Information Operations Across Infospheres

Annual Report

Prepared by

The University of Texas at Dallas

Submitted to:
Air Force Office of Scientific Research

October 10, 2006

Under
Contract: FA9550-06-1-0045

Period of Performance:
December 1, 2005 – August 30, 2006

Subcontractor:
George Mason University

Contact Information

Dr. Bhavani Thuraisingham
Professor of Computer Science
and Director of the Cyber Security Research Center
Erik Jonsson School of Engineering and Computer Science
Box 830688, EC 31
University of Texas at Dallas,
Richardson, TX 75083-0688
Tel: 972-883-4738
Fax: 972-883-2349
Email: bhavani.thuraisingham@utdallas.edu
http://www.utdallas.edu/~bxt043000/

EXECUTIVE SUMMARY OF THE PROJECT

There is a critical need for organizations to share data within and across infospheres and form coalitions so that analysts could examine the data, mine the data, and make effective decisions. Each organization could share information within its infosphere. An infosphere may consist of the data, applications and services that are needed for its operation. Organizations may share data with one another across what is called a global infosphere that spans multiple infospheres. It is critical that the war fighters get timely information. Furthermore, secure data and information sharing is an important requirement. The challenge is for data processing techniques to meet timing constraints and at the same time ensure that security is maintained.

This proposal addresses information operations across infospheres. We first describe secure timely data sharing across infospheres and then focus on Role-based access control and Usage control in such an environment. Our goal is to send timely information to the war fighter while maintaining security. We will also address the application of game theory as well as decision centric data mining techniques to extract information from both trustworthy and untrustworthy partners of the coalition.

In particular, the **objectives** of this project are as follows:

- Develop a Framework for Secure and Timely Data Sharing across Infospheres.
- Investigate Access Control and Usage Control policies for Secure Data Sharing.
- Develop innovative techniques for extracting information from trustworthy and untrustworthy partners.

Technical Merit: While there has been work on data sharing across coalitions, an in-depth investigation of security issues as well as a study of the tradeoffs between security and timely processing has yet to be carried out. To our knowledge, this project is the first to investigate sophisticated security techniques such as Usage Control as well as decision centric data mining techniques for timely and secure data sharing across coalitions.

Broader Impact: The research to be carried out on this project is directly applicable to Network Centric Operations (NCO) that implement Network Centric Warfare (NCW). NCW promotes information sharing, shared situational awareness and knowledge of commander's intent. In addition it also enables war fighting advantage by providing synchronization, speed of command and increased combat power. We focus mainly on information sharing aspects of NCW. In particular, the results of this project can be transferred to the timely and secure data sharing services of the Network Centric Services activity being carried out by the Department of Defense.

Research Team: The research will be carried out both at the University of Texas at Dallas and at George Mason University. The principal investigators are among the leading researchers in Data and Applications Security. They have conducted innovative research in Secure Database Design, the Inference Problem, Role-based Access Control and Usage Control techniques as well as and carried out technology transfer activities. They are Fellows of IEEE, ACM, AAAS and the British Computer Society and have received prestigious awards (including from IEEE) for their research in Data and Applications Security.

ABSTRACT OF THE ANNUAL REPORT

The research presented in this annual report was carried out at mainly the University of Texas at Dallas (UTD) between December 1, 2005 and August 30, 2006. It describes the issues and challenges for information operations across infospheres and focuses on assured information sharing. We have examined three models: In the first model the partners of the coalition are considered to be trustworthy. In the second model, the partners are semi-trustworthy. In the third model the partners are untrustworthy. We need to consider all three models to fight the global war on terror.

This annual report essentially consists of five technical reports published at the University of Texas at Dallas. An overview of Assured Information Sharing is discussed in UTD-CS-43-06 (Report #1). In the case of trustworthy models we conducted experiments on data sharing vs. data policy enforcement. We also carried out data mining before and after policy enforcement. The results are discussed in UTD-CS-44-06 (Report #2). In addition, we conducted design and simulation of trust management techniques for a coalition environment. This research is presented in UTD-CS-45-06 (Report #3). For the semi-trustworthy model we examined the use of game theory for extracting information from the partners. This research is discussed in UTD-CS-46-06 (Report #4). For the untrustworthy model, we examined the use of data mining for defensive operations. This research is discussed in UTD-CS-47-06 (Report #5).

In addition to the above, George Mason University (GMU) received a subcontract from the University of Texas at Dallas to examine the use of Role-based Access Control (RBAC) and Usage Control models for Coalition data sharing. The research was carried out at GMU between June and August 2006. This research is in progress and a summary is discussed in Appendix A. In Appendix B, we discuss some of our related research carried out at UTD under separate grants and contracts that has enhanced our current project for AFOSR.

Our work on AFOSR project for FY07 will include examining service oriented architectures for information sharing between trustworthy partners as well as determining the impact of time constraints for policy enforcement, developing probing techniques for extracting information from semi-trustworthy partners, and developing techniques for offensive information operations to handle untrustworthy partners. In addition we will also have a complete model based on RBAC/UCON for assured information sharing.

ACKNOWLEDGEMENT

Much of the research discussed in this annual report was supported by the Air Force Office of Scientific Research under Contract FA9550-06-1-0045. We thank Dr. Robert Herklotz of AFOSR for funding his encouragement and motivation. This research as also partially supported by the Erik Jonsson School of Engineering and Computer Science at the University of Texas at Dallas under the Texas Enterprise Funds. We thank Prof. Robert Helms (Dean) and Prof. Andrew Blanchard (Senior Associate Dean) for this support.

Table of Contents

Report #1:

ASSURED INFORMATION SHARING: TECHNOLOGIES, CHALLENGES AND DIRECTIONS

Bhavani Thuraisingham
The University of Texas at Dallas

Published as Technical Report: UTD-CS-43-06

ABSTRACT

This paper describes issues, technologies, challenges, and directions for Assured Information Sharing (AIS). AIS is about organizations sharing information but at the same time enforcing policies and procedures so that the data is integrated and mined to extract nuggets. This is the first in a series of papers we are writing on AIS. It provides an overview including architectures, functions and policies for AIS. We assume that the partners of a coalition may be trustworthy, semi-trustworthy or untrustworthy and investigate solutions for AIS to handle the different scenarios.

1. INTRODUCTION

Data from the various data sources at multiple security levels as well as from different services and agencies including the Air Force, Navy, Army, Local, State and Federal agencies have to be integrated so that the data can be mined, patterns and information extracted**,** relationships identified, and decisions made. The databases would include for example, military databases that contain information about military strategies, intelligence databases that contain information about potential terrorists and their patterns of attack, and medical databases that contain information about infectious diseases and stock piles. Data could be structured or unstructured including geospatial/multimedia data. Data also needs to be shared between healthcare organizations such as doctors' offices, hospitals and pharmacies. Unless the data is integrated and the big picture is formed, it will be difficult to inform all the parties concerned about the incidences that have occurred. While the different agencies have to share data and information, they also need to enforce appropriate security and integrity policies so that the data does not get into the hands of unauthorized individuals. Essentially the agencies have to share information but at the same time maintain the security and integrity requirements.

This is the first in a series of reports we are writing on Assured Information Sharing. The reports that follow will include applying game theoretical techniques for AIS among semi-trustworthy partners, defending against malicious attacks while data sharing, applying RBAC (role-based access control) with UCON (Usage Control) extensions for AIS and carrying out offensive operations against untrustworthy partners. We are also investigating risk-based access control, data origin and provenance issues as well as geospatial data management for AIS.

In this paper we describe Assured Information Sharing that will ensure that the appropriate policies for confidentiality, privacy, trust, release, dissemination, data quality and provenance are enforced. We discuss technologies for AIS as well as novel approaches based on game theoretical concepts. In section 2 we will provide an overview of an AIS architecture. Data integration and analysis technologies for AIS will be discussed in section 3. Security policy aspects including confidentiality, privacy and trust policies will be discussed in section 4. Integrity and dependability issues such as data provenance and quality and real-time processing will be

discussed in section 5. Balancing conflicting requirements including security vs. real-time processing will be discussed in section 6. Some novel approaches will be discussed in section 7. In particular applications of game theoretical techniques for handling semi-trustworthy partners will be discussed. Approaches for handling untrustworthy partners will be discussed in section 8. Discussion of the series of reports we will be writing on AIS is mentioned in section 9. The paper is concluded in section 10.

2. ORGANIZATIONAL DATA SHARING

A coalition consists of a set of organizations, which may be agencies, universities and corporations that work together in a peer-to-peer environment to solve problems such as intelligence and military operations as well as healthcare operations. Figure 1 illustrates an architecture for a coalition where three agencies have to share data and information. Coalitions are usually dynamic in nature. That is, members may join and leave the coalitions in accordance with the policies and procedures. A challenge is to ensure the secure operation of a coalition. We assume that the members of a coalition, which are also called its partners, may be trustworthy, untrustworthy or partially (semi) trustworthy.

