

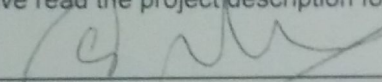
### MCS PROJECT PORTFOLIO COVER SHEET

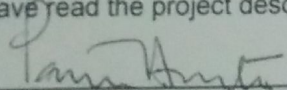
**Instructions:** Students must complete all sections below and obtain the signatures of the faculty members who taught the courses included portfolio projects. The completed form must be submitted to the CIDSE Advising Office (BYENG 208) with the project portfolio by the posted semester deadline.

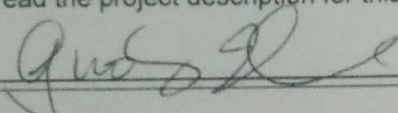
**Student Information:** Student must complete this section

Last Name Manikarnike	First Name Mukund	ASU ID 1208597425
Graduation Term Summer 2016	Current GPA 3.57	Date submitted 06/06/2016

**Project Information:** Student must complete the sections below. Faculty must review the portfolio project that pertains to their specific course and sign in the appropriate section.

PROJECT 1: Instructor name - Dr. Sandeep Gupta		
Course Advanced Operating Systems (CSE 536)	Semester completed Spring 2015	Final grade in course A
Faculty Statement: My signature below certifies that the project was at least 30% of the final grade for the course listed above, and I have read the project description for this course and approve the content.		
Faculty signature: 		Date: 3/29/16

PROJECT 2: Instructor name - Dr. Partha Dasgupta		
Course Applied Cryptography (CSE 539)	Semester completed Spring 2015	Final grade in course B
Faculty Statement: My signature below certifies that the project was at least 30% of the final grade for the course listed above, and I have read the project description for this course and approve the content.		
Faculty signature: 		Date: 4/5/16

PROJECT 3: Instructor name - Dr. Guoliang Xue		
Course Game Theory With Applications (CSE 556)	Semester completed Fall 2015	Final grade in course B
Faculty Statement: My signature below certifies that the project was at least 30% of the final grade for the course listed above, and I have read the project description for this course and approve the content.		
Faculty signature: 		Date: 4/5/2016

The portfolio consists of 3 projects, 1 in each of foundation, systems and applications areas. A summary of each of the projects is provided below.

### **Miniscule RAMdisk**

This project which falls under the systems area, involved the development of a file system which persists files on the RAM instead of the traditional secondary storage. The purpose of this project was to develop a system that can illustrate how files can be persisted on main memory. Such an activity would be carried out because the access rate on the RAM is much higher and utilizing this attribute of the RAM is important. Given that the filesystem persists files on the RAM, there are newer problems that need to be handled including concurrency, atomicity and fault tolerance which the implemented file system aims at achieving. The filesystem acts as a system that can support a miniscule number of files, but with the support of concurrency and atomicity. The fault tolerance aspect is only considered through a simulation rather than a real failure. The system is easily extensible and provides clear and modular APIs for the objectives that it aims to provide.

### **Cryptographic Algorithms**

This project which falls under the applications area, involved the development of several cryptographic algorithms. The first algorithm involved designing an encryption algorithm from scratch, implementing bit-wise encryption of any given file using the encryption algorithm that was designed. The second algorithm involved breaking an encryption mechanism whose source code was provided and decrypting files encrypted using the algorithm for which the source code was provided. This activity involved finding out the weakness of the algorithm and using that weakness to break the encryption scheme. The third algorithm was to design a brute force attack on a few stolen password hashes and also to design a rainbow table to perform the same, given the rules of what the passwords can contain. Using rules of the password, certain optimizations on the brute-force attack on the rainbow table were done. The fourth algorithm involved designing a public key encryption system and simulating message passing between two parties using RSA. The fifth algorithm also involved using RSA, but the purpose was to design a digital signature mechanism around it.

### **Task Allocation – A Game Theoretical approach in Project Management**

This project which falls under the foundations area was much more theoretical and research oriented than the other projects. As part of this project, the goal was to design a game theoretical algorithm that performs task allocation to members in a team such that no team member has an incentive to change their preferences on the tasks that they would like to do. By doing so, it would ensure that all requirements of team members in a team are met and there by the retention percentages in a company would increase. The approach used to design this algorithm was to treat each project as a q-stage game, use the performance indices of each team member in a previous stage as a feedback to perform allocation of tasks in a consequent stage in addition to the preferences provided by each member in the team for a set of tasks. In order to do this, utility functions for each user for each stage were defined and a task allocation function which uses the utilities combined with the effort estimates was defined. The task allocation function that produced the lowest possible result would be used as to perform allocation of tasks to all users. The paper also goes on to show, theoretically, that this task allocation function satisfies properties of being truthful, individually rational, member sovereignty, realistic computational efficiency and always exhibits a Nash Equilibrium. The details of the proofs are available in the paper, a link for which is provided in the portfolio report.

