quality is achieved in the product.

- Managed the acceptance test

  This activity ensured that the developed product fulfills its acceptance criteria at both the development, and target installation site and deemed acceptable to the user prior to its general release. The researcher organized the installation of the product on the test environment configurations that represents an environment acceptable to the user. Once installed, the researcher runs through a pre-selected set of tests and determined the test results. Then, the researcher reviewed the test results for anomalies.

- Packaged the product

  This activity described the necessary activities in creating a *shrink-wrapped* product. The idea is to take the deployment unit, installation scripts, and user manuals, and then packaged them for mass production.

- Managed the product baseline

  This activity ensured that all developed artifacts are captured and archived, at given points in time, as a basis for further product development. A baseline identifies one and only one version of an element. When the combined set of artifacts reached a specified level of maturity, the artifacts will be baselined to assist managing availability for release and reuse. Then, the artifacts are available for released and reused in the subsequent project iterations or other projects.

This final iteration in the transition phase culminates in the delivery to the user of a complete system with functionality and performance as specified, and demonstrated in

acceptance testing. The user takes ownership of the software after a successful acceptance test. The following are the artifact that will be concerned during the transition phase.

- Software User Manual
- Installation Instructions
- Release 2.0

3.4 Software Tools

There are a wide range of OO methodologies, which have different views of development process. To achieve the target in implementation of RUP software process and OO methodology, Unified Modeling Language (UML) is used throughout the development of OraCrypt™ product for specifying, visualizing, constructing, and documenting the deliverables of software product. It enables all users that are involved in software development, documentation, and described the software in standard way.

UML is a process independent. It is not tied to any particular software development life cycle. This modeling technique is refers to the use of notation to represent both the object-oriented analysis and object-oriented design model. Notations play an important part in methodology because in order to ensure UML can provide the most benefit, software process must also be implemented during the software development. The benefits of UML are listed as following.

- UML is easy enough to use and provides clear thought.
- UML is expressive enough to express the design aspects of software.
- It unambiguous features helps to solve the misunderstandings.
- Supported by suitable tools.

3.5 Problem Solving Methodology

Software development always running into problems when the development is carried out without a proper software process follows the defined methodology. However, OO paradigm can be better use to overcome the problems that describe below.

- Conformity

  Software should work when it is required to fit together with the existing system. This includes software and hardware interfaces that need to be interface with the software. This conformity can be achieved through the use of OO approach where each of software components can be modeled clearly.

- Changeability

  It is inevitably occur that user always demand major changes to the software. Therefore, the changes are easier to be tackled and organized through the use of OO methodology because of its key features facilitate reusability and maintainability aspects.

- Complexity

  Software is undeniably a complex structure made by human beings. Huge software may leads to a problem in an attempt to understand the software in it is entirely. It is also affects the management of the process. Therefore, the complexity can be better managed by adopting an OO approach, which can help reduce the complexity.

- Invisibility

  A problem with the essence of software is that invisible and invisualizable. This problem can be overcome by using an OO approach to visualize the software model and components by using the UML diagram. Therefore, it provides the better means of communication between the user and researcher.

CHAPTER 4

PROJECT DISCUSSION

4.1 Introduction

The OraCrypt™ development process describes the life cycle model for database security projects. This project followed the unified process model, which is a generic process framework that can be specialized for database security software systems. In this chapter, the researcher discusses the results obtained from the development of OraCrypt™ product. It focuses on the analysis of the system development and other deliverables such as documentation produced. Besides that, the constraints faced during the software development and some recommendations in improving the software development will be covered.

4.1.1 Project Phases And Milestones

The development of the OraCrypt™ project was conducted using a phase approach where multiple iterations occur within a phase. *Table 4.1* describes each phase and the major milestone that marks the completion of the phase.

Table 4.1 Project Phases and Major Milestones

Phase	Description	Milestone
Inception Phase	The inception phase developed the product requirements for the OraCrypt™ project. The major use cases were developed as well as the high level Software Development Plan (SDP). At the completion of the inception phase, the researcher decided to proceed with the project based upon the estimates time and budget.	*Business Case Review Milestone* at the end of the phase marks the Go/No decision for the project budget and time as planned.
Elaboration Phase	The elaboration phase defined and baselined the Software Architecture Document (SAD). This phase analyzed the requirements and developed the architectural prototype (based on the architecturally-significant use cases). At the completion of the elaboration phase, all use cases selected for Release 1.0 will have completed analysis and design.	The *Architectural Prototype Milestone* marks the end of the elaboration phase. This prototype signifies verification of the major architectural components that comprise the R1.0 Release.

Table 4.1 Project Phases and Major Milestones (continue)

Phase	Description	Milestone
	In addition, the high-risk use cases for Release 2.0 have been analyzed and designed. The architectural prototype was tested the feasibility and performance of the architecture that is required for Release 1.0.	
Construction Phase	During the construction phase, remaining use cases were analyzed and designed. The Beta version for Release 1.0 was developed and distributed for evaluation. The implementation and test activities to support the R1.0 and R2.0 releases was completed.	The *R2.0 Operational Capability Milestone* marks the end of the construction phase. Release 2.0 Software is ready for packaging.
Transition Phase	The transition phase prepared the R1.0 and R2.0 releases for distribution. It provided the required support to ensure a smooth installation including user training.	The *R2.0 Release Milestone* marks the end of the transition phase. At this point all capabilities are installed and available for the users.

The following table describes the iterations along with associated milestones and addressed risks.

Table 4.2 Project Iteration with the Milestones and Risks

Phase	Iteration	Description	Associated Milestones	Risks Addressed
Inception Phase	Preliminary Iteration	Defined product requirements, and Software Development Plan.	Business Case Review	Clarified user requirements up front. Developed realistic Software Development Plans and scope. Determined feasibility of project from a business point of view.
Elaboration Phase	E1 Iteration – Develop Architectural Prototype	Completes analysis and design for all R1 use cases. Developed the architectural prototype for R1. Completed analysis & design for all high-risk R2 use cases.	Architectural Prototype	Architectural issues clarified. Technical risks mitigated. Early prototype for user review.