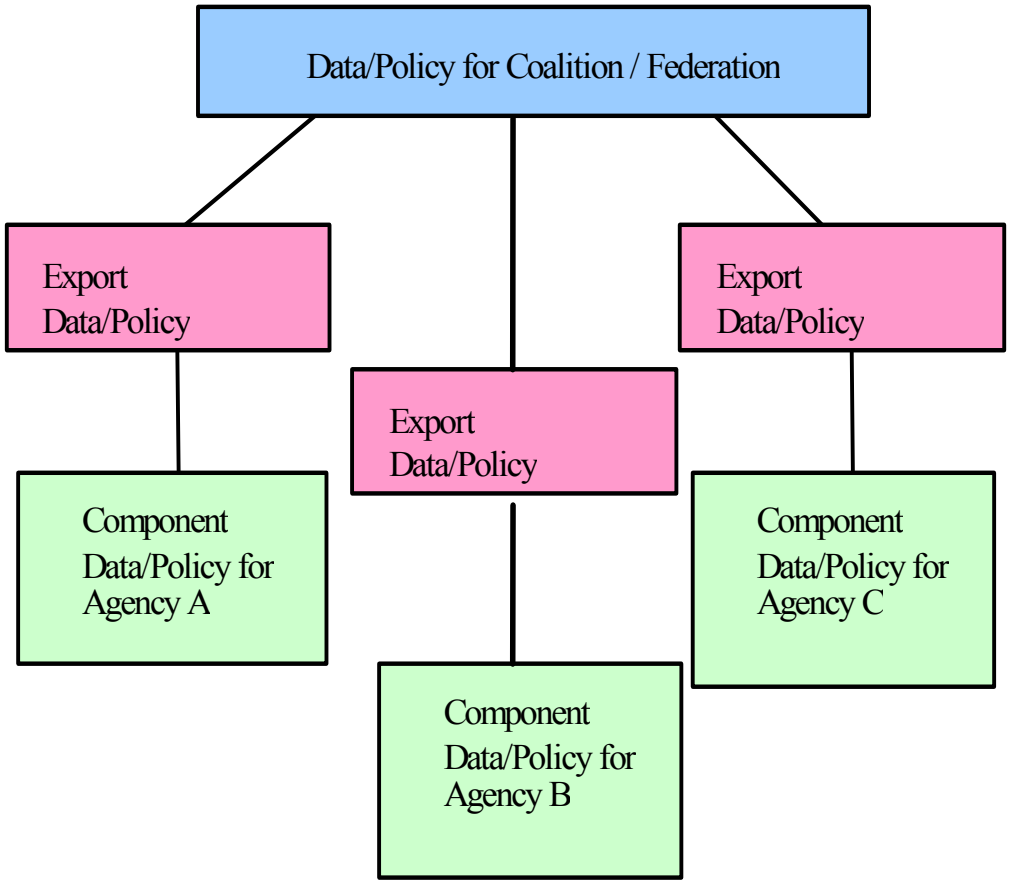


Figure 1. Architecture for Organizational Data Sharing

Various aspects of coalition data sharing are discussed in the Markle report [MARK03]. However, security including confidentiality, privacy, trust, integrity, release and dissemination has been given little consideration. Much of the prior work on security in a coalition environment has focused on secure federated data sharing. Thuraisingham was one of the first to propose multilevel security for federated database systems [THUR94]. Discretionary security was

proposed in [OLIV95]. None of the previous work has focused on determining the amount of information that is lost for conducting military operations by enforcing security. Furthermore, developing flexible policies in a coalition environment are yet to be examined. Enforcing security while meeting timing constraints remains a largely unexplored topic. A discussion of information survivability issues and the need for flexible policies for enforcing security and meeting timing constraints are given in [THUR99] and [SON95]. However, to our knowledge, no research has been reported on secure (including confidentiality, privacy, trust and integrity) and timely data sharing for a coalition environment. Some of the challenges include the following:

Data Sharing: One of the main goals of coalition data sharing is for organizations to share the data but at the same time maintain autonomy. For example, one database could be used for travel data while another database could be used to manage data pertaining to airplanes. For counter-terrorism applications and military operations, the key is to make links and associations as rapidly as possible. We need policies and procedures to determine what data to share under what conditions.

Data Mining: Data mining techniques extract patterns and trends often previously unknown from large quantities of data [THUR98]. However data mining tools could give out false positives and false negatives. This is especially critical for applications such as counter-terrorism and military operations as it could result in catastrophic consequences [THUR03]. Therefore, we need human analysts to examine the patterns and determine which ones are useful and which ones are spurious. The challenge is to develop automated tools to sift through the data and produce only the useful links and associations.

Security: Confidentiality, privacy, integrity, trust, real-time processing, fault tolerance, authorization and administration policies enforced by the component organizations via the local agencies have to be integrated at the coalition level. As illustrated in Figure 1, each organization may export security policies and data to the coalition. The component systems may have more stringent access control requirements for foreign organizations. The challenge is to ensure that there is no security violation at the coalition level.

In sections 3 through 6 we discuss various aspects on AIS assuming that the partners are trustworthy. Semi-trustworthy partners will be discussed in section 7. Untrustworthy partners will be discussed in section 8.

4 DATA INTEGRATION AND ANALYSIS TECHNOLOGIES

Data Integration: As illustrated in Figure 2, data from the various data sources at multiple levels such as local, state and federal levels have to be integrated so that the data can be mined, patterns extracted and decisions made. Data integration has been attempted for about 20 years. Until recently brute force integration techniques consisting of translators and gateways were used between the multiple data management systems. Standards such as RDA (Remote Database Access) were developed initially for client-server interoperability. Later object-base wrappers were used to encapsulate the multiple systems including the legacy systems. For example, distributed object management standards were used to encapsulate systems and applications into objects. However, common representation of the data remained a challenge. It is only recently that we have a good handle on syntactic integration through standards such as XML (eXtensible Markup Language). The idea is as follows: each data system publishes its schema (also called metadata) in XML. Since all the systems now represent their schema in XML, the systems can talk to each other in a seamless fashion.

A major challenge for data integration is semantic heterogeneity. While much progress has been made on syntactic integration, not much work has been reported on semantic integration. For example, multiple systems may use different terms for the same data; the procedure EKG (Electro

Cardiogram) is called ECG in the United Kingdom. Even within the same state, different hospitals may use different terms to mean the same entity. For example, one hospital may use the term influenza while another hospital may use the term flu. In some cases, the same term may be used to represent different entities. While repositories and dictionaries have been built, a satisfactory solution for semantic heterogeneity is still not available. The development of semantic web technologies including the Resource Description Framework (RDF) language standard shows promise to handle semantic heterogeneity.

Multimedia and Geospatial Data: Data will include structured data as well as unstructured data such as text, voice, video and audio. Data emanating from multiple data sources including sensor and surveillance data have to be integrated and shared. Managing, integrating and mining multimedia data remains a challenge. We need efficient indexing techniques as well as XML and RDF based representation schemes. Furthermore, the data has to be mined so that patterns and trends are extracted. Video data could be data emanating from surveillance cameras or news feeds such as CNN (Cable News Network) video data. Emergency response systems have to integrate geospatial data such as maps together with structured data, make sense out of the data and rapidly produce summaries so that the emergency response teams can read and understand the data [ASHR06].

Data Mining: Integrated data may be mined to extract patterns for suspicious and unusual behavior. Much of the work in data mining has focused on mining relational and structured databases. While some work has been reported on text, image, audio and video data mining, much remains to be done. For example, how can one mine integrated geospatial and multimedia data? How can false positives and false negatives be eliminated or at least reduced? What are the training models used for multimedia data? What are the appropriate outcomes for multimedia data mining? Does it make sense to extract metadata and then mine the metadata? Much remains to be done before operational tools for multimedia and geospatial data mining are developed.

Semantic Web: Semantic web is the vision of Tim Berners Lee and is utilized by many applications including e-business [LEE01]. Due to the extensive investments by the DoD (Department of Defense) and other agencies, many semantic web technologies such as XML, RDF and Ontologies have been developed for applications such as interoperability. Furthermore, semantic web technologies are being developed for different communities. These technologies are critical for AIS. For example, we need ontologies specified in languages such as OWL (web ontology language) to specify objects so that multiple systems can work with the ontologies to handle semantic heterogeneity. A member organization of a coalition can publish its schema in languages such as XML or RDF to facilitate interoperability and information extraction.

While semantic webs are being developed for different communities, there is little work on enforcing security, privacy and trust for these semantic webs. XML, RDF and Ontologies have to be secure. Furthermore, there is a need to incorporate trust negotiation for the semantic web. We are developing secure semantic web technologies for AIS [BERT04], [THUR05a].

4. SECURITY POLICY ENFORCEMENT

Security policies include policies for confidentiality, privacy, trust, release, dissemination and integrity. A broader term is dependable systems or trustworthy systems that also include real-time processing and fault tolerance. We will discuss dependability in the next section. By confidentiality we mean that data is only released to individuals who are authorized to get the data. Privacy in general deals with the situation where an individual determines what information should be released about him/her. (Note that different definitions of privacy have been proposed.) Trust policies may add further restriction to privacy and confidentiality policies. For example, a user may be authorized to get the data according to the confidentiality policies, but the system

may not trust the individual in which case the data is not released. Similarly a person may give permission to release certain private information about him or her but that person may not trust a particular web site in which case the private information is not released to the web site. Alternatively one could argue that one needs to establish trust first before establishing the confidentiality and privacy policies. For example, a user's (or web site's) trust is established before determining that the user (or web site) can received confidential (or private) information. Release policies specify rules for releasing data while dissemination policies specify rules for disseminating the data. Integrity within the context of security ensures that only authorized individuals can modify the data so that the data is not maliciously corrupted [TSYB06]. We are conducting extensive investigation on privacy preserving data mining [LIU05]. We are also investigating the use of these techniques for AIS [LIU06].

