DATA SECURITY IN FAULT TOLERANT HARD REAL TIME SYSTEMS - USE OF TIME DEPENDANT MULTIPLE RANDOM CIPHER CODE

A thesis submitted

by

VARGHESE PAUL

in partial fulfillment of the requirements
for the degree of
DOCTOR OF PHILOSOPHY
of
COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY
UNDER FACULTY OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE
COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY
KOCHI – 682 022

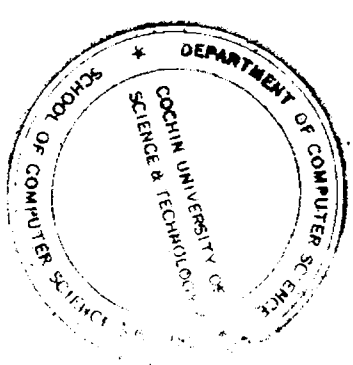
APRIL 2003

CERTIFICATE

This is to certify that the thesis entitled *DATA SECURITY IN FAULT TOLERANT HARD REAL TIME SYSTEMS – USE OF TIME DEPENDANT MULTIPLE RANDOM CIPHER CODE* is a report of the original work carried out by Mr. Varghese Paul, under my supervision and guidance in Department of Computer Science, Cochin University of Science and Technology. No part of the work reported in this thesis has been presented for any other degree from any other University.

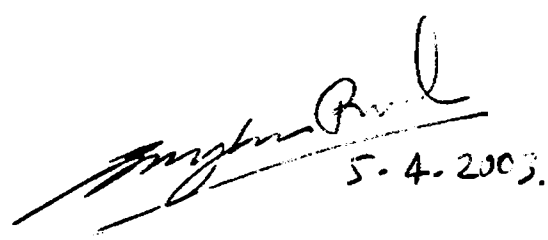
Kochi 682 022
April 05, 2003

Dr.K Poulose Jacob
(Supervising Teacher)
Professor and Head
Department of Computer Science
Cochin University of Science and Technology



DECLARATION

I hereby declare that the work presented in this thesis is based on the original work done by me under the supervision of Dr. K Poulose Jacob, Professor and Head, Department of computer Science, Cochin University of Science and Technology. No part of this thesis has been presented for any other degree from any other institution.



Kochi 682 002
April 05, 2003

VARGHESE PAUL

ACKNOWLEDGEMENTS

I wish to place on record my profound sense of gratitude and thanks to Prof (Dr.) K Poulose Jacob, Head, Department of Computer Science for his inspiring guidance and constant encouragement throughout this investigation.

I am deeply indebted to Dr. S. Sasikumaran, Professor of Computer Applications, Institute of Management, Government of Kerala, Thiruvanathapuram for the guidance and supervision in the initial stage and valuable advice and unstinted support throughout the course of this work.

Dr. A. P. Kuriakose, Former Dean, Faculty of Technology, CUSAT, deserves special thanks for arranging the global contest for checking the vulnerability of TDMRC Code developed during this research work.

I thank Dr. Jacob Mathew, President, International Cyber Business Systems Inc, Cleveland, USA for the help and guidance rendered to complete this work.

My sincere thanks to the Librarian, Central Library, VSSC, Thiruvananthapuram for granting permission and facilitating reference of the books and journals there.

The help and assistance rendered by the teachers and staff of the department of Computer Science and School of Engineering are gratefully acknowledged.

VARGHESE PAUL

CONTENTS

CHAPTER - I

INTRODUCTION

Any system where a timely response by the computer to external stimuli is vital is a Real Time System. Real Time Systems must satisfy explicit response time constraints or risk severe consequences including failure. Logical correctness of the output of such systems is based on both the correctness of the inputs and their timeliness.

Real Time Systems are typically designed for specific applications. They have the advantage that the characteristics of the application and it's environment are more precisely known compared to their general purpose counterpart. As a result it is possible to fine tune real time systems more precisely for optimum performance.

Real Time Systems can be classified into two categories – *Soft Real Time Systems and Hard Real Time Systems*. In Soft Real Time Systems performance is degraded but not destroyed by failure to meet response time constraints whereas in Hard Real Time systems failure to meet response time constraints will lead to failure of the system itself.

Distributed computer systems are increasingly being employed for critical applications, such as aircraft control, industrial process control,

and banking systems. Maximizing performance has been the conventional objective in the allocation of tasks for such systems. Inherently, distributed systems are more complex than centralized systems. The added complexity could increase the potential for system failures. Some work has been done in the past in allocating tasks to distributed systems, considering reliability as the objective function to be maximized. Reliability is defined to be the probability that none of the system components fails while processing. This, however, does not give any guarantees as to the behavior of the system when a failure occurs. A failure, not detected immediately, could lead to a catastrophe. Such systems are unsafe.