Table 4.2: Project Iteration with the Milestones and Risks (continue)

Phase	Iteration	Description	Associated Milestones	Risks Addressed
Construction Phase	C1 Iteration – Develop R1 Beta	Implemented and tested R1 use cases to provide the R1 Beta Version.	R1 Beta	All key features from a user and architectural prospective implemented in the Beta. User feedback prior to release of R1.
	C2 Iteration – Develop R1 Release	Implemented and tested remaining R1 use cases, fixed defects from Beta, and incorporated feedback from Beta. Developed the R1 system.	R1 Software	R1 fully reviewed by user community. Product quality should be high. Defects minimized. Cost of quality reduced.
	C3 Iteration – Develop R2 Release	Designed, implemented, and tested R2 use cases. Incorporated enhancements and defects from R1. Develops the R2 system.	R2 Software	Quick release of R2 addresses customer satisfaction. All key functionality provided in System by R2 Release.
Transition phase	T1 Iteration – R1 Release	Packaged, distributed, and installed R1 Release.	R1 Release	Two-stage release minimizes defects. Two-stage release provides easier transition for users.
	T2 Iteration – R2 Release	Package, distribute, and install R2 Release.	R2 Release	Two-stage release minimizes defects. Two-stage release provides easier transition for users.

4.1.2 Project Deliverable(s)

Besides the OraCrypt™ product, there are several deliverables, which were completed during the development of OraCrypt™ product. The deliverable items are:

- Project Glossary
- Iteration Plan
- Software Development Plan

- Software Architecture Document
- Software Requirements Specification
- Use Case Specification
- Supplementary Specification
- Software Design Description
- Software User Manual

4.2 Project Constraint(s)

During the development of OraCrypt™ product, the researcher faced several problems and constraints in carry out the task given. The researcher had listed the difficulties as following:

i. Time constraints

In the OraCrypt™ product development, it is important for researcher to study in details on technology and concepts that is being used. For the limited time available to carry out the project, the researcher has to consider in time factor and effort so that the system developed within the schedule and achieve all the system objectives.

ii. Lack of experience and knowledge

As the researcher is new in the cryptography technology and database security development, this contributes to small problem in OraCrypt™ product development. The researcher had to take much time for research in development techniques, instead of the time can be used to focus on upgrade the system performance and reliability.

iii. Absence of previous product documentation

The OraCrypt™ product involved specialized cryptography technology and focus on database security application. This is one major constraint because no software

document can be used as a reference. As a result, OraCrypt™ product involved had a difficulty in understand the software requirements.

4.3 Project Advantages / Uniqueness

The product is designed with the capability to be incorporated into existing database application system. Therefore, the benefits to the industry namely to all users of Oracle RDBMS without regards to the application system they are using, will be it allows the users to use their existing application system with added capability i.e. encryption and decryption of data.

There are a few competing products namely DBEncrypt and Secure.Data from the USA. The evaluation and comparison with them showed that OraCrypt™ can encrypt and decrypt data at three levels i.e. data item, row and column. None of the other two can and theirs limited to whole column encryption. One other advantage of OraCrypt™ product is that it does not store any crypto keys in the DBBMS. Instead all our crypto keys are external to the security system and will be fetched from secure smartcard systems.

In terms of efficiency, the norm in cryptography is that the data size will increase almost double after encryption. This happens to OraCrypt™ also but the researcher had designed a new 64-bit storage format that will reduced the storage need from double to about 1.33 percent only.

4.4 Project Innovation

This product is a homemade product i.e. a product from the outcome of a PhD research done in Malaysia. The design of the encryption system is an original research and utilized indigenous cryptographic algorithm that was also a product from previous

PhD research effort. Therefore, there is no question on the originality of concept and its implementation. It was successfully implemented in Oracle 8i RDBMS using quite a substantial amount of data obtained from a library information system. There are many outstanding features and is summarized as follows:

i. Able to encrypt and decrypt data at all level i.e. data element, row and column.
ii. Data can be wholly or selectively encrypted at those three levels.
iii. No crypto keys are to be found anywhere in the RDBMS.
iv. All crypto keys are securely stored in smartcards that are using proprietary smartcard operating system.

4.5 Project Potential

The target markets are all users of RDBMS with or without application system embedded in their DBMS. This includes the military, police, governments, banking and finance, and the private sector at large. This product has international potential especially with import restriction on encryption algorithms and security products like from the USA. With the current development of the product, there are great potential to create employment for its marketing, support and training services since the product is aimed at international market.

One important strategy to be ahead of competitors especially in the international arena is that the researcher designed and developed the product based on sound software engineering practices that allows for easy maintenance, upgrades and adaptability to many types of RDBMS.

4.6 Project Contribution

The researcher had discussed the issues of multilevel database encryption, encryption in relational database management systems, direct and hierarchical access controls to encrypted databases, cryptographic key management, encryption schemes, and database security in general. Through the development of this project, the researcher had made numerous progress and achievements particularly in the design of a multilevel encryption scheme and issues relevant to it. In the following paragraphs, the researcher present the views of the advancements made thus far and its contributions towards the body of knowledge mentioned earlier.

- A new encryption scheme

  The researcher employed and implemented various concepts and tools from the fields of cryptography, data security, and encryption algorithm design. The new database encryption scheme is able to encrypt data at all levels of the relational database model, not only at the row and column levels but also at the finest level of granularity that is at the data element level. A most significant characteristic about the new scheme is its high security feature contributed by the fact that no cryptographic keys are stored in the scheme, and no security information needed to be stored anywhere in the database as has been the tradition.

- A new approach in accessing and processing of encrypted data

  The new approach controlled access and processing of encrypted data uses Initialization Vectors (IV) for data. In this approach, including the use of IV in key generation, data IV are never secret and do not have any security significance. Because of this, the researcher is able to store them together with the encrypted data. In this scheme, once data are encrypted in the database their corresponding data IV are captured for later decryption and retrievals. In short, this approach is a new solution to the problem of access controls in a multilevel secure RDBMS and it solves the problem of discretionary and mandatory access controls.

- A new key management technique

  The new key management technique allows the generation, management, and distribution of system and user keys in a very secure fashion. For higher security requirements, key management is better done off-line, in a stand-alone mode, and stored securely in tamper proof smartcards. The key management technique based on user IV results in a hierarchical access control to encrypted data. Discretionary and direct access to encrypted information is made possible with the production of shared keys.