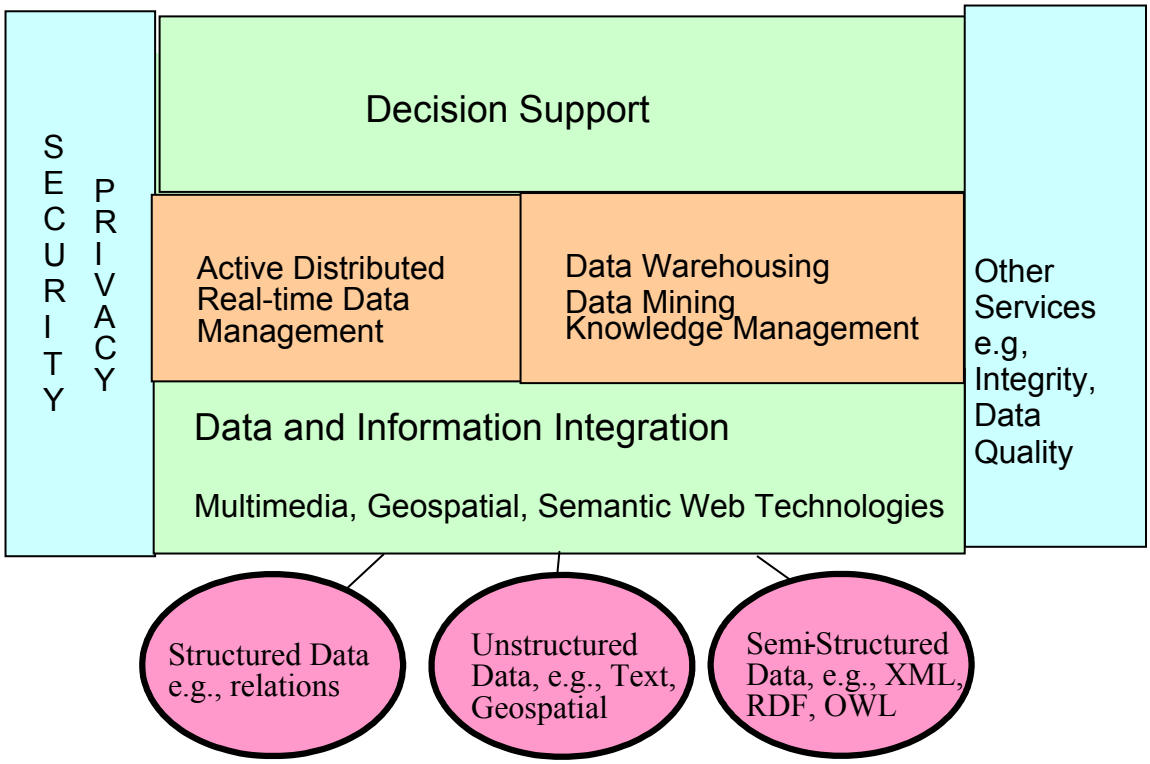


Figure 2. Data Integration and Analysis

Security for relational databases has been studied extensively and standards such as secure SQL (Structured Query Language) have been developed. In addition several secure data management system products have been developed. There has been research on incorporating security into next generation data management systems. There is also work on data quality as well as trust management. Security has also been investigated for secure object request brokers as well as for secure e-commerce systems. Finally W3C (World Wide Web Consortium) is specifying standards for privacy such as the P3P (Platform for Privacy Preferences). While there is research on incorporating security for semantic webs and heterogeneous data systems, this research is in the early stages. There is an urgent need to develop operational systems that enforce security. Furthermore, security has conflicting requirements with real-time processing. We need to enforce flexible policies and subsequently standards for specifying these policies. Security is critical for many of the information technologies we have discussed here. For a discussion of secure data sharing and related standards we refer to [THUR05b].

Security Policy Integration: There is a critical need for organizations to share data as well process the data in a timely manner, but at the same time enforce various security policies. Figure 3 illustrates security policy integration in a coalition environment. In this example, A and B form a coalition while B and C form a second coalition. A could be California, B could be Texas and C could be Oklahoma. California and Texas could form a coalition as part of the larger states in the US and Texas and Oklahoma could form a coalition as part of the neighboring states in the South of US for emergency management. There is also an urgent need for multiple organizations to share data and at the same time enforce security policies. These policies include policies for confidentiality, privacy, and trust. For example, patient data may be shared by multiple organizations including hospitals, levels of government and agencies. It is important to maintain the privacy of patient data. However it is also important that there are no unnecessary access controls so that information sharing is prohibited. One needs flexible policies so that during emergency situations it is critical that all of the data is shared so that effective decisions can be made. During normal operations, it is important to maintain confidentiality and privacy. In addition, trust policies ensure that data is shared between trusted individuals. The standards efforts in this area include Role-based access control (RBAC) [SAND96] as well as P3P (Platform for Privacy Preferences). Our partners at George mason University are examining the use of models such as RBAC and UCON for AIS [SAND06].

5. DEPENDABILITY ASPECTS

By dependable systems we mean systems that are fault tolerant and meet timing constraints. The time-critical, information-sensitive goals of managing a crisis include actions such as the early confirmation of cases and correct identification of exposed populations over a relevant time period. Early confirmation means that triggers have to be activated when certain situations (such as anomalies) occur. Suppose a hospital is flooded with 30 patients within 15 minutes who are all reporting a temperature of 105 degrees. There has to be a rule such as "If more than 30 patients register at a hospital within 20 minutes with temperature greater than 102 degrees then alert the emergency response system". To effectively process a large number of rules, we need active data management. Furthermore, the various parties involved such as federal, state and local governments have to be informed within a certain time. That is, if the authorities are notified after say 2 hours then it will be difficult to contain the spread of the disease. This means we need real-time data management capabilities. Some initial research on dependable and secure systems is discussed in [KIM06a].

While there are techniques for active real-time data management, the challenge is to develop an integrated system for end-to-end data management. For example, the data manager will ensure that the data is current and the transactions meet the timing constraints. However in an emergency situation there are numerous dependencies between different data sources. For example when rule A gets triggered, that would result in rules C, D, and E getting triggered in multiple data management systems. Such chain rule processing remains a challenge. We also need end-to-end real-time processing. That is, in addition to the data manager, the infrastructure, the network and the operating system have to meet timing constraints. This remains a challenge. Incorporating security into real-time processing techniques remains largely unexplored. For example, in an emergency situation, real-time processing and activating triggers may be more critical than enforcing access control techniques. Furthermore, the system must ensure that the deadlines are not missed due to malicious code and attacks (e.g., denial of service).

While integrity within the context of security implies that the data is not maliciously corrupted, integrity also includes policies for data quality and data provenance management. Data quality determines the accuracy of the data. This would depend on who updated the data, who owns the data and what is the accuracy of the source of the data. That is, as data moves from organization to organization, its quality may vary. Some measure to compute the quality of the data is needed.

Data provenance is about maintaining the history of the data. That is, information as to who accessed the data from start to finish is needed to determine whether data is misused [KIM06b].

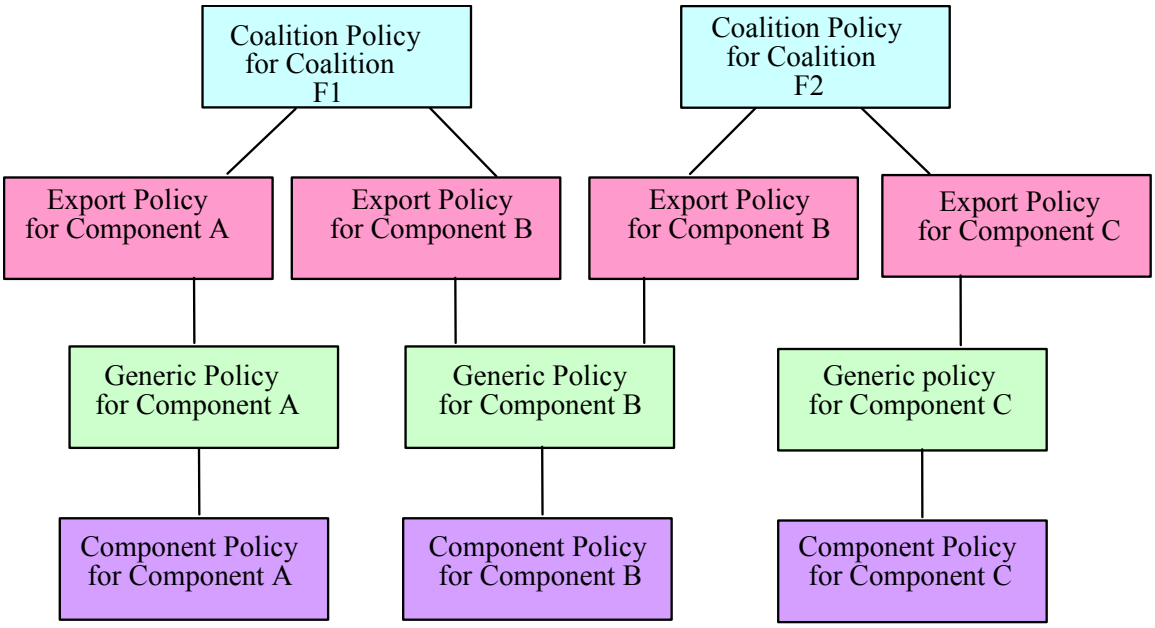


Figure 3. Security Policy Integration and Transformation for Coalitions

6. BALANCING CONFLICTING REQUIREMENTS

There are two types of conflicting requirements: one is security vs. data sharing. The goal of data sharing is for organizations to share as much data as possible so that the data is mined and nuggets obtained. However when security policies are enforced then not all of the data is shared. The other type of conflict is between real-time processing and security. The war fighter will need information at the right time. If it is even say 5 minutes late the information may not be useful. This means that if various security checks are to be performed then the information may not get to the war fighter on time.

We are conducting research in both areas. For example, we are integrating the data in the coalition databases without any access control restrictions and apply the data mining tools to obtain interesting patterns and trends. In particular, we are developing associations between different data entities such as "A and B are likely to be in a location 50 miles from Baghdad". Next we are using the same tool on the integrated data after enforcing the policies. We can then determine the patterns that might be lost due to enforcing the policies (note that there is some relationship between this work and the research on privacy preserving data mining). Our research is described in [AWAD06].

In addition, we are conducting research on examining the extent to which security affects timing constraints. For example, we enforce timing constraints on the query algorithms. That is, we first process the query using the enforcement algorithms without enforcing any of the policies. Then we enforce the security policies and determine whether the timing constraints can be met. This will determine the extent to which security impacts timely information processing.

Our goal is to develop flexible approaches and balance conflicting requirements. That is, if timely processing of data is critical then security has to be relaxed. Similarly say during non combat operations, security will have to be given full consideration. The same applies for data sharing vs. security. If during an emergency operation such as say the operation just before, during or soon after Hurricane Katrina, then several agencies will need the data without any restrictions. However during non emergency operations, security policies need to be enforced. Our research is reported in [KIM06c]. In particular, we are examining the application of RBAC and UCON models for timely data sharing.