# A Miniscule Ramdisk<sup>12</sup>

Mukund Manikarnike<sup>34</sup>

Master of Computer Science, Arizona State University

[mmanikar@asu.edu](mailto:mmanikar@asu.edu) [mukunm@gmail.com](mailto:mukunm@gmail.com)

## ABSTRACT

A Ramdisk is an implementation of a file system on the RAM instead of traditional secondary storage. As memory sizes grow and become cheaper in accordance with Moore's law, RAMs as well can be used to store large enough data, although volatile. In order to enable such a storage mechanism, there is a necessity for a file system on the RAM which can handle storage of files, concurrent accesses, fault tolerance and more such features. This paper describes the implementation of a miniscule version of such a software artifact.

## Keywords

Filesystem, RAM, Ramdisk

## 1. INTRODUCTION

With the increase in the amount of memory that can fit on one die in accordance with Moore's law, we're seeing an increased usage of RAMs for all purposes. As the size grows, given the need for faster writes and reads, we would require to have a file system that abstracts out writes and reads into and from the memory, provides concurrency control and fault tolerance on the RAM. This project, as described in this paper, attempts to implement the basic requirements of such a file system on the RAM.

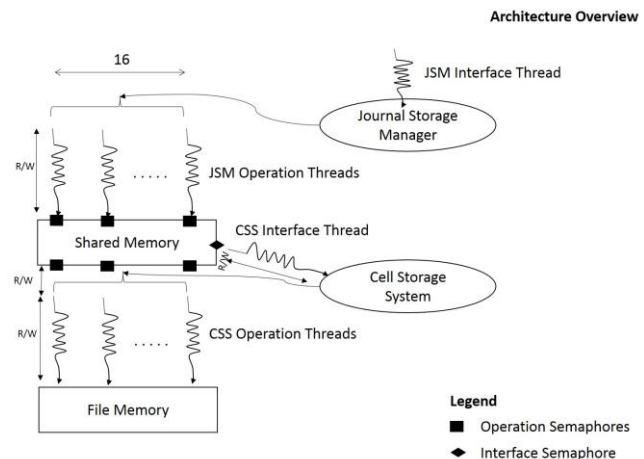
The following sections of the paper are organized as follows.

- Section 2 provides a brief description of different aspects of the file system and how each of them are important for the design of such a system.
- Section 3 describes the results that were obtained as part of this project

- Section 4 talks about the team members involved in the project and my contributions in detail.
- Section 5 ends with a few conclusions on the project and talks about future enhancements for the project.

## 2. DESCRIPTION

The filesystem implemented consists of a Journal Storage Manager and a Cell Storage System both of which run as 2 separate processes and communicate with each other using shared memory. The architecture of the system is as shown in **Figure 1**



**Figure 1 File System Architecture**

The following sub-sections describe the different memory regions and each of the key components like the Journal Storage Manager (JSM) and Cell Storage System (CSS).

### 2.1 JSM and Shared Memory

The shared memory structure is organized as shown in **Figure 2**

<sup>1</sup> The source code for this project is available at [https://github.com/mukunm/s15\\_cse536](https://github.com/mukunm/s15_cse536)

<sup>2</sup> The video demos for this project is available at the following links <http://goo.gl/DjUUIw> and <http://goo.gl/VbBfp9>

<sup>3</sup> Graduate Student at School of Computing, Informatics and Decision Systems Engineering

<sup>4</sup> For more information visit <http://www.mmanikar.com/>



Operation Type	Operation Number	Operation Bitmap	File - 1 Size	File - 1 Name	File - 1 Data	...	File - 16 Size	File - 16 Name	File - 16 Data
1 Byte	1 Byte	2 Bytes	2 Bytes	28 Bytes	512 Bytes		2 Bytes	28 Bytes	512 Bytes

Figure 2 Shared Memory Organization

The typical workflow of the Journal Storage Manager is as follows

1. Writes the operation type in the first byte of the shared memory
2. Scans for available operation number based on the operation bitmap, writes the appropriate operation number and updates the operation bitmap.
3. Writes the file size, name and data into the file section that corresponds to the operation number chosen.