- A new and innovative stream cipher design

  Database applications are normally involved in online transaction processing besides the processing of huge amount of data. With the above experience and knowledge, the researcher was able to make a major modification to the basic design of TSSC stream cipher. Because of this great modification, the researcher considered it a new TSSC. The major modifications made were to its basic design in using four (4) linear feedback shift registers instead of eight. The combination of these four (4) registers produces a total key length of 128 bits instead of 189 bits (Part I algorithm) or 349 bits (Part 2 algorithm). The new TSSC algorithm is a very much faster and is yet secure cipher system for the processing of large amount of data in databases at greater speed.

- A local database encryption package

  This invention is a local database security product, with the implementation of local made cryptographic algorithm. This research contributes to the development of local software industry in the fields of database programming, software engineering, and information security in general.

- An independent and secure database security mechanism

  Findings from this research have allowed the building of an independent and secure solution for commercial RDBMS including Oracle. This security mechanism can be added to the existing security features offered by a DBMS as an independent module. With proper design considerations, the security mechanism can be easily

implemented in applications to become as part of its application security procedure and transparent to users.

4.7 Future Work

This research has opened up a new avenue for further research in the role cryptography can play in database security, and in the wider scope of Information Technology Security (ITSEC). Research in database encryption schemes also has evolved beginning from pure cryptographic study for database security to the integration of modern cryptographic technology into database management systems thus providing the required cryptographic supports. The following list provides some insights into the possible future work that may be carried out in this area. The list however is not meant to be exhaustive.

- Security of the scheme

  In the scheme's implementation, user keys are communicated between the smartcard and the encryption module. Creating a secure channel between the client (smartcard reader) and the back-end database machines will render a more secure system. The current implementation of the scheme also assumed the underlying operating system and database system to be at a certain level of security.

- Speed and efficiency of the scheme

  The redesign of the cipher algorithms has helped to increase the speed in the encryption process. Another major factor in implementation is to ensure that the encryption scheme is implemented efficiently and effectively. Further research should be conducted to find out the best implementation methodology to gain maximum performance from the scheme.

- Scheme Implementations

  The current scheme implementation is specific to Oracle8i RDBMS running on Windows 2000 server in a stand-alone single-user mode. Future work can be conducted to implement the scheme on different database platforms running on different operating systems and using different hardware platforms. Besides this, the scheme can be implemented in a networking environment and possibly in a multi-user mode. Testing for effectiveness and efficiency in Internet access to the secure RDBMS should also be conducted.

- Security of the cryptographic keys

  All cryptographic keys are stored in smartcards acquired from the market place. To be more secure, our own smartcard technology needs to be developed to be independent, full control, and free from possible Trojan horses. Further study on the full life cycle of a key definitely can increase the security including the methods for key storage, distribution, and destruction. Proper procedures on issues related to cryptographic keys need to be resolved and standardized.

- Issues in encrypted data

  A better way of storing encrypted data than in Base64 format as proposed in the research should be seek. Most encryption schemes evaluated by the researcher were using the hexadecimal format for their cryptograms. More research should be conducted in the processing of encrypted data including query processing that allows a wider range of operations to be performed on the cryptograms.

- Adaptation of software engineering practices

  To adapt the software engineering practices on working environment, it is suggested that some study has to be done on the software process concepts to find the best solution for the project development in future. Even training can be conducted as to get the benefits of software engineering practices.

CHAPTER 5

CONCLUSION

5.1 Introduction

This chapter summarized the whole research work done in this project emphasized on the project perspective and software engineering perspective. These include a review of the objective and purpose of the research as whole, the methodology used, various decision made, lesson learnt, experience accumulated, the limitations, and most important of all its contributions.

5.2 Conclusion

This research endeavor is related to the design, build and implement a multilevel security scheme for database management systems. The main research theme covers the protection of information processed by a computer against 268 unauthorized observation, unauthorized or improper modification, and the denial of service. The protection scheme is a simple but high assurance with tasks given to central security personnel who focused on data and users. The security scheme is placed partly in the operating system and database application layers. The bulk of the scheme is integrated into the selected RDBMS. There are four main objectives to the research in solving the major database security problem for multilevel databases.

The four main objectives achieved by the research are to design a new multilevel security scheme for database management systems, to design a key management system, to implement both the security scheme and the key management systems, and finally to benchmark the security and performance of the security scheme as a whole. It was observed that the design of the new security scheme together with its key management subsystem was successfully tested and implemented in a commercial database management system. The results and findings derived from this research showed very promising with respect to enhancing the security of databases without affecting the efficiency of the DBMS. The new cryptosystem can be well adapted in many database application systems to benefit local authorities and business entities. With due considerations, the research effort can be transformed into a commercial product providing security solutions to organizations in protecting their precious information in databases.

REFFERENCES

Abbey, M., Corey, M. J., and Abramson, I. (1999). *Oracle8i: A Beginner's Guide.* Oracle Press Edition, Berkeley, C.A.: Osborne McGraw-Hill.

Abrams, M.D., Jajodia, S. and Podell, H.J. eds. (1995). *Information Security: An Integrated Collection of Essays.* Los Alamitos, California: IEEE Computer Society Press.

Al-Salqan, Y.Y., Jagannathan, V.J., Davis, T., Zhang, N. and Keddy, Y.V.R. (1996). Security and Confidentiality in Health Care Informatics. *Proceedings of the First ACM Workshop on Role-based Access Control.*

Assoc Prof. Zailani Mohamed Sidek. (2003). Universiti Teknologi Malaysia. *The Development of a Commercially Viable Database Encryption Tool for Oracle8i RDBMS*. PhD Thesis in Computer Science.

Barry W. Boehmn. (1996). IEEE Software. *Anchoring the Software Process*. July 4, 13. 73-82.

Barry W. Boehmn. (1988). TRW Defense Systems Group. *A Spiral Model of Software Development and Enhancement*. May 5. 61-72

Bertino, E., Jajodia, S., and Samarati, P. (1995). *Database Security: Research and Practice*. Information Systems. 20(7): 537-56.