Another aspect of our research on AIS is risk analysis. For example, if the security risks are high and the cost to implement security features are low, then security should be given high consideration. If the risks are low and the cost is high, one needs to evaluate whether it is worth the effort and cost to incorporate security. Our research on risk based access control is reported in [CELI06].

7. GAME THEORY APPLICATIONS AND SEMI-TRUSTWORTHY PARTNERS

In the previous sections we assumed that the organizations were trustworthy and would enforce the policies while data sharing. However in many cases the organization may be semi-honest or completely dishonest. In the case of semi-honest partners, organizations may have to play games to extract data. In the case of dishonest and untrustworthy partners, one may not only have to defend against malicious code, but also have to figure out what the partner is up to by monitoring his machine. In this section we will address semi-trustworthy partners and in the next we will discuss untrustworthy partners.

Semi-Honest Partners and Game Playing

To handle secure data sharing especially with semi-trustworthy partners, modeling the query processing scenario as a non cooperative game may be more appropriate especially between two partners. The players are the partners, which could be agencies or countries of a coalition. Lets assume we have Agency A and B as two partners. The objective of agency A is to extract as much information as possible from agency B. Essentially agency A wants to compromise information managed by Agency B. B's goal is to prevent this from occurring. Cooperative games on the other hand may have applications among friendly partners of a coalition. A mixture of cooperative and non-cooperative strategies may be applied for multi-party coalition.

Two-party information sharing: Information sharing between two agencies A and B may be modeled as a non-cooperative game. A has a specific objective; for example, it may know that B has some sensitive data and it wants to extract the value of that data from B. B knows A's objective. A move made by A is a query. A move made by B is the response. The game continues until A achieves its objectives or gets tired of playing the game. As stated in [JONES80], the game can be represented as a graph theoretic tree of vertices and edges. The tree has a distinguished vertex, which is the initial state. There is a payoff function, which assigns a pair of values say (X,Y) where X is the payoff for A and Y is the pay for B for each move. The payoff for A is high if it is close to obtaining the sensitive value. The payoff for B is high if the response does not reveal anything about the sensitive value. Note that if B does not give out any information or if it gives erroneous information then it cannot be regarded as a game, That is, the aim here is for B to participate in the game without giving away sensitive information.

Multi-party information sharing: The idea here is that certain parties play cooperative games while certain other parties play non-cooperative games. We illustrate with an example consisting of three parties. Let's consider an example. Suppose the year is 2006 and the UK has obtained some sensitive information on Operation Iraqi Freedom that the US needs. However, the UK is reluctant to share this information. The US in the meantime has formed an alliance with

Argentina by giving some incentive either in the form of money or weapons. When the UK hears this, it is concerned thinking about the Falklands situation. However, in reality the US has no intention of doing anything about the Falklands but does not want the UK to know the truth. So the UK may reason about the benefits it receives by sharing the data with the US and makes a determination.

Cooperative games have also been called Coalition games. In a true coalition the players are friendly and therefore share the information and determine a collective payoff. However in our environment, organizations form coalitions only to solve a particular problem. An agency that is a trustworthy party in a particular coalition may turn against its partner at a later time and divulge the information gathered during the coalition operation.

We have conducted some initial research on game theory applications for AIS. Our objective has been to consider the interaction of participants within a loose coalition. In particular, we are interested in a scenario in which those involved have made a reluctant but necessary decision to trade information to achieve some goal. A great deal of work has already been done in the areas of secret sharing and protocol enforcement. However, even if agreements to exchange are kept, there is no guarantee what is shared is legitimate. The ultimate goal of this research is to create a behavior which works optimally against lying agencies while taking advantage of implicit trust. Our results at this point in the research suggest our algorithm is effective against basic opponents, though more refinement is needed. We report which behaviors work for the players and why, with regards to the motivating factors for each strategy. Our research will be described in Volume 3 of these series [LAYF06].

8. HANDLING UNTRUSTWORTHY PARTNERS

Note that in fighting the global war on terrorism we have to work with our allies as well as with countries that we may not trust. If our partners our untrustworthy, then we have to not only defend against malicious code but also figure out what the partners are doing both with their computers as well as their activities. Essentially we need to conduct information operations [SPIT02]. We will first discuss our research on defensive operations and then discuss some aspects of offensive operations.

Defensive Operations: In the case where partners are untrustworthy we have to defend ourselves against malicious code such as viruses and worms planted by our partners. In order to accomplish this, we are applying data mining techniques to detect such malicious code. Some of our research in this area can be found in [MASU06] and will be published in Volume 4 of these series [KHAN06].

Offensive Operations: There is little work in the unclassified published literature on offensive operations. However recently we are seeing articles published in Signal magazine on the importance of monitoring the adversaries' computing activities [SIGN05a], [SIGN05b]. Three of the techniques for handling untrustworthy partners include the following:

Trojan Image Exploitation: Modern anti-virus and anti-spy ware detection packages rely on the presence of malicious code within an executable or script to prevent attacks. This is done by detection methods that are carried out when the program first loads. In theory, it is possible to circumvent this detection by designing a program without any explicit malicious code; instead, a memory leak in this program's security is purposefully created. This weakness is exploited by downloading a tailored file from the Internet, such as a picture, after the program is loaded. As a result, this program could be used as a staging area for a malicious attack.

Web Browser Customization: Web browsers have been enhanced dramatically in the past year to prevent attacks from malicious web pages. For the benefit of the user, these features are frequently made optional, allowing a great deal of customization. By compromising a user's

customization features covertly, it becomes possible to execute potential attacks without the user detecting any warning signs normally visible in the user's browser such that the attacker's methods can be hidden from the user. The attacker could use browser customization, such as enabling JavaScript, to create a shadow copy of the web and gain classified information from the victim without certain warning signs, such as URLs being correctly displayed. All user-entered information would be funneled through the attacker's spoofed world and thus the attacker could easily take advantage of the situation in order to retrieve any type of information.

Message Interception: Enron data set (publicly available) may be used to send emails to the partners of the coalition as well as to those outside of the coalition. Messaging may be simulated in such a way that they are sent at random intervals. We can then determine whether interception techniques can be used to extract some of the messages sent. This is a very challenging problem.

9. SERIES OF REPORTS

As we have stated in section 1, this paper is the first in a series of papers we will publish as part of the AIS. In this section we briefly discuss the contents of some of the other reports. Note that the research is sponsored by grants from different organizations including AFOSR.

Experimental Analysis: In this report we will discuss the experiments we are conducting on how much information ism lost by enforcing security policies in a coalition environment.

Game Theory Applications: In this report we will discuss the application of game theoretic techniques for extracting information when partners are semi-trustworthy.

Defensive Operations: In this report we will discuss our approach to defending the systems from worms when the partners are untrustworthy.

RBAC for AIS: In this report our partners at GMU will discuss the application of Role-based access control for assured information sharing.

Offensive Operations: In this report we will discuss techniques for finding out the activities of untrustworthy partners.

We are conducting research in related topics that will support AIS. Some related reports that we will publish include the following:

Risk-based access control: In this report we will discuss data sharing when taking security risks into consideration.

Data provenance: In this report, we will use healthcare applications as an example and discuss data provenance issues for AIS.

Dependable Data Sharing: In this report we will describe our approach to systems meeting security as well as real-time requirements.

Standards: In this report we will discuss data integration standards for AIS.

Privacy Preserving Data Sharing: In this report we will discuss data sharing and at the same time ensuring privacy of the individuals using healthcare applications.

Geospatial data: In this report we will discuss assured information sharing for geospatial and unstructured data.

Semantic web: In this report we will explore the use of semantic web technologies for AIS

Social network analysis: In this report we will examine how organizations form networks and discuss approaches for supporting AIS.

Infrastructure: In this report we will investigate how infrastructures such a data grids support AIS.

In addition to the above reports, we will also publish reports on the implementation of the designs of systems for AIS. For example, implementation of the systems we have designed for geospatial data sharing, risk-based access control and game theory applications will be described in future technical reports.

10. SUMMARY AND DIRECTIONS

In this paper we have defined Assured Information Sharing (AIS) and discussed issues, technologies, challenges and directions for this area. The goal of AIS is for organizations to share data but at the same time enforce security policies. Security includes confidentiality, privacy, trust, and integrity policies. We discussed approaches for AIS when the partners of a coalition are trustworthy, semi-trustworthy and untrustworthy. In particular, we discussed security policy enforcement, game theory applications and defending against worms and viruses. We also discussed AIS technologies including data integration, data mining, and the semantic web.

There are several areas that need further investigation. We need to develop policies for accountability. This is especially important in a coalition environment. In such an environment, there are numerous pieces of hardware and software that interact with each other. Therefore, the action of all the processes has to be recorded and analyzed. Furthermore, risk analysis studies are needed to determine the risks and developing appropriate solutions. For example, in a high risk low cost security environment, there will be no questions about implementing security solutions. However in a low risk high cost environment one needs to think twice before enforcing the security policies. Essentially we need some form of risk-based AIS. We also need to develop web services for AIS. Essentially we need to integrate AIS and semantic web technologies. Finally we need to investigate several additional technologies such as collaborative services, social network analysis, surveillance data sharing, digital identity management, metadata extraction and management as well as policies for identification and authentication for AIS. We also need to investigate the use of standards as well as infrastructures such ass data grids for AIS. Some of our preliminary research in some of these topics is reported in [THUR05b], [ZHU06], [LAVE05], [LAYF05].

We are conducting extensive investigation on AIS with our partners George Mason University and Purdue University. In addition to the technical aspects discussed in this paper, we are also investigating the connection between AIS and the Global Information Grid as well as Network centric Operations. While our primary application is counter-terrorism, we are also focusing on other applications such as Emergency preparedness and Healthcare. Future papers will focus on the design of our approaches as well as our experimental results for AIS.