## 2.2 CSS and File Memory

The file memory structure is organized as shown in Figure 3

File - 1 Name	File - 2 Name	...	File - N Name	File Use Bitmap Address	File - 1 Size	File - 1 Data	...	File - N Size	File - N Data
28 Bytes	28 Bytes		28 Bytes	N/8 Bytes	2 Bytes	512 Bytes		2 Bytes	512 Bytes

Figure 3 File Memory Organization

The CSS runs in an eternal while loop and performs the following actions

1. Reads data from the shared memory
2. Decides what operation needs to be performed and on which file it needs to be performed
3. Based on the operation requested, it updates the corresponding file structure and the file inode.

The life cycle of an operation in this system is as shown in

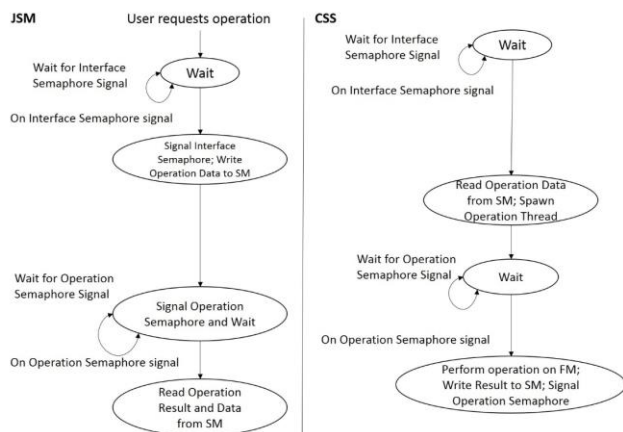


Figure 4 Operation Lifecycle

## 3. RESULTS

The following tests were carried out as part of this project and all of them were found to be successful

1. Simple read/write with or without errors.
2. Concurrent read/write with or without errors.

Some of the memory sizes used were measured and the following is a summary of the memories.

1. Shared Memory – 8.47 KB
2. File Memory – 542.125 KB.

The sizes have been limited only illustration purposes. If required, there isn't any functional limitation as part of the implementation to scale it to higher number files and bigger file sizes.

## 4. CONTRIBUTION SUMMARY

This project was carried out with only a single member in the team who is the author of the paper as well.

The key contributions of this project are

1. Implementation of a file Inode and file storage structure on the RAM.
2. Implementation of concurrent file reads/writes.
3. Implementation of inter-process communication between the journal storage manager and the cell storage system using shared memory.
  - a. Handling Multi-threading with the use of semaphores
  - b. Handling file accesses using locks
4. Modular design of the entire file system which spans 2901 lines of code in C.

## 4.1 Key Learnings

### 4.1.1 Theoretical Learnings

The following key system design principles discussed as part of [1] have been used as part of this project

1. Separating Control and Data Flow
  - The JSM interface threads run on their own. When a user requests an operation, only control data of what operation is requested is exchanged between the interface threads. The operation threads interact with each other at a later state thereby ensuring

- that the interface thread doesn't become a bottleneck.
2. Optimizing for the common case
    - The aim of the project was to employ OS concepts and design a file system. Hence, the focus was to implement the system with smaller memory regions and limited number of files.
  3. Fault Tolerance
    - Faulty commits ensures that there's a logging system through which the system recovers on subsequent reads.
  4. End to End Principle
    - The lower layer which is the CSS is limited to simple writes and reads to or from the file memory which is in accordance with the principle
  5. Randomization
    - The operation bitmap introduces randomness in the operation number chosen to complete a user request.
  6. Exchanging state explicitly.
    - All statuses such as the operation statuses and operation go-ahead permission statuses are exchanged explicitly between the CSS and the JSM. This ensures that when a certain operation is done or the JSM gives a status message to the user, it ensures that they are correct information.
  7. Extensibility
    - As mentioned earlier, the memory section, the system has been implemented in a very easily extensible fashion if a higher size RAM is available.