Blakley, B. (1996). The Emporer's Old Armor. *ACM New Security Paradigm Workshop*. Lake Arrowhead, CA, 2-1.

Booch, G., Jacobson, I., and Rumbaugh, J. (1999). *The Unified Software Development Process.* Addison-Wesley.

Booch, G., Rumbaugh, J., and Jacobson, I. (1998). *The Unified Modeling Language User Guide*. Addison-Wesley.

Castano, S., Fugini, M. G., Martella, G., and Samarati, P. (1995). *Database Security*. Wokingham, England: Addison-Wesley Publishing Company.

Ciechanowicz, C. (2001). *Database Security.* Royal Holloway University of London: Lecture Notes. Not Published.

Dastjerdi, A. B., Pieprzyk, J., and Naini, R. S. (1996). *Security in Database: A Survey Study*. Communications of the ACM. 44(2): 38-44.

Denning, D. E. (1983). Field Encryption and Authentication in Advances in Cryptology. *Proceedings of CRYPTO 83 (D. Chaum, ed.), (Santa Barbara, CA).* Plenum Press, New York. 231-247.

Humphrey, Watt (1990). *Managing The Software Process*. Addison-Wesley.

Jacobson, J., Griss, M., and Jonsson, P. (1997). *Software Reuse – Architecture, Process and Organization for Business Success*. Harlow, England: Addison Wesley Longman.

Jacobson, I., Christerson, M., Jonsson, P., and Overgaard, G. (1992). *Object-Oriented Software Engineering - A Use Case Driven Approach.* Wokingham, England: Addison-Wesley. 582.

Kruchten, P. (2000). *The Rational Unified Process—An Introduction*. 2nd ed. Addison Wesley Longman.

Kruchten, P. (1995). IEEE Software. *The 4+1 View Model of Architecture*. 12 (6): November. IEEE, 42-50.

Menezes, A. J., Oorschot, P. C. V., and Vanstone, S. A. (1997). *Handbook of Applied Cryptograph.* Boca Raton, U.S.A.: CRC Press.

Michael, A., Michael J. C., and Abramson, I. (1999). *Oracle 8i, A Beginner's Guide*. Osborne McGraw-Hill.

Pfleeger, S. L. (2000). *Software Engineering Theory and Practice*. 2nd ed. Prentice Hall.

Pernul, G. (1994). *Information Systems Security: Scope, State-of-the-art, and Evaluation of Techniques*. Int. Journal of Information Management, Butterworth-Heinemann. 15(3): 105-121.

Rumbaugh, J.; Booch, G.; Jacobson, I. (1999). *UML Reference Manual*. Addison Wesley Longman.

Roger, S. P. (2001). *Software Engineering A Practitioner's Approach*. 5th ed. McGraw-Hill International Edition.

Royce, W. (1998). *Software Project Management: A Unified Framework.* Addison Wesley Longman.

Sandhu, R. and Jajodia, S. (1990). Integrity mechanisms in database management systems. *Proc. 13th National Computer Security Conference,* 526-540.

Sandhu, R. and Samarati, P. (1996). Authentication, Access Control and Audit. *ACM Computing Surveys.* 28(1): 241-243.

Schneier, B. (1998). IEEE Computer. *Cryptographic Design Vulnerabilities.* 29-33.

Seberry, J. and Pieprzyk, J. (1989). *Cryptography: An Introduction to Computer Security.* Victoria, Australia: Prentice Hall of Australia Pty. Ltd.

Smith, G. W. (1989). Multilevel Secure Database Design: A Practical Application. *Proc. 5th IEEE Annual Computer Security Application Conference*. IEEE

Computer Society Press. 314-321.

Theriault, M. and Heney, W. (1998). *Oracle Security. 1st. ed.* Sebastopal, CA: O'Reilly & Associates.

Tuan Sabri bin Tuan Mat @ Tuan Muhammad. (2000). *Design of New Block and Stream Cipher Encryption Algorithms for Data Security.* Universiti Teknologi Malaysia: Ph.D. Thesis.

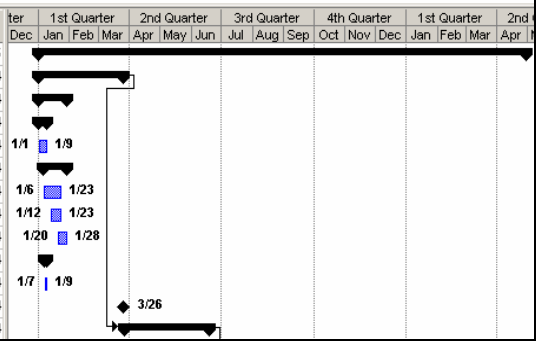
Universiti Teknologi Malaysia. (2003). *Panduan Menulis Tesis*. Universiti Teknologi Malaysia.

APPENDICES

APPENDIX A

OraCrypt Development Plan

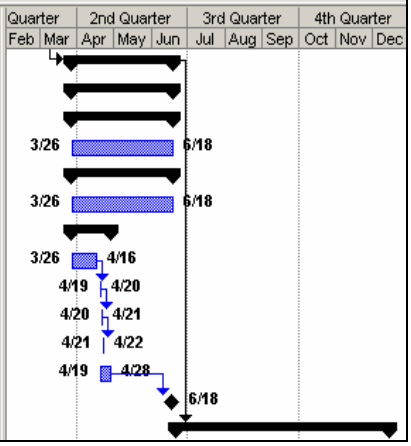
Task Name	Duration	Start	Finish
OraCrypt Development Plan	**261.25 days**	**Thu 1/1/04**	**Sat 4/30/05**
Inception Phase	**45.75 days**	**Thu 1/1/04**	**Fri 3/26/04**
Iteration I1 - Preliminary Iteration	**15 days**	**Thu 1/1/04**	**Wed 1/28/04**
Project Management	**5 days**	**Thu 1/1/04**	**Fri 1/9/04**
Develop Software Development Plan	5 days	Thu 1/1/04	Fri 1/9/04
Requirement Management	**12.25 days**	**Tue 1/6/04**	**Wed 1/28/04**
Analyze the Problem	10 days	Tue 1/6/04	Fri 1/23/04
Define the System	7 days	Mon 1/12/04	Fri 1/23/04
Manage Scope of the System	5 days	Tue 1/20/04	Wed 1/28/04
Environment	**2 days**	**Wed 1/7/04**	**Fri 1/9/04**
Prepare Environment of the Project	2 days	Wed 1/7/04	Fri 1/9/04
Business Case Review Milestone	0 days	Fri 3/26/04	Fri 3/26/04
Elaboration Phase	**45.75 days**	**Fri 3/26/04**	**Fri 6/18/04**