ACKNOWLEDGEMENTS

I thank Dr. Robert Herklotz for funding our research on Information Operations Across Infospheres which supported much of the research discussed in this paper. I thank my colleagues Profs. Latifur Khan, Murat Kantarcioglu, Ravi Sandhu and Elisa Bertino as well as Dr. Mamoun Awad and Dr. Ebru Celikel for discussions and inputs on AIS. I also thank my students Ryan Layfield, Nathalie Tsybulnik, Li Liu, Alam Ashraful, Ganesh Subbiah, Gal Lavee, Srinivsan Iyer, Dilsad Cavus and Kim Jungin as well as many others for discussions on AIS, and especially Ryan Layfield, Nathalie Tsybulnik and Li Liu for writing the techniques for information operations in section 8.

REFERENCES

[AWAD06] M. Awad, B. Thuraisingham, and L. Khan, et al, Assured Information Sharing: Volume 2: Experimental Analysis of Data Integration, Mining and Security, Technical Report, The University of Texas at Dallas, 2006 (to appear)

[ASHR06] A. Ashraful, G. Subbiah, L. Khan, and B. Thuraisingham, Geospatial Semantic Web, Technical Report, The University of Texas at Dallas, 2006 (to appear).

[BERT04] E. Bertino, B. Carminati, E. Ferrari and B. Thuraisingham, *Secure Third Party Publication of XML Documents*, IEEE Transactions on Knowledge and Data Engineering, October 2004

[CELI06] E. Celikel, M. Kantarcioglu and B. Thuraisingham, Assured Information Sharing: Risk-based Data Sharing, Technical Report, The University of Texas at Dallas, 2006 (to appear)

[JONE80] A. Jones, Game Theory, Mathematical Models of Conflict, Halstead Press, 1980.

[KHAN06] L. Khan, B. Thuraisingham et al, Assured Information Sharing: Volume 4: Data Mining Applications for Defensive Operations in a Coalition, Technical Report, The University of Texas at Dallas, (to appear).

[KIM06a] J. Kim and B. Thuraisingham, Dependable and Secure TMO Scheme, Proceedings of IEEE ISORC Conference, April 006.

[KIM06b] J. Kim, B. Thuraisingham, et al, Data Provenance in Healthcare Systems: Survey and Research Issues, UTD Technical Report, to appear.

[KIM06] J. Kim and B. Thuraisingham, Applying RBAC and UCON to TMO, Technical report, University of Texas at Dallas, to appear.

[LAVE05] G. Lavee et al, Suspicious Event Detection with Surveillance Data, Proceedings of the ACM SIGKDD Conference Workshop on Multimedia Data Mining, 2005.

[LAYF05] R. Layfield, et al, Design of a Social Network Analysis System, Proceedings of the ACM SIGKDD Conference Workshop on Multimedia Data Mining, 2005.

[LAYF06] R. Layfield, M. Kantarcioglu and B. Thuraisingham, Assured Information Sharing: Volume 3: Using Game Theory to Enforce Honesty Within a Competitive Coalition, Technical Report, The University of Texas at Dallas, 2006 (to appear)

[LEE01] Berners Lee, T., et al., The Semantic Web, Scientific American, May 2001.

[LIU05] L. Liu, M. Kantarcioglu, N. Thuraisingham, L. Khan, An Adaptable Perturbation Model of Privacy Preserving Data Mining, Proceedings of the IEEE ICDM Data Mining Conference Workshop on Privacy preserving Data Mining, 2005 (also published as technical report, UTDCS-03-06, January 2006).

[LIU06] L. Liu, et al, Privacy Preserving Data Sharing, Technical Report, The University of Texas at Dallas, 2006 (to appear)

[MARK03] Creating a Trusted Network for Homeland Security, Markle Report, 2003 (Editor: M. Vatis)

[MASU06] Masud, M, L. Khan, B. Thuraisingham and M. Awad, Detecting New malicious Executables Using Data Mining, UTDCS-27-06 Technical Report, The University of Texas at Dallas, June 2006, also submitted for publications. (version to be published as UTD AIS Technical Report series)

[NCW05] The Implementation of Network Centric Warfare, Office of Force Transformation, 2003.

[OLIV95] Martin S. Olivier: Self-protecting Objects in a Secure Federated Database, Proceedings of the IFIP Database Security Conference, NY, August 1995.

[SAND96] Ravi Sandhu, Edward Coyne, Hal Feinstein and Charles Youman, "Role-Based Access Control Models." *IEEE Computer*, Volume 29, Number 2, February 1996.

[SAND06] R. Sandhu et al, RBAC for AIS, to be published as AIS Technical Report Series, 2006.

[SIGN05a] Signal Magazine, AFCEA, May 2005

[SIGN05b] Signal Magazine, AFCEA, February 2005

[SPIT02] Lance Spitzner, Honeypots, Tracking Hackers, Addison Wesley, 2002.

[SON95] S. Son, R. David and B. Thuraisingham, *An Adaptive Policy for Improved Timeliness in Secure Database Systems,* Proceedings of the 9th IFIP Working Conference in Database Security, New York, August 1995.

[THUR90] B. Thuraisingham, *Novel Approaches to the Inference Problem*, June 1990, Proceedings of the 3rd RADC Database Security Workshop, New York.

[THUR94] B. Thuraisingham, *Security Issues for Federated Database Systems,* 1994, Computers and Security (North Holland), December 1994.

[THUR98] B. Thuraisingham, Data Mining: Technologies, Techniques, Tools and Trends, CRC Press, December 1998.

[THUR99] B. Thuraisingham and J. Maurer, *Information Survivability for Real-time Command and Control Systems*, IEEE Transactions on Knowledge and Data Engineering, January 1999

[THUR03] B. Thuraisingham, Web Data Mining and Applications in Business Intelligence and Counter-terrorism, CRC Press, Boca Raton, FL, 2003.

[THUR05a] B. Thuraisingham, Security Standards for the Semantic Web, Computer Standards and Interfaces Journal, 2005.

[THUR05b] B. Thuraisingham, Database and Applications Security: Integrating Information Security and Data Management, CRC Press, May 2005

[THUR06] B. Thuraisingham, D. Harris, L. Khan, R. Paul, "Standards for Secure Data Sharing across Organizations," Accepted in Computer Standards and Interfaces Journal, 2005. (version to be published as part of UTD AIS technical report series)

[TSYB06] N. Tsybulnik, B. Thuraisingham, A. Ashraful, CPT: Confidentiality, Privacy and Trust for the Semantic Web,UTDCS-06-06, Technical Report, the University of Texas at Dallas, March 2006, Also to appear in the Journal of Information Security and Privacy.

[ZHU06] J. Zhu, B. Thuraisingham, Grid Computing and Grid Security, Technical; Report, The University of Texas at Dallas, to appear. (also published in International Journal of Computer and Network Secuirty, August 2006).

Report #2

Design and Implementation of Policy Enforcement, Data Sharing and Mining Components for Trustworthy Coalitions

Mamoun Awad, Latifur Khan, Dilsad Cavus, Bhavani Thuraisingham

The University of Texas at Dallas

Published as Technical Report UTD-CS-44-06

Abstract

Sharing data among organizations plays an important role in security and data mining. In this study, we present Data Sharing Miner and Analyzer (*DASMA*) system that simulates *N* organizations. Each organization has its own enforced policy. The *N* organization share their data based on trusted third party. The system collect the released data from each organization, process it, mine it, and analyze the results. Mining is based on applying Association Rule Mining. Analyzing is the process of measuring the differences between mining with enforced policy and mining the data as a whole.

Sharing in DASMA is based on trusted third parties. However, organization might choose to encode some attributes, for example, by categorizing numeric data instead of providing the exact data values. Also an organization can provide randomized data in which random values for some attributes are added. Each organization has it own policy represented in XML format. The policy states what attributes they can release, what attributes they need to encode, and what attributes they need to randomize. DASMA processes the data set and collect the data, combine it, and prepare it for mining. After mining a statistical report is produced stating the similarities between mining with data sharing and mining without sharing.

We test, apply data sharing, enforce policy, and analyze the results of two separate datasets in different domains. Our results indicate a fluctuation on the amount of information loss using different releasing factor.

1. Introduction

Data sharing among organization has become a critical research topic. Sharing data among organization is governed by the sharing policies maintained and enforced by the organization rules and by the government laws. As a result of that, the amount of information used, in any certain sharing scenario among organization, is smaller than or equal to the whole information maintained in all such organizations.

In this research, we study the effect of information hiding on the amount of knowledge obtained using standard machine learning techniques. Hiding information is represented by the policies and regulations enforced by the organization. We introduce the *releasing factor* measure that indicates the percentage of attributes an organization releases to the total number of attributes such organization has. For mining the shared data , we consider Association Rule Mining.

It is important to point out that, in this study, we assume that all organizations are trusted parties. However, each organization abides by its policies and rules in order to release data. For each organization, we develop sharing policies that govern what kind of data an organization can release. For example, a medical organization, can release information about blood pressure and temperature of patients. However, it cannot release type of illness each patient has.

Also, we try to simulate a realistic scenarios of data partitioning. For example, for a specific entity, such as patient, one organization, such as the hospital, might have attributes/fields about the patient medications. However, for another organization, such as insurance companies, such fields are missing. We consider three different partitioning of the attributes, namely, horizontal, vertical, and hybrid partitioning. In horizontal partitioning, we simulate the scenario in which one organization has all fields/attributes about some entities. In vertical partitioning, an organization knows all entities, however, it has some of the fields/attributes about each. In hybrid partitioning, we assume horizontal and vertical knowledge about entities and attributes/fields, i.e., some entities are know totally or partially by some organizations. Notice that data partitioning is related to the layout of the data (see Section 2 for details). *It is also important to point out that we assume that there is a fixed set of attributes/fields about entities.*

After partitioning the dataset, we assume a centralized trust broker, which request the information from different parties and mine the data. When the broker requests data from an organization *x*, organization *x* will apply its policy first, and then send a compliant data, with *x* policy, to the broker (See Figure 1).