A system is considered fault tolerant if the behaviour of the system, despite the failure of some of it's components, is consistent with it's specifications. Fault tolerant systems have the capability to function in the presence of fault. By employing fault tolerance, many potential failures are averted, thereby increasing the reliability. Another goal of fault tolerance is to increase the system availability, that is, increase the time for which the system is available for user services. Redundant systems are used for achieving this quality. When redundant systems are used consistency of data among various systems is of prime importance. The available data and the processed output should be compared between various redundant systems at frequent intervals.

When the redundant systems are located at geographically distant places this comparison is to be done by transmitting data and output through communication links between various constituent systems. The rate of data transmission should also be high. These

communication channels are to be well protected against intruders especially when the system is used for strategic applications like military, aerospace research, nuclear research etc. Since Fault Tolerant Hard Real Time Systems are widely used in high tech warfare also, the chance of intrusion and risk of forced leakage of confidential information is very high in this field. To ensure correct data reception there exist many error checking and error correcting codes. But for security from eves droppers it is better to use encryption techniques in this kind of network so that the actual information can be kept away from the intruders even if they manage to gain access to the communication channel.

The present research problem is to study the existing encryption methods and to develop a new technique which is performance wise superior to other existing techniques and at the same time can be very well incorporated in the communication channels of Fault Tolerant Hard Real time systems along with existing Error Checking / Error Correcting codes, so that the intention of eaves dropping can be defeated. There are many encryption methods available now. Each method has got it's own merits and demerits. Similarly, many crypt analysis techniques which adversaries use are also available.

Information on fault tolerant hard real time systems and encryption methods, were surveyed in research journals as well as books. The internet was often found as a good repository in collecting details.

Detailed study conducted on data encryption techniques lead to the development of a new data encryption method named *Time Dependant*

Multiple Random Cipher Code (TDMRC Code) which can be very effectively used for this purpose. This particular method has many complexities compared to other methods and cryptanalysis is practically impossible.

This method is a product code which uses variable block length where as the conventional methods are of fixed block length.

The code used for any particular character differs depending upon time – that is, coding is time dependant. Even for centi second difference, the codes will change.

The code used for the same character at different locations of the plain text are different – that is, coding is poly alphabetic. Also, Pseudo Random Number generation technique is used for code generation.

Vulnerability check of the proposed system was carried out during the course of the work. Also students, researchers and professionals were involved in the checking. A global contest with a reward was arranged to check the computational security and vulnerability of the proposed scheme. The details are given in Appendix - I

This thesis contains details of fault tolerant hard real time systems, various encryption systems in current practice, mandatory requirement of encryption methods, details of TDMRC Code and how to use this technique in communication channels linking redundant sub systems of fault tolerant real time systems.

The contents of each chapter are briefly described below.

Chapter I presents a brief description of the investigations carried out, highlighting the significance of the work. The methodology adopted and scope of the thesis is outlined.

Chapter II contains an overview of the relevant literature bringing out details of fault tolerant hard real time systems and existing security arrangements. Also limitations that exist with the current systems and the need for present day study has also been brought out.

Chapter III gives detailed review of existing encryption methods.. Also, merits and demerits of various techniques in this field are dealt with in this chapter.

Chapter IV contains details of Time Dependant Multiple Random Cipher Code. Time Dependant Multiple Random Cipher Code is a new technique of data encryption. This particular method has many complexities which make it more secure against crypt analysis. The various complexities are explained in detail in this chapter.

Chapter V deals with the implementation part of Time Dependant Multiple Random Cipher Code in Fault Tolerant Hard Real time system. Experimental set up and results of performance evaluation are described in this chapter.

Chapter VI summarises the conclusions drawn from the above investigations and discusses the scope for further work.

CHAPTER II

DATA SECURITY IN FAULT TOLERANT HARD REAL TIME SYSTEMS – REVIEW OF EARLIER WORK

2.1 Introduction

There are many phases that a system typically undergoes for supporting fault tolerance. These phases are error detection, damage confinement, error recovery, and fault treatment and continued service. Since error detection is the starting point of supporting fault tolerance, a fault tolerance strategy can be, at most, as good as its error detection method. Some of the common error detection methods are replication checks, timing checks, structural and coding checks, reasonableness checks, and damage checks.

As the error may be detected sometimes after the failure has occurred, the next step in supporting fault tolerance is to determine the extent of damage to the system state by the failure. This is done in the damage confinement phase. For damage assessment, interaction between different components will have to be examined because it is by interaction that errors can propagate. The goal is to identify some boundaries within which the spread of the error is confined. These boundaries can be dynamically determined after the error has been detected by examining the component interactions, or the component interaction can be constrained in such a manner that the error spread is limited to some predefined boundaries.