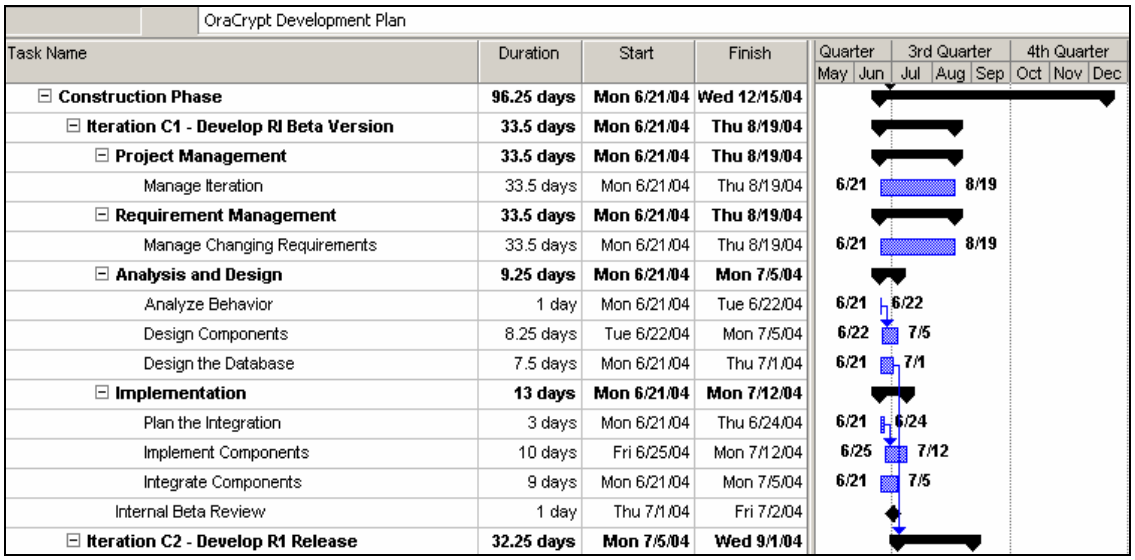
Appendix A-1 Project Schedule for Inception Phase

OraCrypt Development Plan

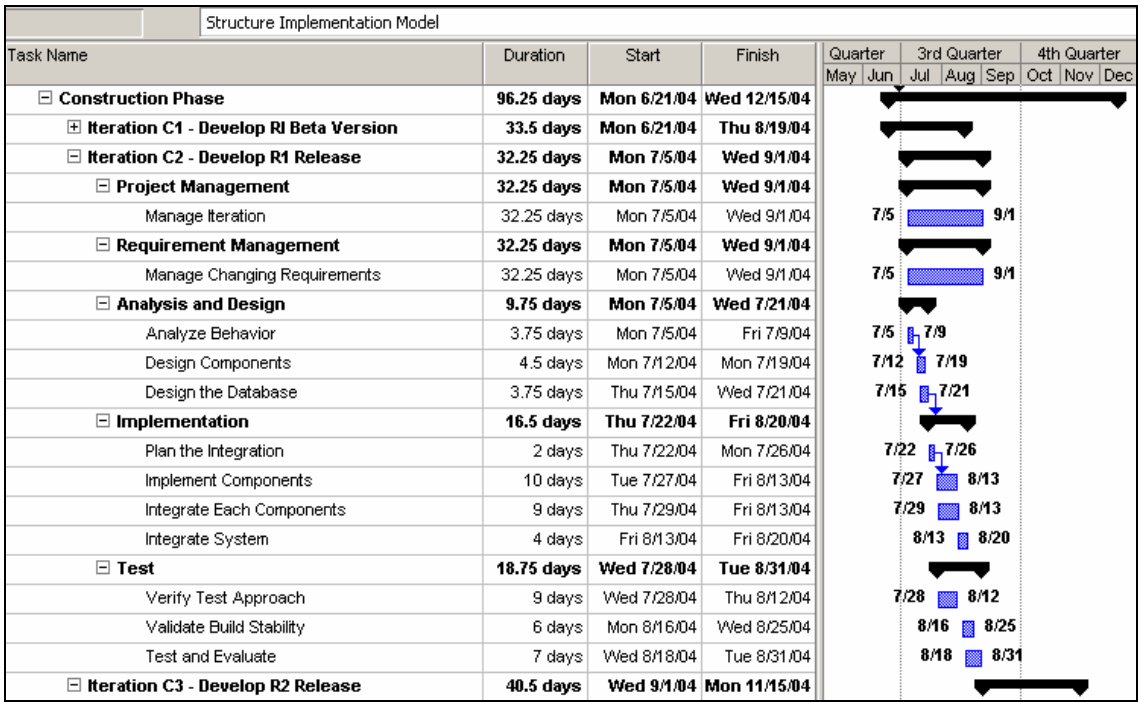
Task Name	Duration	Start	Finish
Elaboration Phase	**45.75 days**	**Fri 3/26/04**	**Fri 6/18/04**
Iteration E1 - Develop Architectural Prototype	**45.75 days**	**Fri 3/26/04**	**Fri 6/18/04**
Project Management	**45.75 days**	**Fri 3/26/04**	**Fri 6/18/04**
Manage Iteration	45.75 days	Fri 3/26/04	Fri 6/18/04
Requirement Management	**45.75 days**	**Fri 3/26/04**	**Fri 6/18/04**
Manage Changing Requirements	45.75 days	Fri 3/26/04	Fri 6/18/04
Analysis and Design	**18 days**	**Fri 3/26/04**	**Wed 4/28/04**
Define Candidate Architecture	12 days	Fri 3/26/04	Fri 4/16/04
Analyze Behavior	1 day	Mon 4/19/04	Tue 4/20/04
Design Components	1 day	Tue 4/20/04	Wed 4/21/04
Design the Database	1 day	Wed 4/21/04	Thu 4/22/04
Structure Implementation Model	6 days	Mon 4/19/04	Wed 4/28/04
Lifecycle Architecture (LCA) Milestone	0 days	Fri 6/18/04	Fri 6/18/04
Construction Phase	**96.25 days**	**Mon 6/21/04**	**Wed 12/15/04**