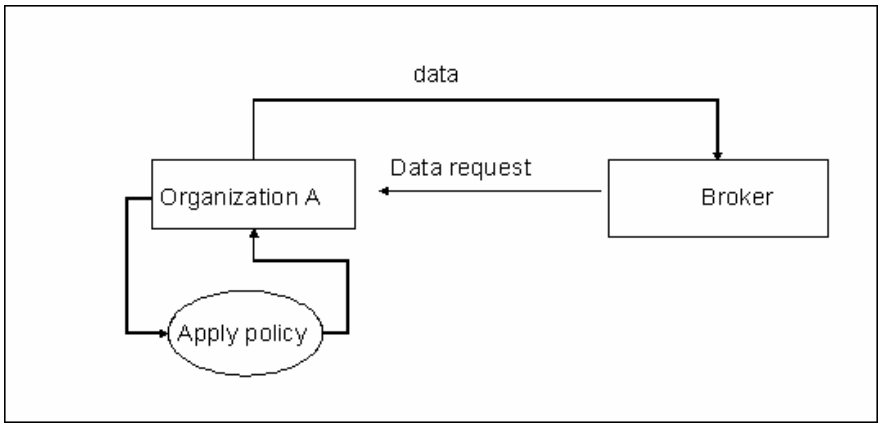


Figure 1 communications between the broker and an organization

The process of mining shared data among organization poses several challenges related to the automation of data sharing. First, data disclosure might not be possible because organizations are limited to their sharing policy, i.e., an organization might not release all the data that it has because, for example, of privacy issues. Next, data reprocessing as a result of discrepancies of the format, representation, scales, etc. of the data among organizations. Finally, human intervention to resolve issues such as mapping data from one organization data base to another. That is because it is possible that two attributes has the same names, however, different meaning and vice versa.

In this study, we randomly partition the data (using three schemes, namely, horizontal, vertical, and hybrid partitioning) among N organizations. In order to resolve the issue of data disclosure, we create several test cases with increasing releasing factor. For example, we create 25 test cases 1, 2, 3, 4, …,25 and applied releasing factors 1, 4, 8, 12, …,100 for each test case, respectively. We do not try to solve the problem of mismatch in the formatting of the data among organizations; however, we introduce an xml based framework that can be extended later to handle such mismatch.

The report is organized as follows. Section 2 presents the data partitioning schemes we use. Section **Error! Reference source not found.**

2. Data Partitioning

2.1 Overview

In this section, we describe and explain the different schemes of partitioning data. Since we do not have real time shared data, we try to partition a dataset among N imaginary organizations. There are several ways to share the data. The data is distributed among different organizations with three different scenarios. We discuss the various types of partitioning in the ensuing subsections.

2.2 Horizontal Partitioning

In the horizontal partitioning an organization has all records/information about some entities (such as persons). Figure 2 presents an example of such partitioning in which we have four organizations sharing information about persons. The first row in Figure 2 s the name of attributes/fields about people, for example, here we have the following attributes: SSN, data of birth, credit history, annual income, occupation, and auto insurance company. The rest of the rows represent records about different persons. Notice that each organization has all the attributes/fields about some persons; however, it lacks information about other persons in the dataset. Also, in this partitioning, some records can be redundant in more than one organization. One realistic scenario of such partitioning is the case in which department of vehicles in Texas has all the records about people in Texas, however, it does not have any of such information about people in California and vice versa.

	SSN	Dob	Credit history	Annual income	Job	Insurance
1	123-…	1-8-90			Engineer	
2	321-…	2-1-02			teacher	
3						
4						
5	143-…	1-1-62			Artist	
6	325-…	1-1-82			Lawyer	
7						
8						
9	223-…	1-1-92			Professor	
10	521-…	1-1-72			driver	

Org 1 (rows 1–2), Org 2 (rows 5–9), Org 3 (rows 3–6), Org 4 (rows 7–10)

Figure 2 Horizontal partitioning

2.3 Vertical Partitioning

In vertical partitioning each organization has part of the fields/attributes about some entity in the dataset. For example, in a dataset of patient records, an insurance organization knows specific attributes about the patients such as their ids, cost of treatment, drug used, duration of treatments. However, such insurance organization has no specific information about the illnesses, readings (blood pressure, sugar level, etc.). Figure 3 presents an example of vertical partitioning for the same dataset used in Figure 2. Notice that some organization such as organization 1 has only information about SSN and date of birth. Organization 4 has information about the annual income, occupation, and the auto insurance company. In real time scenario, there might be different ID used for each person. For example, Department of Motor Vehicles might use driver license to identify a person, on the other hand FBI might have both driver license and SSN and use both to identify persons. In this study, we assume that there is a unique id that identifies each person, and all organizations are using this id.

Org 1 Org 2 Org 3 Org 4

	SSN	Dob	Credit history	Annual income	Job	Insurance
1	123-…	1-8-90			Engineer	
2	321-…	2-1-02			teacher	
3						
4						
5	143-…	1-1-62			Artist	
6	325-…	1-1-82			Lawyer	
7						
8						
9	223-…	1-1-92			Professor	
10	521-…	1-1-72			driver	

Figure 3 Vertical Partitioning

2.4 Hybrid Partitioning

Hybrid partitioning is the combination of vertical and horizontal partitioning. This means that an organization might have a complete record about some entities, however, it lacks some of the attributes about others. Such lack of information presents optional information, such as race, age, etc. So, it is up to the person to provide it or not; hence, it might be available or unavailable.

Org 1 / Org 2 / Org 3 / Org 4

	SSN	Dob	Credit history	Annual income	Job	Insurance
1	123-…	1-8-90			?	
2	321-…	?			Teacher	
3			?	?		
4			?			?
5	?…	1-1-62			Artist	
6	325-…	1-1-82			?	
7			?			
8						
9	?…	1-1-92			Professor	
10	521-…	1-1-72			driver	

Figure 4 Hybrid Partitioning

Figure 4 present an example of hybrid partitioning of a dataset among four organizations. The question mark in Figure 4 denotes unknown field value. Notice that some records are complete in some organization, for example, organization 4 has all the fields of record number 10. However, organization 4 lacks some attributes, such as occupation, of record number 6.

3. Policy Representation and Enforcement

3.1 Overview

In this section, we present the representation of organization policy and the details of enforcing them. But before we present that, we introduce some terminology and definitions.

Policy

In this section, we refer to policy as an xml document that is mainly designed to inform which attributes can be released. Figure 5 presents a policy of an organization in which such organization can only release the attributes of type of employment, county of birth, and income type for each record it has about people. Such scheme can be extended to include encoded data or randomized data, i.e., what are that attributes that we should encode them (using randomization) before releasing them.

```
<?xml version="1.0"?>
  <DATASET>
   <RELEASE_ATTR>
     <REL_ATTRIB>full_or_part_time_employment</REL_ATTRIB>
     <REL_ATTRIB>country_of_birth_mother</REL_ATTRIB>
     <REL_ATTRIB>income_type</REL_ATTRIB>
    </RELEASE_ATTR>
   </DATASET>
```

Figure 5 example of xml policy

Release Factor

The release factor is the percentage of attributes which are released from the dataset by an organization. For example, assume we have a dataset that has 40 attributes and "Organization 1" releases 8 attributes. The release factor in this case is 8/40=20%. Here, we use the release factor to analyze the amount of information loss as a result of sharing and applying policies.

3.2 POLICY STRUCTURE AND DETAILS

In this section, we present the structure of the xml files that we used to configure partitioning and policy enforcement.

```
<?xml version="1.0"?>
  <ORGANIZATIONS>
      <ORGANIZATION>
   <ORG_ID>1</ORG_ID>
   <XML_POLICY_FN>org_1.xml</XML_POLICY_FN>
  </ORGANIZATION>
  <ORGANIZATION>
   <ORG_ID>2</ORG_ID>
   <XML_POLICY_FN>org_2.xml</XML_POLICY_FN>
  </ORGANIZATION>
  <ORGANIZATION>
   <ORG_ID>3</ORG_ID>
   <XML_POLICY_FN>org_3.xml</XML_POLICY_FN>
  </ORGANIZATION>

  <NUM_ORG>3</NUM_ORG>
  <DATASET_FN>census_income/census_income_50K.dat</DATASET_FN>
  <ARFF_PREFIX>census_income</ARFF_PREFIX>
  <TEST_CASE_ID>census_income_test_20_1</TEST_CASE_ID>
  <DATASET_PROCESSOR>
    <CLASS_NAME>processors.CensusIncomeProcessor</CLASS_NAME>
      <ATTRIB_FN>census_income/attributes.xml</ATTRIB_FN>
  </DATASET_PROCESSOR>
  <POLICY_DIR>census_income_20/testcases/test_case_1</POLICY_DIR>
  <DELIM>,</DELIM>
</ORGANIZATIONS>
```

Organization information

Data set & Processing data

Figure 6 an example of an xml file used for partitioning and sharing.

Figure 6 present an example of partitioning a dataset among three organizations. In the top part of the xml file, noted by organization information, we state the number of organizations, and each organization id and its releasing policy. Each releasing policy file is an xml file which is similar to Figure 5, in which we define the attributes/fields that we can release.

The second part of the xml in

Figure 6 is a details about the dataset file, dataset preprocessor, and meta data about that. **Table 1** presents the name and definition of each xml tag in the configuration xml file.

Table 1 Xml tags names and their definitions.

Xml tag	**Definition**
ORG_ID	Unique organization id.
XML_POLICY	The xml file that contains policy, see Figure 5.
NUM_ORG	The number organization involved in data sharing.
Dataset Information	
DATASET_FN	The dataset file name including its path.
ARFF_PREFIX	The arff prefix file name.
TEST_CASE_ID	A unique id for the test case generated.
Dataset Processor	
CLASS_NAME	Java class name of the processor dedicated for this dataset.
ATTRIB_FN	The file name that contains all the attributes in this dataset along with their types (for example, nominal, binary, etc.)
POLICY_DIR	The directory where the xml policies of the organizations reside in.
DELIM	The delimiter used to separate fields in the dataset.