#### 4.1.2 Practical Learnings

The following things were the key things learned as part of this project

1. Implementation of memory allocation to store different data in different sections in the memory and how to index them to serve data upon request
2. Usage of Pthreads to achieve multi-threaded access to shared resources
  - a. Usage of locks and semaphores
3. Avoiding memory corruptions in low level operations involving memory accesses.

4. Implementation of inter-process communication using shared memory.
5. Implementation of a finite state machine which handles user requests based on the state of the software artifact.

## 5. CONCLUSION

This project implements the basic structure of a Ramdisk with limitations on the file names, number of files supported and the maximum data that each file in the file system supports. The design however is scalable but lacks other features like deciding on maximum possible usable memory and back-up in case of power outages in order to run as a full-fledged Ramdisk. The Cell storage system also fails to de-allocate the shared memory that it allocates at the start which can lead to memory leaks. However, the concepts that this project introduces and demonstrates through the implementation are the foundations for building a software such as a Ramdisk. Hence, this project has a very extensible future if required to be extended to larger memories and to be made more durable.

## 6. ACKNOWLEDGEMENTS

I would like to express my sincere thanks to Dr. Sandeep Gupta for his efforts in discussing interesting topics and papers in class and providing me with an opportunity to present papers relevant to this area that motivated me to do well in this project. I consider the implementation of a file system on the RAM from scratch as one of the best projects I've worked on because I was able to use a lot of concepts introduced by several papers that were discussed in class.

## 7. REFERENCES

- [1] Jerome H Saltzer and M. Frans Kaashoek, Principles of Computer System Design, An introduction
- [2] Linux man pages  
<http://linux.die.net/man/>

# Cryptographic algorithms<sup>1</sup>

Mukund Manikarnike<sup>2,3</sup>

Master of Computer Science, Arizona State University

[mmanikar@asu.edu](mailto:mmanikar@asu.edu) [mukunm@gmail.com](mailto:mukunm@gmail.com)

## ABSTRACT

Cryptography is an aspect of security in system design which has several intricacies to be handled. This project aims at exploring these intricacies through implementation of certain algorithms and presenting results for the same.

The intricacies explored are in the areas of encryption, attacks on encryption, password hashes and public key systems like RSA and the usage of the same to communicate between two parties.

## Keywords

Encryption, Brute-Force Attack, Known Plaintext attack, Rainbow Table, Password Hashes, Public Key Systems, Blind Signatures

## 1. INTRODUCTION

This section talks about different aspects of the project and the organization of this paper in brief. The algorithms implemented as part of this project were

1. Self-designed 32-bit encryption algorithm and brute force attack for the same.
2. Known-plain text attack on a given encryption algorithm.
3. Breaking password hashes through brute-force and using a rainbow table.
4. Encryption using Rivest Shamir Adleman (RSA) algorithm.
5. Challenge Response and Blind Signature using Rivest Shamir Adleman (RSA) algorithm.

The following sections are organized as follows. The detailed description of each of the above algorithms are described in **DESCRIPTION**

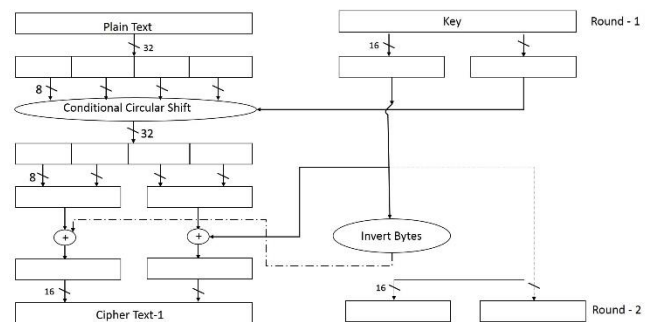
The results of the project are presented in the **RESULTS** section. The paper then concludes with key learnings in **CONCLUSION** and acknowledgements in **ACKNOWLEDGEMENTS**.

## 2. DESCRIPTION

The following sub-sections describe details of each of the algorithms that were implemented for this project.

### 2.1 Encryption Algorithm

The algorithm was designed as shown in **Figure 1**. The algorithm takes a 32 bit plain text and a 32 bit key and produces a 32 bit cipher text. The algorithm described in the figure is repeated for 7 rounds to produce the final cipher text.