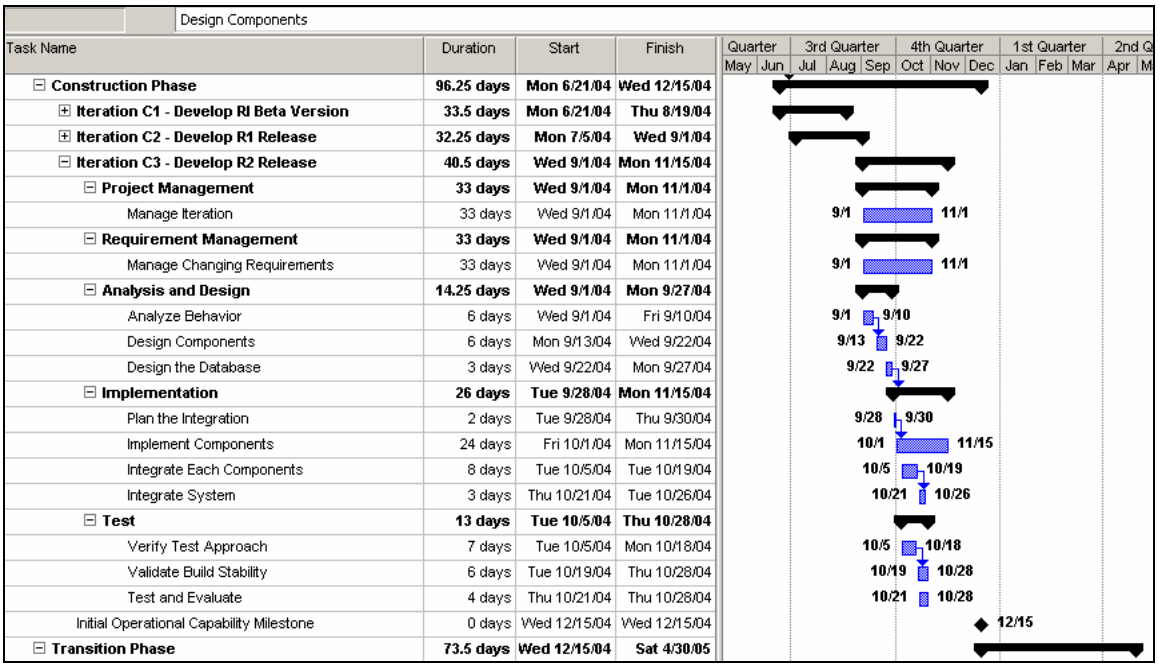
Appendix A-2 Project Schedule for Elaboration Phase



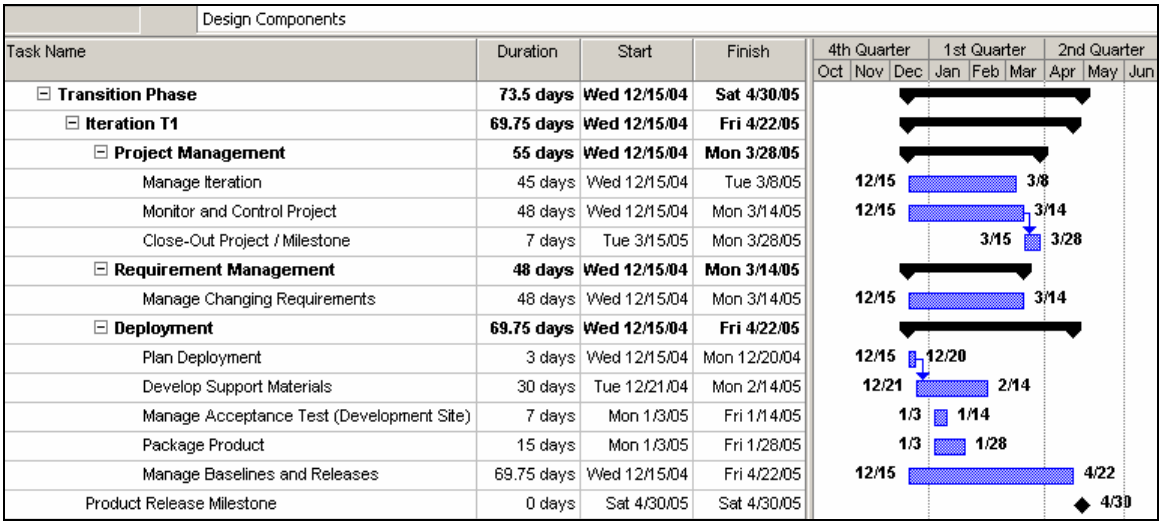
Appendix A-3 Project Schedule for Construction Phase – Iteration C1



Appendix A-4 Project Schedule for Construction Phase – Iteration C2



Appendix A-5 Project Schedule for Construction Phase – Iteration C3



Appendix A-6 Project Schedule for Transition Phase

End of Project Report Guidelines

A. Purpose

The purpose of the End of Project is to allow the IRPA Panels and their supporting group of experts to assess the results of research projects and the technology transfer actions to be taken.

B. Information Required

The following Information is required in the End of Project Report :

- Project summary for the Annual MPKSN Report;
- Extent of achievement of the original project objectives;
- Technology transfer and commercialisation approach;
- Benefits of the project, particularly project outputs and organisational outcomes; and
- Assessment of the project team, research approach, project schedule and project costs.

C. Responsibility

The End of Project Report should be completed by the Project Leader of the IRPA-funded project.

D. Timing

The End of Project Report should be submitted within three months of the completion of the research project.