Notice that we randomly assign a record to an organization. Once all records are assigned to organizations, we start applying the policy of that organization. In the GUI, the user can choose the type of data partitioning (see Figure 7). Notice that the use use the browse xml button to find his policy file, then from the drop box, the user can choose either horizontal, vertical, or hybrid partitioning.

Notice that in the previous xml configuration file, we did not mention the releasing factor. That is because each test case is a single test case that is specific for a set of attributes provided by the user. Alternatively, the user can choose batch processing, in which she chooses the releasing factor step and the GUI generates a set of test cases. In other words, the system will generate a set of xml files similar to

Figure 6. In this study, we adopt the batch processing because, first, we need large number of test cases. Next, each test case should represent different releasing factor. Finally, we need to see the effects of different releasing factor on the information loss.

Figure 8 presents an example of xml file used in batch processing. In addition to the xml tags Figure 6, we added few xml tags to indicate the releasing factor step, template information for xml generation, and policy directory to build the directory structure of the test cases (See Table 2). Notice that the RELEASE_FACTOR indicates the step of increasing the number of attributes. For example, if the RELEASE_FACTOR is 10, that means we will generate 10 test cases from 1

to 10 and the number of attributes released for all organization in these test cases is 10, 20 , 30, 40, 50, …, 100%, respectively. The MANDATOY_ATTRIB is the list of attributes that all organizations have to release. This is very convenient because it allows the system to enforce a unique id to be released; hence, it can connect different attributes from different organizations and remove duplicates. Figure 9 presents the list of test cases generated from
Figure 8.

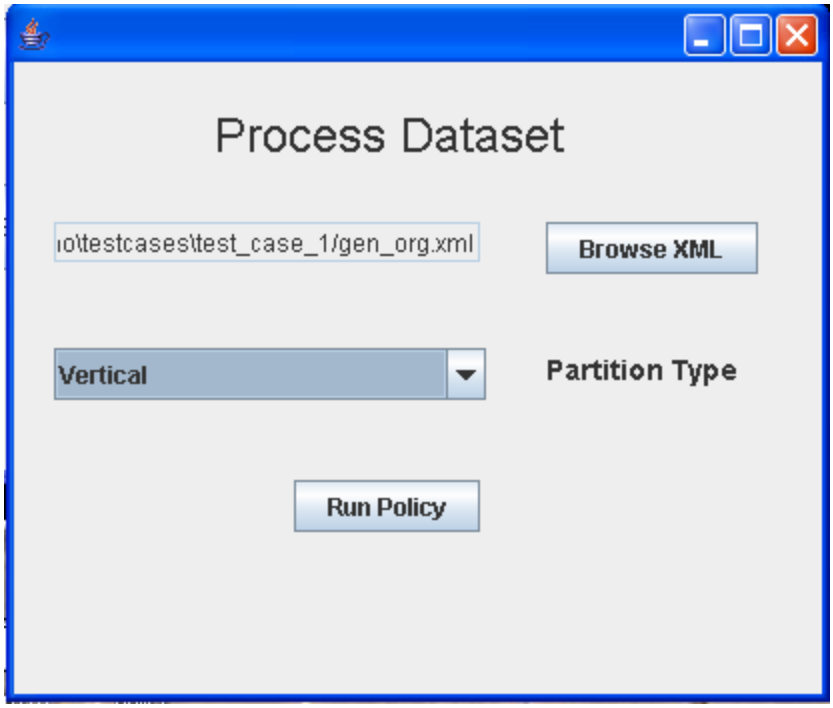


Figure 7 Processing dataset and policy enforcement GUI

4. Data Sharing

4.1 Overview

The data sharing happens among N organizations with three different partitionings (horizontal, vertical and hybrid). The dataset is divided into N parts randomly so that each organization has its own dataset. We used 3 different organizations for each test case, so that each partition uses its own partitioned dataset. i.e If the dataset's file name is dataset.dat, the dataset will be randomly distributed into three data files and named horizontal_dataset_org1.dat, horizontal_dataset_org2.dat and horizontal_dataset_org3.dat for horizontal partitioning. The same dataset will again be partitioned for vertical and hybrid partitionings.

4.2 Using the Policies:

Our gui uses several policies to get information. It starts with a policy named <*dataset_file_name*>_<*release_factor*>.xml. i.e census_income_10.xml

There are two parts in the policy. The first part of the xml policy shows how to name the directories and prefixes to eliminate overwriting, and uses some policies about the information that is needed to process the dataset. The second part is related to batch processing and partitioning. So we are going to explain the first part which is not related to the programming part. Before starting to explain the elements of the xml policy, let's assume that we have created a new project file named NewProj under H directory. The command prompt looks like this:
H:/**NewProj**

We will create new folders to save what we process and keep them in an understandable hirerarchy. i.e. The files related to dataset will be under the "dataset" folder, and the policies will be under the "policy" folder.
Now, let's get started with the XML file shown above. The first line of the file is an XML declaration.

```
<?xml version="1.0"?>
```

```
<?xml version="1.0"?>
 <TEST_CASE>
  <BASE_POLICY_DIR>/data/policy/</BASE_POLICY_DIR>
  <!-- make sure to have different tc_id for the bundle -->
  <TC_ID>census_income_4</TC_ID>
  <TEST_CASE_DIR>testcases</TEST_CASE_DIR>
  <NUM_ORG>3</NUM_ORG>
  <RELEASE_FACTOR>4</RELEASE_FACTOR>
  <ATTRIB_XML>attributes.xml</ATTRIB_XML>
  <DATASET_BASE>/data/dataset/census_income/</DATASET_BASE>
  <MANDATORY_ATTRIB>income_type</MANDATORY_ATTRIB>
  <POLICY_XML>gen_org.xml</POLICY_XML>
  <ORG_PREFIX>org_</ORG_PREFIX>

   <!-- information about the dataset -->
   <DATASET_FN>census_income/census_income_50k.dat</DATASET_FN>
   <ARFF_PREFIX>census_income</ARFF_PREFIX>

   <!-- for each testcase bundle, used different test_case_id -->
   <TEST_CASE_ID>census_income_test_4</TEST_CASE_ID>
   <DATASET_PROCESSOR>
    <CLASS_NAME>processors.CensusIncomeProcessor</CLASS_NAME>
    <ATTRIB_FN>census_income/attributes.xml</ATTRIB_FN>
   </DATASET_PROCESSOR>
   <POLICY_DIR>census_policy_</POLICY_DIR>
   <DELIM>,</DELIM>
   <TEMPLATE_FN>gen_template.xml</TEMPLATE_FN>
 </TEST_CASE>
```

Figure 8 xml file used in batch processing.

Table 2 additional xml tags used in batch processing

Xml tag	Definition
RELEASE_FACTOR	The step in which we increase the releasing factor for the next test case.
MANDATORY_ATTRIB	Mandatory attributes to be included in the releasing policy of all organizations.
TEMPLATE_FN	A template file used to generate organization policies.
TC_ID	Unique batch test case id.
TEST_CASE_DIR	The test case directory name.

Every XML file starts with an "XML declaration", which indicates several pieces of information that is used to parse the file. "1.0" is the XML version number which is the only official version of the XML specification.
The policy starts in the second line with "**TEST_CASE**" element. This element is the root of the tree which contains all the information in the file.

The **BASE_POLICY_DIR** element contains the name of the directory/ies which is going to be created under our project. The command prompt line will look like this.
H:/NewProj/**data/policy/**
The next line specifies some comments using **<!—**and --> syntax.

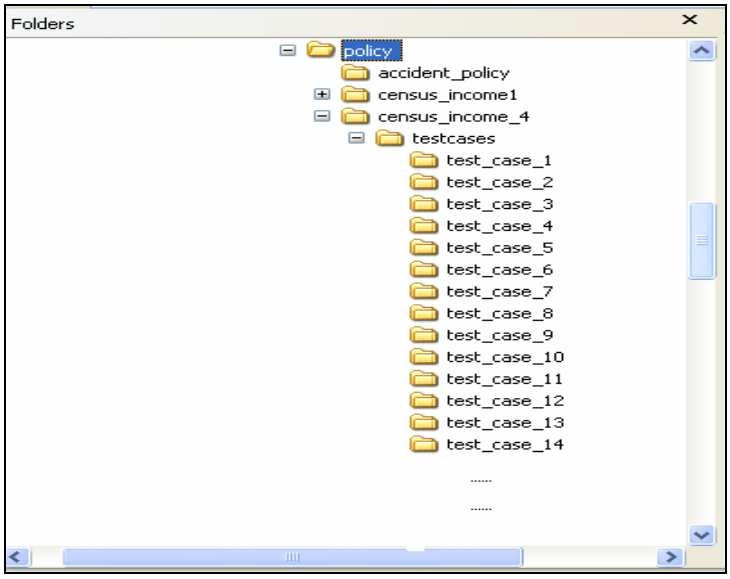


Figure 9 generated test cases for Figure 8

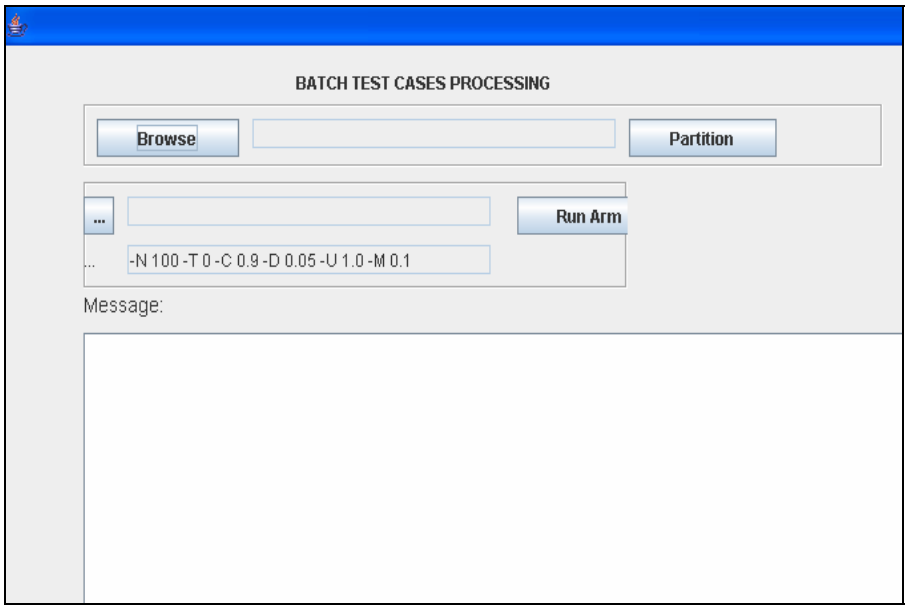


Figure 10 batch processing GUI

TC_ID is the test case id which is related to the xml policy file name. This folder is created under BASE_POLICY_DIR. The preferred name for the test case id is *"dataset name"_"release factor number"* as shown in **Error! Reference source not found.**.
TEST_CASE_DIR contains the name of another folder which is created under TC_ID.
NUM_ORG keeps the number of organizations and **RELEASE_FACTOR** keeps the release factor number.