**Figure 1 Encryption Algorithm Design**

In addition to the encryption algorithm, the following were also implemented

1. Integrity check using CRC-32
2. File Encryption using the underlying 32-bit encryption
3. Brute Force Attack on the encryption algorithm

The chosen language for implementation was C

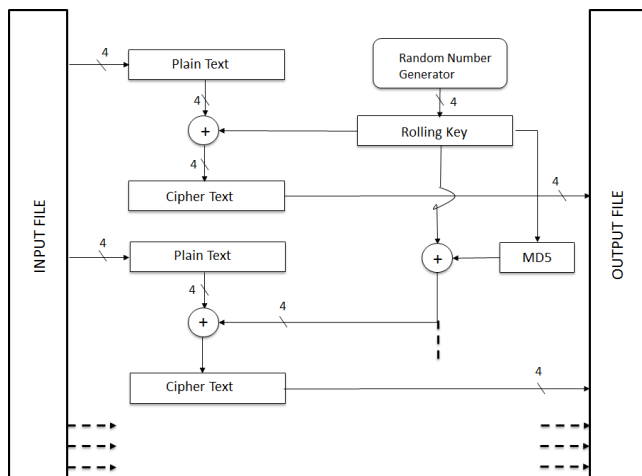
<sup>1</sup> Source code for this project is available at [https://github.com/mukunm/s15\\_cse539](https://github.com/mukunm/s15_cse539)

<sup>2</sup> Graduate Student at School of Computing, Informatics and Decision Systems Engineering

<sup>3</sup> For more information visit <http://www.mmanikar.com/>

## 2.2 Known Plain-text attack

The aim of this activity was to find the weakness in a given encryption algorithm and use the weakness to decrypt files encrypted using the algorithm without the key provided. It turned out that the encryption algorithm just did an XOR operation with the first 32-bit word of the key and the plaintext. The encrypted files provided were PDF and PNG files. It was known that the first 4 bytes of a PDF file contain %PDF and %PNG. So, given this known plaintext, the approach taken was to reverse the XOR operation and obtain the original key with which the document was encrypted with. The design of the algorithm is illustrated in **Figure 2**.



**Figure 2 Flawed Encryption Algorithm Design**

The chosen language for implementation was C.

## 2.3 Breaking Password Hashes

The aim of this experiment was to find passwords of 10 stolen password hashes. The passwords and hashes were of size 32 bits. The passwords could only contain digits from 0 to 9, upper and lower case alphabet.

The hashes were broken to obtain the original passwords using 2 approaches

### 2.3.1 Brute Force Approach

The brute force approach involved producing all possible passwords and hashing them to see if it matched any of the hashes provided. If the hash matched, then the password would be found. All provided hashes were broken using this method.

### 2.3.2 Rainbow Table Approach

The rainbow table approach involved choosing 'n' random possible passwords, designing a reduction function which would output resultant passwords that satisfy rules of a password. Using this, the rainbow

table was generated by alternately hashing and reducing each of the 'n' random passwords chosen.

The rainbow table was then used to look-up a password by re-computing the hash chain starting from each stolen hash. If the hash produced a match in the rainbow table, the previous value in the hash chain would be the password for the stolen hash.

The chosen language for implementation was C.

## 2.4 Encryption using RSA

The aim of this experiment was to perform the following activities given the Digital Certificate of a person and a certificate authority.

1. Verify the certificate of the person using the certificate of the certificate authority.
2. Read public and private keys of the person and the certificate authority.
3. Read the signature string on the person's certificate.
4. Encrypt a sample text using the public key of the person and decrypt the encrypted output using the private key of the person.

All of the above tasks were implemented as expected using the following classes in Java which have APIs to do each of the above mentioned tasks.

1. X509Certificate
2. CertificateFactory
3. PublicKey
4. PrivateKey
5. PKCS8EncodedKeySpec
6. Cipher

## 2.5 Challenge-Response and Blind Signature

The aim of this experiment was three-fold

1. Experiment with different prime and non-prime numbers as inputs RSA and see how encryption and decryption are affected by the same.
2. Implement a Challenge-response mechanism using RSA.
3. Implement a Blind Signature mechanism using RSA.

The following sub-sections talk about the Challenge Response and the Blind Signature mechanism

### 2.5.1 Challenge-Response Scheme

The challenge-response scheme was to encrypt an input using the public key and decrypt using the private

key of another person. The success of the decryption determined the success of the scheme.

### 2.5.2 Blind Signature Scheme

Blind Signature is a kind of signature, where the signing authority doesn't know what is being signed.