The next step is error recovery. Once the spread of an error has been identified, the error has to be removed from the system. This is done by error recovery. The two major techniques are backward error recovery and forward error recovery. In backward error recovery,

during normal computation the state of the system is periodically checkpointed. For recovery, the checkpointed state of the system is restored. If the failure is occurred after the checkpoint, this rollback will remove the error. In forward recovery, on the other hand, no previous system state is available. The goal is to make the system state error free by taking corrective actions. While backward recovery is a general technique, forward recovery requires a good diagnosis about the nature of the error.

The last phase is fault treatment and continued service. In the earlier phases, the focus was on error and error removal. But the root cause of any error is fault. Though in some cases, particularly with transient faults, just error recovery may suffice, in others, after error recovery, we must remove the fault that caused the error in order to avoid future failures. This is done in this phase. First the fault is located by identifying the faulty component. Then the system is repaired by reconfiguring the system by using the built in redundancy such that either the failed component is not used or is used in a different manner.

The availability of a system can be defined as

$$MTBF / (MTBF + MTTR)$$

where MTBF is Mean Time Between Failures and
MTTR is Mean Time To Repair.

2.1 Fault Tolerant Hard Real Time System Design

[AVI 1977] presents an excellent review of the methodology of fault tolerant system design. Road blocks in fault tolerant computing are (i) lack of continuity – many of the techniques are never disclosed (trade secrecy) thus resulting in the repetition of many mistakes in the past (ii) lack of cost / benefit measures (iii) lack of specification and acceptance tests (iv) fragmentation of efforts (v) inertia in the design process (vi) resistance to potential impact – successful introduction of fault-tolerance may cause some de-emphasis of several currently flourishing activities.

A systematic methodology for the incorporation of fault-tolerance into the architecture of computing systems is presented in [AVI 1975]. Two approaches are, fault- tolerance and fault-intolerance. In the first approach, reliability is obtained by the use of protective redundancy for error detection and recovery, while in the second approach, reliability must be obtained by the priority for elimination of the causes of unreliability.

[WEN 1974] examines the reliability, availability, recovery time, data protection and maintainability requirements for five classes of computer applications (i) general purpose time shared (ii) general purpose batch (iii) communication (iv) super fast and (v) aerospace. Possible ways of introducing redundancy are given – starting from a system containing only byte error detection in many memory to a system containing uniform redundancy (i.e. where programs are run simultaneously on two computer units).

The STAR is a fault-tolerant computer primarily intended for use in hard real time application like spacecraft guidance, control and data acquisition systems on long unmanned space missions. [AVI 1971] explains the notable features of this computer as (i) use of special processor (TARP – Test And Repair Processor) to monitor the performance of the computer and to arrange recovery when detected an error, (ii) use of hybrid redundancy – STAR employed masking redundancy (triple modular redundancy) for the implementation of TARP and standby sparing redundancy for the other modules of the computer.

[AVI 1971] describes a hybrid-redundant multiprocessor organization for space applications. Each processing unit and memory unit is triplicated and there are number of spare units available to replace failed units.

In [MER 1976], a multiprocessor system for aerospace application is described. The system uses standby sparing redundancy for processors and memory units. The notable features of this system are (i) it is reconfigurable – processors and memory units can be removed or added dynamically (ii) the provision of an automatic rollback facility whereby the state of a computation can be restored to an earlier state, (iii) implementation of this rollback facility by hardware, and (iv) automatic generation of rollback facility by a programmer is not concerned with their specification

[HAM 1972] concentrates on hardware fault tolerance of applications such as telephone exchanges which use stored program control and requires a mean time between failures (unavailability exceeding 10 minutes) of 50 years. This kind of system consists of functionally equivalent processors connected to store and input-output modules. An important aspect of these processors is their use of a capability mechanism for the protection of information stored in memory modules. When a processor detects an error it generates a fault interrupt so that programmed error recovery may be initiated.

[HOP 1975] describes a multiprocessor system designed for use as a switching node in the ARPA network. The reliability goal was to construct a system that would survive not only transient failures but also solid failures of any single component. The hardware consists of buses joined together by special bus couplers allowing units on one bus to access those on another. The buses are of three kinds: (i) processor bus, each bus can contain two processors with local memory, (ii) memory bus, to house the segments of large shared memory, and (iii) I/O bus for device controllers. Hardware reliability is achieved by keeping sufficient extra copies of hardware resources and by ensuring that these hardware copies are isolated as much as possible (so that a failure of one unit should not affect others). The paper also describes the software strategies used for error detection and recovery.