E. Submission Procedure

One copy of the End of Project is to be mailed to :

IRPA Secretariat
Ministry of Science, Technology and the Environment
14th Floor, Wisma Sime Darby
Jalan Raja Laut
55662 Kuala Lumpur

End of Project Report

A. Project number : 74228

Project title : The Development of A Commercially Viable Database Encryption Tool for Oracle8i RDBMS

Project leader: Mohd Nazri Bin Kama

Tel: 03-26154379 **Fax: 03-26930933**

B. **Summary for the MPKSN Report (**for publication in the Annual MPKSN Report, please summarise the project objectives, significant results achieved, research approach and team structure)

This project investigates the application of state-of-the-art cryptography for the security of commercially available relational database systems which is Oracle8i RDBMS.

This project produced a new commercial database encryption tool that solves the problems of secure access to databases both in hierarchical manner and/or direct access based on some form of classifications.

Encrypted data in the DBMS are accessible only to those having the appropriate cryptographic keys securely stored in smartcards.

The software development methodology was backed by the experts from the Centre for Advanced Software Engineering (CASE), UTM. This project used Thales Software Engineering Methodology (Soft-EM) which renders the system ease of maintenance and adaptability. We are now porting the system to SQL*Server 2000 and IBM-DB2 fairly easily.

This project team consists of four (4) staff which are (1) Project Advisor, (1) Project Leader, and (2) System Analysts working closely with each other in research and development activities.

C. Objectives achievement

- **Original project objectives** (Please state the specific project objectives as described in Section ll of the Application Form)

  i. To fully develop a cryptographic database security mechanism for Oracle8i RDBMS.
  ii. To implement the new design for IV-based multilevel database encryption scheme in Oracle8i rdbms environment.
  iii. To fully develop and implement a cryptographic key generation, management and distribution system.
  iv. To benchmark the database security tool that is developed.

- **Objectives Achieved** (Please state the extent to which the project objectives were achieved)

  i. Succesfully produced a software known as OraCrypt
     - Able to encrypt-decrypt data in
       - Oracle8i
       - Oracle9i (additional achievement)
     - Able to manage key generation, management and distribution

- **Objectives not achieved** (Please identify the objectives that were not achieved and give reasons)

Objective no (iv): Benchmark the database security tool that is developed.
This project has not enough time and budget to implement this objective.

D. Technology Transfer/Commercialisation Approach (Please describe the approach planned to transfer/commercialise the results of the project)

1. The database security tool developed can be used by organizations using Oracle8i databases as an "off-the-shelf" package.
2. Training and support by local expertise in the implementation and use of the package.
3. Information dissemination via seminars and conferences.
4. Users will be utilizing an independent database security system.

Because the product is an "off-the-shelf" package, the installation and training on the use of the product will make it sustainable in the long run.

E. Benefits of the Project (Please identify the actual benefits arising from the project as defined in Section lll of the Application Form. For examples of outputs, organisational outcomes and sectoral/national impacts, please refer to Section lll of the Guidelines for the Application of R&D Funding under IRPA)

- **Outputs of the project and potential beneficiaries** (Please describe as specifically as possible the outputs achieved and provide an assessment of their significance to users)

  Output from the project:
  1. An initial version of a commercial database security product based on local expertise and technology.
  2. An implementation of the newly designed database encryption scheme for Oracle8i RDBMS systems.
  3. A subsystem for the generation, management and distribution of cryptographic keys.
  4. A benchmarking technique for the performance and evaluation of database security products.

  Potential beneficiaries:
  1. Security of public databases like JPJ, Inland Revenue, Registration Department, etc.
  2. Financial and Banking databases - require high security for their data and information.
  3. Malaysian Arm Forces and Police databases – safeguard national security.
  4. Private organizations and individuals – personal privacy and secrecy.

- **Organisational Outcomes** (Please describe as specifically as possible the organisational benefits arising from the project and provide an assessment of their significance)

  1. M.Sc. candidate – 2 persons
  2. Producing local software expertise in the security of Oracle databases.
  3. The development of a new tool in database security.

- **National Impacts** (If known at this point in time, please describes specifically as possible the potential sectoral/national benefits arising from the project and provide an assessment of their significance)

  Impacts to organizational linkages:
  1. In Malaysia, close cooperations will be generated among government agencies like database users, research departments like in MINDEF and USM, and stimulate the participation of local researchers.
  2. Internationally, presentations in seminars and conferences results in close collaborations and works, like with the famous Royal Holloway center for security studies.

F. Assessment of project structure

- **Project Team** (Please provide an assessment of how the project team performed and highlight any significant departures from plan in either structure or actual man-days utilised)

This project involves 4 members who are:

- 1 Project Advisor
- 1 Project Leader
- 2 Research Officers

The project recruited 2 ROs in order to speed up the project's development process. However, due to the problem in the management of RMC Skudai, both ROs were terminated at the third phase of the project. Project Leader continued the tasks till the end of the project.

Collaborations (Please describe the nature of collaborations with other research organisations and/or industry)

No

G. Assessment of Research Approach (Please highlight the main steps actually performed and indicate any major departure from the planned approach or any major difficulty encountered)

This project was managed in 4 main phases:
Inception Phase - develop the product requirements for the project
Elaboration Phase - define and baseline the Architectural prototype
Construction Phase – develop the actual product
Transition Phase – test and deploy the product in user community

There is no significant change in the research approach even though 2 of the team members were terminated in the middle of the project timeline.

H. Assessment of the Project Schedule (Please make any relevant comment regarding the actual duration of the project and highlight any significant variation from plan)

Significant variation can be seen when the project was at the constuction phase. Actual plan was:
Inception Phase – 2 monts
Elaboration Phase – 5 months
Construction Phase – 7 Months
Transition Phase – 4 months

Due to the Budget constraints, project had to drag the timeframe from 7 months to 9 months.

I. Assessment of Project Costs (Please comment on the appropriateness of the original budget and highlight any major departure from the planned budget)

The original budget was spent appropriately throughout the project for the purchase of hardware, fees of ROs as well as the administration of the project. There is no change from the planned budget. However, a slight adjustment was made in the cost structure to recruit 2 ROs instead of only one RO as per planned.

J. Additional Project Funding Obtained (In case of involvement of other funding sources, please indicate the source and total funding provided)

No..