Given that Alice is the one signing blindly and Bob is requesting a blind signature from Alice, the scheme was implemented as follows

1. Alice obtains the public key and Modulus N of the person (Bob) who is to sign the message.
2. Obtain a random number and its inverse with respect to the Modulus [Not phi] of Bob.
3. Alice obtains/generates a message to be signed.
4. Alice encrypts the random number with the public key.
5. Alice multiplies this value by the message
6. Alice then takes a modulus over N
7. Alice sends it to Bob
8. Bob simply decrypts the received value with the private key
9. Bob sends it back to Alice
10. Alice then multiplies the received value with the inverse and take a modulus over N.
11. The value obtained above is the signed message.

The chosen language of implementation was C++.

## 3. RESULTS

The following sub-sections describe the results obtained from each of the activities described in **DESCRIPTION**. The results of the experiments involving RSA are pretty much self-explanatory and hence, aren't included as part of this section.

### 3.1 Encryption Algorithm

The theoretical analysis of the algorithm suggests a weakness in the algorithm which is if the key were to be 0 or  $2^{32}$ , the cipher text produced would be same as the plain text.

The statistical analysis of average bit changes in cipher text with a change of one bit in the key was carried out. The results of the experiment are shown in **Figure 3**.

Minimum	Maximum	Average
0	14	2

Figure 3 Encryption Algorithm Statistics

### 3.2 Known plain-text attack

Using the weakness in the algorithm, the key was obtained and the PDF and PNG file were decrypted. Since, it was a known plain-text attack, the attack was completed in a very short time.

### 3.3 Breaking Password Hashes

#### 3.3.1 Brute Force approach

The brute force approach produced passwords for each stolen hash as shown in **Figure 4**. The password hashes were broken in 110s.

Hash	Password
0x97e75d32	A1B2
0x19fbc7c1	LOVE
0x8f6bb61b	MOVE
0x88df723c	lOvE
0x655ca818	mOvE
0x14928501	1a2b
0x3974cffc	LoVe
0x58712b2b	MoVe
0x7e1d96fd	love
0x8e564270	move

Figure 4 Passwords and Hashes

#### 3.3.2 Rainbow Table approach

The rainbow table approach was able to break 7 out of the 10 password hashes provided. The statistics shown in **Figure 5** represent the characteristics of the Rainbow Table.

RB Table – Number of rows	4766560
RB Table – Number of Hash-chains	186
Size of the Rainbow Table	36.3MB

Figure 5 Rainbow Table Parameters

## 4. CONTRIBUTION SUMMARY

The team consisted of the following two members

1. Mukund Manikarnike

Master of Computer Science, CIDSE, ASU

[mmanikar@asu.edu](mailto:mmanikar@asu.edu)



2. Lakshmi Srinivas  
Master of Science in Computer Engineering,  
CIDSE, ASU  
[lsriniv2@asu.edu](mailto:lsriniv2@asu.edu)

As part of this project, I designed and implemented all parts of the project except for

1. Statistical Analysis of the Encryption algorithm
2. Challenge Response and Blind Signature Scheme using RSA.

#### 4.1 Key Learnings

The following were the key things learned from the project

1. Implementation of bit level operations and wrapping on top of them to encrypt and decrypt files byte-wise.
2. Detecting weaknesses in encryption algorithms
3. Designing a rainbow table that suits both the space and time complexity requirements and is also able to break password hashes efficiently.
4. Understanding of RSA internals.

#### 5. CONCLUSION

There are several conclusions that can be drawn from this project. The basic requirement for an encryption algorithm is that it has to satisfy the basic properties of confusion and diffusion to produce a good encryption scheme. The quality of an encryption algorithm is defined by the number of changing bits when one bit in the plain-text or the key changes. The algorithm should produce a cipher text that has no part of it as just a simple function of the plain text and the key for it to not be susceptible to known plain-text attacks. If password hashes are small, they can easily be broken through brute force or through the use of a rainbow table.

#### 6. ACKNOWLEDGEMENTS

I would like to express my sincere thanks to Dr. Partha Dasgupta for his efforts in discussing interesting topics in class that motivated me to do well in this project.