[FISH 1973] describes design philosophy for multiprocessor systems intended for ultra reliable hard real time applications. The design conditions are (i) use of *off-the-shelf* components and subsystems, (ii) realistic cost constraints (i.e. only a limited use of hardware

redundancy), and (iii) dedicated application usage. The authors make two observations: (i) increased use of LSI circuits would make exhaustive testing of complex units infeasible, and (ii) in the case of software, it is common knowledge that the complexity of such programs also makes their exhaustive testing impractical. Thus, both the hardware-and software may contain undetected design faults. The design presented in [FISH 1973] is tolerant to both classes of undetected faults. The basic idea is to run three or more versions of the application software on a suitably designed multiprocessor system that is capable of checking for any discrepancy in the results.

There are many ways of introducing redundancy into computer systems. Mathematical modeling plays an important role in the selection of appropriate techniques for meeting the given reliability goal. The reliability of a system can be quite sensitive to even small variations in certain design parameters; mathematical models provide the understanding and insight into the nature of this sensitivity. [LYO 1962] presents a thorough mathematical analysis of the triple-modular redundancy (TMR) technique. A TMR configuration with perfect voting circuits is first analysed and then the effect of imperfect voters on the reliability is considered.

[MAT 1970] says standby sparing redundancy technique has gained widespread usage in the implementation of fault-tolerant computers since it offers several advantages over static redundancy techniques. Computers employing the standby spare redundancy technique often need a hard core module for error detection and recovery. This module must be ultra reliable since its failure would leave the system fault

intolerant. The authors propose a *hybrid redundancy* technique for the design of hard core modules. It consists of a TMR (or its generalised version – NMR) system with standby spares. A detailed mathematical analysis of such a 'hybrid redundant' system is presented to show that a significant improvement over NMR systems can be obtained.

The authors of [BOU 1971] present reliability equations for most of the well known redundancy techniques. These techniques include : (i) TMR, (ii) TMR with sparing (hybrid redundancy), (iii) NMR with sparing (hybrid redundancy), and (iv) standby sparing. The last technique needs the facility of error detection and automatic reconfiguration (replacement of the failed component by one of the spares). Hence the authors introduce the important, notion of coverage, defined to be the conditional probability. A comparison of TMR and standby sparing is performed which indicates that TMR is almost unbeatable for short missions.

In [MAT 1975], the authors have developed a generalised reliability model (named GMR: General Modular Redundancy) such that the different redundancy techniques become particular cases of the model. It is therefore possible to present a unified treatment of reliability modeling. The advantage of this approach is that several different redundancy techniques can be compared with relative ease.

In [BOR 1974], a reliability model of PRIME is developed. A *crash* is defined as an interruption in the availability of a predefined minimum amount of computing power for a period of time exceeding the system's automatic recovery time. Four distinct causes of crashes

are assumed: (i) time domain multiple faults - crash due to a fault while recovering from an earlier fault, (ii) resource exhaustion - not enough resource units left to provide an acceptable service, (iii) space domain multiple faults - a crash due to the inadequacy of fault detection and recovery mechanisms, and (iv) solitary faults - the inability of the system to recover from a single fault

According to [SOM 1997], a good fault-tolerant system design requires a careful study of design, failures, causes of failures, and system response to failures. Planning to avoid failures is the most important aspect of fault tolerance. A designer must analyze the environment and determine the failures that must be tolerated to achieve the desired level of reliability. To optimize fault tolerance, it is important to estimate actual failure rates for each possible failure. The basic principle of fault-tolerant design is redundancy and there are three basic techniques to achieve it, namely, spatial (redundant hardware), informational (redundant data structures), and temporal (redundant computation).

2.2 Data Security in Fault Tolerant Hard Real Time System.

One of the essential properties a reliable system must possess is that of error confinement: the property of preventing an erroneous or corrupted software module from damaging other modules. Of equal importance is the requirement that the information stored in the system be secure from unauthorised access.

[WIL 1972] contains a concise and very reliable account of protection in computer systems. The chapter on memory management describes the two well known protection schemes: access list based and capability based. A discussion on hardware features necessary to support these schemes is also included. Later chapters describe user authentication mechanisms and file protection techniques. This book also contains details of file recovery techniques and methods of system restart after a failure.

[SAL 1975] gives a very comprehensive survey of techniques for protecting computer-stored information from unauthorised use or modification. Eight design principles for designing a protection system are given: (i) economy of mechanism. (ii) failsafe defaults (iii) complete mediation (iv) open design (v) separation of privilege (vi) least previlege (vii) least common mechanism, and (viii) psychological acceptability.

[POP 1975] describes the work undertaken at the University of California at Los Angeles with the aim of building a kernel for multi-user operating systems. The special feature of the kernel is that it is intended to provide a provably secure environment for information. The basic security is achieved by the creation of isolated virtual machines – the isolation guaranteeing the error confinement property. Great care has been taken to keep the security kernel as small and simple as possible so as to make the task of proving its correctness manageable.