K. Other Remarks (Please include any other comment which you feel is relevant for the evaluation of this project)

Refer to OraCrypt Profile

Date : 18 July 2005 **Signature :**

OraCrypt Profile

Sypnosis of Product

OraCrypt is a product for the Encryption and Decryption of data in Oracle RDBMSs. The product consists of 1) the key management module, 2) module for the successful encryption and decryption of data in databases. The key management module allows the generation of Master and Shared Keys. Master keys are for the hierarchical access control to encrypted data while Shared keys are useful for both direct access and hierarchical access controls. The encryption system is designed with no keys (Master or Shared keys) to be found or stored in the DBMS concern. Instead all keys are stored on smartcards.

The encryption and decryption module allow users to encrypt and decrypt data at all levels (row, column, and data element). And in each level, users may provide condition for the selective encryption and decryption of data at those three levels.

Advantage / Uniqueness

The product is designed with the capability to be incorporated into existing database application system. Therefore, the benefits to the industry namely to all users of Oracle RDBMS without regards to the application system they are using, will be it allows the users to use their existing application system with added capability i.e. encryption and decryption of data.

There are a few competing products namely DBEncrypt and Secure.Data from the USA. Our evaluation and comparison with them showed that ours can encrypt and decrypt data at three levels i.e. Data Item, Row and Column. None of the other two can and theirs limited to whole column encryption. Secure.Data usage is very cumbersome and lengthy on the part of the user requirement. One other advantage of our product is that we do not store any crypto keys in the DBMS. Instead all our crypto keys are external to the security system and will be fetched from secure smartcard systems.

In terms of efficiency, the norm in cryptography is that the data size will increase almost double after encryption. This happens to us also but we have designed a new 64-bit storage format that will reduce the storage need from double to about 1.33 percent only. This is a big achievement.

Innovation

This product is a home-made product i.e. a product from the outcome of a PhD research done in Malaysia. The design of the encryption system is an original research and utilized indigenous cryptographic algorithm that was also a product from previous PhD research effort.

Therefore, there is no question on the originality of concept and its implementation. It was successfully implemented in Oracle8i RDBMS using quite a substantial amount of data obtained from a library information system.

There are many outstanding features and is summarized as follows:

i. Able to encrypt and decrypt data at all levels i.e. data element, row and column.
ii. Data can be wholly or selectively encrypted at those three levels.
iii. No crypto keys are to be found anywhere in the RDBMS.
iv. All crypto keys are securely stored in smartcards that are using proprietary smartcard operating system.

The software development methodology is backed by the software engineers from the Centre for Advanced Software Engineering (CASE), UTM. We are using a component based system development methodology based on object orientation which will render the system ease of maintenance and adaptability. We are now porting the system to SQL*Server 2000 fairly easily.

Potential

The target market is all users of RDBMS with or without application system embedded in their DBMS. This includes the military, police, Governments, Banking and Finance, and the private sector at large.

This product has international potential especially with import restriction on encryption algorithms and security products like from the USA. With the current development of the product, there are great potential to create employment for its marketing, support and training services since the product is aimed at international market.

One important strategy for us to be ahead of competitors especially in the international arena is that we designed and developed our product based on sound software engineering practices that allow for easy maintenance, upgrades and adaptability to many types of RDBMS.

Further areas identified for expansion including:

i. Enhanced crypto key management system
ii. Increase in the types of data that can be encrypted, e.g. multimedia data types besides numeric and character based.
iii. To provide for many other established encryption algorithms besides our proprietary block and stream algorithms. This includes Blowfish, Twofish, AES, etc.
iv. Enhanced the security of key storage in smartcards.
v. Increase the efficiency of the encryption process.

Appendix A: Technology Description

This project is in the domain of Database Security, i.e. in the field of Information Security and specifically to secure data in commercial databases. The project also involves the utilization of cryptographic tools/primitives developed in Malaysia. Since no cryptographic keys are stored in the secured databases, the research project makes use of proprietary smartcard security system to store securely all cryptographic keys produced by the system.

This invention is new because:

1. The design of the database security mechanism is based on a new database encryption scheme. The scheme allows the encryption and decryption of data using both block and stream cipher algorithms.
2. The scheme is being driven by a new key management system which was designed and developed using the block cipher algorithm.
3. The implementation of the scheme does not require any cryptographic keys to be stored anywhere in the database management system. Instead, all cryptographic keys are secure stored in proprietary smartcard security system developed in-house.
4. Encrypted data in the databases are accessed using a new approach based on Initialization Vectors.
5. The database security mechanism is implementable in Oracle RDBMS. In addition, the security mechanism may utilize any third party encryption algorithm including those supplied by Oracle RDBMS.
6. It is proven that implementation of the package allows data in databases to be encrypted at the data element, row, and column levels.

The package solves the security problem of commercial databases which currently does not provide a full capability for database encryption. For example, Oracle only provide DES and 3DES algorithms without any key management system. Users are not inclined to use encryption in Oracle because they do not have a key management system which they need to develop their own. With this technology, any database applications can incorporate the proposed security mechanism into their varied information application systems. In other words, applications related to banking, finance, military, and many other national security requirements can take benefit of securing their databases.

A good example of this type of application is to secure personnel databases of a company's Human Resource application system.

Appendix B: Market Potential

The target markets are all users of RDBMS with or without application system embedded in their DBMS. This includes the military, police, Governments, Banking and Finance, and the private sector at large.

This product has international potential especially with import restriction on encryption algorithms and security products like from the USA. With the current development of the product, there are great potential to create employment for its marketing, support and training services since the product is aimed at international market.

One important strategy for us to be ahead of competitors especially in the international arena is that we designed and developed our product based on sound software engineering practices that allow for easy maintenance, upgrades and adaptability to many types of RDBMS.

Further areas identified for expansion including:

1. Enhanced crypto key management system
2. Increase in the types of data that can be encrypted, e.g. multimedia datatypes besides numeric and character based.
3. To provide for many other established encryption algorithms besides our proprietary block and stream algorithms. This includes Blowfish, Twofish, AES, etc.
4. Enhanced the security of key storage in smartcards.
5. Increase the efficiency of the encryption process.

Appendix C: Commercialization Strategies

Refer to OraCrypt Brochure..