#### 7. REFERENCES

- [1] A PAINLESS GUIDE TO CRC ERROR DETECTION ALGORITHMS  
[http://www.zlib.net/crc\\_v3.txt](http://www.zlib.net/crc_v3.txt)
- [2] PDF File format  
[https://en.wikipedia.org/wiki/Portable\\_Document\\_Format](https://en.wikipedia.org/wiki/Portable_Document_Format)
- [3] PNG File format  
[https://en.wikipedia.org/wiki/Portable\\_Network\\_Graphics](https://en.wikipedia.org/wiki/Portable_Network_Graphics)
- [4] Philippe Oechslin, Making a Faster Cryptanalytic Time-Memory Trade-Off

# Task Allocation

## A Game theoretical approach in Project Management<sup>1</sup>

Mukund Manikarnike<sup>23</sup>

Master of Computer Science, Arizona State University

[mmanikar@asu.edu](mailto:mmanikar@asu.edu) [mukunm@gmail.com](mailto:mukunm@gmail.com)

### ABSTRACT

Game theory has several fields to which it can be applied to. This project explores the application of game theory to task allocation in project management, specifically software projects. This paper describes the game designed for this particular allocation strategy, how different aspects of game theory are satisfied by the game model designed for this purpose.

### Keywords

Software Project Management, Game Theory, Task allocation, Project Manager, Team Members

## 1. INTRODUCTION

Project management is an important aspect in any project. It requires the project manager to take into account, the interests and performances of an employee and targets the company has set for itself and perform an efficient task allocation so that the following two criterion are met.

1. The employees are satisfied with their assigned tasks.
2. The company is able to meet its targets.

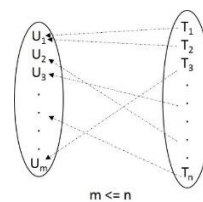
There are a lot of variables in performing such an allocation and qualitative analyses of such a problem do not produce definite results. This project as described by this paper proposes a game theoretical model to perform task allocation. It takes the above mentioned parameters into consideration and proposes a game model which satisfies truthfulness, individual rationality, member sovereignty, computational efficiency and Nash equilibrium.

The following sections of this paper are organized as follows

- Section 2 provides a brief description of different aspects of the game that was designed for task allocation how it fits into the real world.
- Section 3 describes the results that were obtained as part of this project
- Section 4 talks about the team members involved in the project and my contributions in detail.
- Section 5 ends with a few conclusions on the game theoretical model

## 2. DESCRIPTION

The game model starts with the assumption that given a project manager, a set of tasks and a set of team members, every team member has a certain amount of non-zero interest level in every task, has full information about all tasks, is assigned at least one task and every task is assigned to more than one team member.



**Figure 1 Task Assignment**



**Figure 2 q-Stage Game**

The task assignment is carried out as illustrated in **Figure 1**. The task assignment is done in a q-stage game as shown in **Figure 2** which is very similar to the real world where a project is planned in several stages or sprints. It takes into account, the satisfaction factor of each team member, performance index of each team

<sup>1</sup> The complete paper is available at <http://goo.gl/2W06G2>

<sup>2</sup> Graduate Student at School of Computing, Informatics and Decision Systems Engineering

<sup>3</sup> For more information visit <http://www.mmanikar.com/>

member in all stages of the game except for the first stage in calculating the utilities of each user. It then defines a task allocation function which is computed for each possible task allocation vector and chooses the assignment which computes to the lowest value of task allocation function.

This paper describes the model in a very qualitative fashion. For quantitative results, please go through the complete paper<sup>4</sup>

### 3. RESULTS

The project goes on to prove certain aspects of the model which hold true and hence make it practical. Each of these aspects are briefly discussed below.

#### 3.1 Theoretical Analysis

##### 3.1.1 Truthfulness

The aspects used in assigning tasks are the effort estimates and the performance indices of the team members in the previous stages. Hence, each member is forced to be truthful because if he lies, he wouldn't be able to fulfil the targets and would get lesser preferred tasks allocated to him in the upcoming stages. Hence, there is always an incentive for every team member to be truthful in their effort estimates.

##### 3.1.2 Individual Rationality

This means that each team member's utility is non-zero. The model guarantees this because the task allocation function is designed in such a way that the utility of each team member is non-zero.

##### 3.1.3 Member Sovereignty

This requires that each member be assigned some task. Given the fact that the model ensures individual rationality, it also means that a member whose utility is non-zero gets assigned at least one task which is also stated as part of the assumptions. Hence, it ensures member sovereignty.

##### 3.1.4 Computational Efficiency

The model designed computes the task allocation function for each possible task allocation vector and then decides the appropriate allocation vector which

can be inefficient. But, the argument is that team sizes in such a model are usually low (typically  $< 20$ ) and hence given that scenario, this model could be easily computationally efficient.

##### 3.1.5 Nash equilibrium

This is a very important aspect of the model. It states that given certain choices made by other team members, a chosen team member doesn't have an incentive to change his decision. The fact that the model forces every team member to be truthful shows that no team member can increase his overall gain by changing his decision. Hence, the task allocation function if computed properly for each task allocation vector, produces a unique Nash equilibrium.

This is important because it shows that the game model is able to assign tasks in such a way that no one wants to change to a different task and that everyone is happy while meeting the set targets for the project by the company.

#### 3.2 Comparative Study

This section looks at other studies in this area. The following are the comparisons

1. There are studies involving the time, money and features required to be developed for a project being used as a factor to maximize the utilities of both, the employees as well as the organization <sup>[1]</sup>.
2. There is a similar one to our study which uses game theory in task assignment which provides certain game theoretic algorithms to solve the problem at hand <sup>[2]</sup>.

But, none of these provide the quantitative analysis required to prove a concept as provided as part of our effort.

### 4. CONTRIBUTION SUMMARY

The team consisted of the following four members

1. Mukund Manikarnike

Master of Computer Science, CIDSE, ASU

[mmanikar@asu.edu](mailto:mmanikar@asu.edu)

---

<sup>4</sup> The complete paper is available at <http://goo.gl/2W06G2>

2. Nitish Bhatia  
Master of Computer Science, CIDSE, ASU  
[nbhatia3@asu.edu](mailto:nbhatia3@asu.edu)
3. Surbhi Aggarwal  
Master of Computer Science, CIDSE, ASU  
[saggarw9@asu.edu](mailto:saggarw9@asu.edu)
4. Arpit Gupta  
Master of Computer Science, CIDSE, ASU  
[agupt103@asu.edu](mailto:agupt103@asu.edu)

assignment is an important contribution because it can help the project manager ensure employee retention and target dates are paid equal attention.

The weak point is that it doesn't address the problem of computational inefficiency that the model poses for large teams.

The following were my contributions to the project

1. Survey on related work in this area.
  - a. Evaluation of problem of project estimation as proposed in [1]
2. Design of the proposed game model
  - a. q-Stage game
  - b. Mathematical model for the following
    - i. Utility function
    - ii. Task Allocation function
3. Analysis of the Game model in terms of
  - a. Truthfulness
  - b. Individual Rationality
  - c. Member Sovereignty
  - d. Computational Efficiency

#### 4.1 Key Learnings

The following were the key things learned from this project.

1. The idea behind applying game theory to a real world problem.
2. Designing a practical game theoretical model.
3. Analyzing the required properties of the model.
4. Approximation of models to suit real world scenarios rather than looking for idealistic results.

## 5. CONCLUSION

The strong points of this paper are

1. A q-stage game is a closest to real-world scenarios.
2. Fairness in assignment in the q-stage game averages out and tends to an equilibrium as the game proceeds which is the desired result from this task.
3. Quantification of something as abstract as satisfaction of an employee in the task

## 6. ACKNOWLEDGEMENTS

I would like to express my sincere thanks to Dr. Guoliang Xue for his efforts in discussing interesting topics in class that motivated me to do well in this project.

I would also like to thank my team members for their sincere efforts in supporting my ideas on the game model and collaboratively working towards revising it to make it more practical and computationally efficient.

## 7. REFERENCES

- [1] Anurag Shrama, SDET at Intel Security  
<https://www.linkedin.com/pulse/applying-gametheory-software-project-estimation-anuragsharma>
- [2] Brent Lagesse, University of Texas at Arlington,  
Department of Computer Science and Engineering  
Game-Theoretical Model for Task Assignment in  
Project Management of Innovation and  
Technology, 2006 IEEE International Conference
- [3] Software Project Management - Wikipedia  
[https://en.wikipedia.org/wiki/Software\\_project\\_management](https://en.wikipedia.org/wiki/Software_project_management)
- [4] Task Management – Wikipedia  
[https://en.wikipedia.org/wiki/Task\\_management](https://en.wikipedia.org/wiki/Task_management)