ATTRIB_XML contains each column name and its potential values in the dataset. The sample example for this policy is seen in Sample Policy (Figure).

```
<?xml version="1.0"?>
<ATTRIBUTES>
  <ATTRIBUTE>
    <ATTRIB_NAME>gender</ATTRIB_NAME>
    <ATTRIB_TYPE>Female;Male</ATTRIB_TYPE>
    <ATTRIB_NAME>age</ATTRIB_NAME>
    <ATTRIB_TYPE>0:100</ATTRIB_TYPE>
  </ATTRIBUTE>
</ATTRIBUTES>
```

Figure 11. Sample Policy

Here, **ATTRIB_NAME** gives us the name of the attribute, **ATTRIB_TYPE** gives the attribute values. The consecutive values are separated by ":", the other types are separated by ";"
Let's go back to our first XML policy. The element of the next line, **DATASET_BASE** has the name of a new set of folders which is created under the "NewProj" directory. This directory contains attributes.xml and the original dataset.
MANDATORY_ATTRIB keeps the name of the attribute which is used by all the organizations. If there are more than one mandatory attributes, this element can be used more than once.
POLICY_XML will be explained after the definition of **TEMPLATE_FN** element.
The number of organizations are given in NUM_ORG, let's say it's 3. There will be 3 organizations which releases attributes. These organizations must have different names. The naming is generated by the program using a prefix which is given in **ORG_PREFIX** element. The new xml files will be named like this, `org_1, org_2, .. org_N`
DATASET_FN contains the original dataset file name.
ARFF_PREFIX is the prefix of a folder name which is generated for different test cases. The generated test case and its number is attached as a suffix to this name.
As we said before, part 2 is related to the programming part, however the element named **TEMPLATE_FN** has the XML file name which helps the program generate the "gen_org.xml" defined in **POLICY_XML** element.

Here's an example for TEMPLATE_FN file. (Figure 12)

```
<?xml version="1.0"?>
  <ORGANIZATIONS>
    %ORGANIZATIONS%
    <NUM_ORG>%NUM_ORG%</NUM_ORG>
    <DATASET_FN>%DATASET_FN%</DATASET_FN>
    <ARFF_PREFIX>%ARFF_PREFIX%</ARFF_PREFIX>
    <TEST_CASE_ID>%TEST_CASE_ID%</TEST_CASE_ID>
    <DATASET_PROCESSOR>
      <CLASS_NAME>%CLASS_NAME%</CLASS_NAME>
         <ATTRIB_FN>%ATTRIB_FN%</ATTRIB_FN>
    </DATASET_PROCESSOR>
    <POLICY_DIR>%POLICY_DIR%</POLICY_DIR>
    <DELIM>%DELIM%</DELIM>
  </ORGANIZATIONS>
```

Figure 12. Example Template

gen_org.xml files in all of the test case folders will be a copy of this file except the values between "%". Those values will be generated by the program. The sample for POLICY_XML file is shown in Figure 13.

```
<?xml version="1.0"?>
  <ORGANIZATIONS>
        <ORGANIZATION>
     <ORG_ID>1</ORG_ID>
     <XML_POLICY_FN>org_1.xml</XML_POLICY_FN>
    </ORGANIZATION>
    <ORGANIZATION>
     <ORG_ID>2</ORG_ID>
     <XML_POLICY_FN>org_2.xml</XML_POLICY_FN>
    </ORGANIZATION>
    <ORGANIZATION>
     <ORG_ID>3</ORG_ID>
     <XML_POLICY_FN>org_3.xml</XML_POLICY_FN>
    </ORGANIZATION>

    <NUM_ORG>3</NUM_ORG>
    <DATASET_FN>census_income/census_income_50k.dat</DATASET_FN>
    <ARFF_PREFIX>census_income</ARFF_PREFIX>
    <TEST_CASE_ID>census_income_test_10_1</TEST_CASE_ID>
    <DATASET_PROCESSOR>
      <CLASS_NAME>processors.CensusIncomeProcessor</CLASS_NAME>
        <ATTRIB_FN>census_income/attributes.xml</ATTRIB_FN>
    </DATASET_PROCESSOR>

<POLICY_DIR>census_income_10/testcases/test_case_1</POLICY_DIR>
    <DELIM>,</DELIM>
  </ORGANIZATIONS>
```

Figure 13. Policy File

As it is seen above, some of the elements are rewritten due to the number of organizations.

5. System Manual

MainGUI:

The Main GUI Chooser window is used to launch PD&M graphical environments. Main Window has three buttons:
1. Load and Analysis. Provides a simple GUI interface that allows loading the already generated rules and analyze rules by displaying the charts.
2. Run ARM. Provides an interface to choose the arff file and run Apriori algorithm, and displays the association rules, frequent item sets and their confidence.
3. Process DataSet: Provide the chooser window to select Single Processing dataset or Batch Processing.

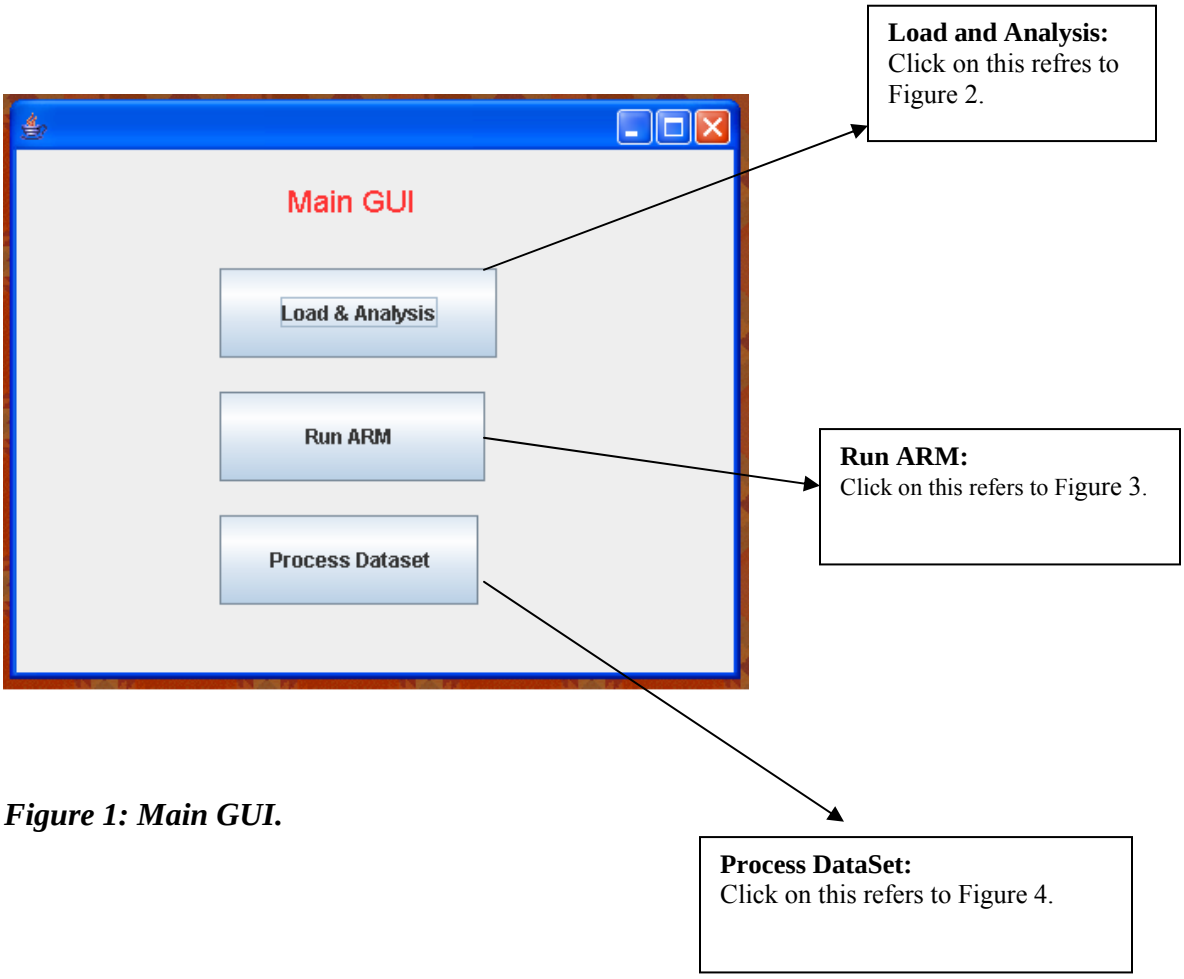


Figure 1: Main GUI.

Load And Analysis:

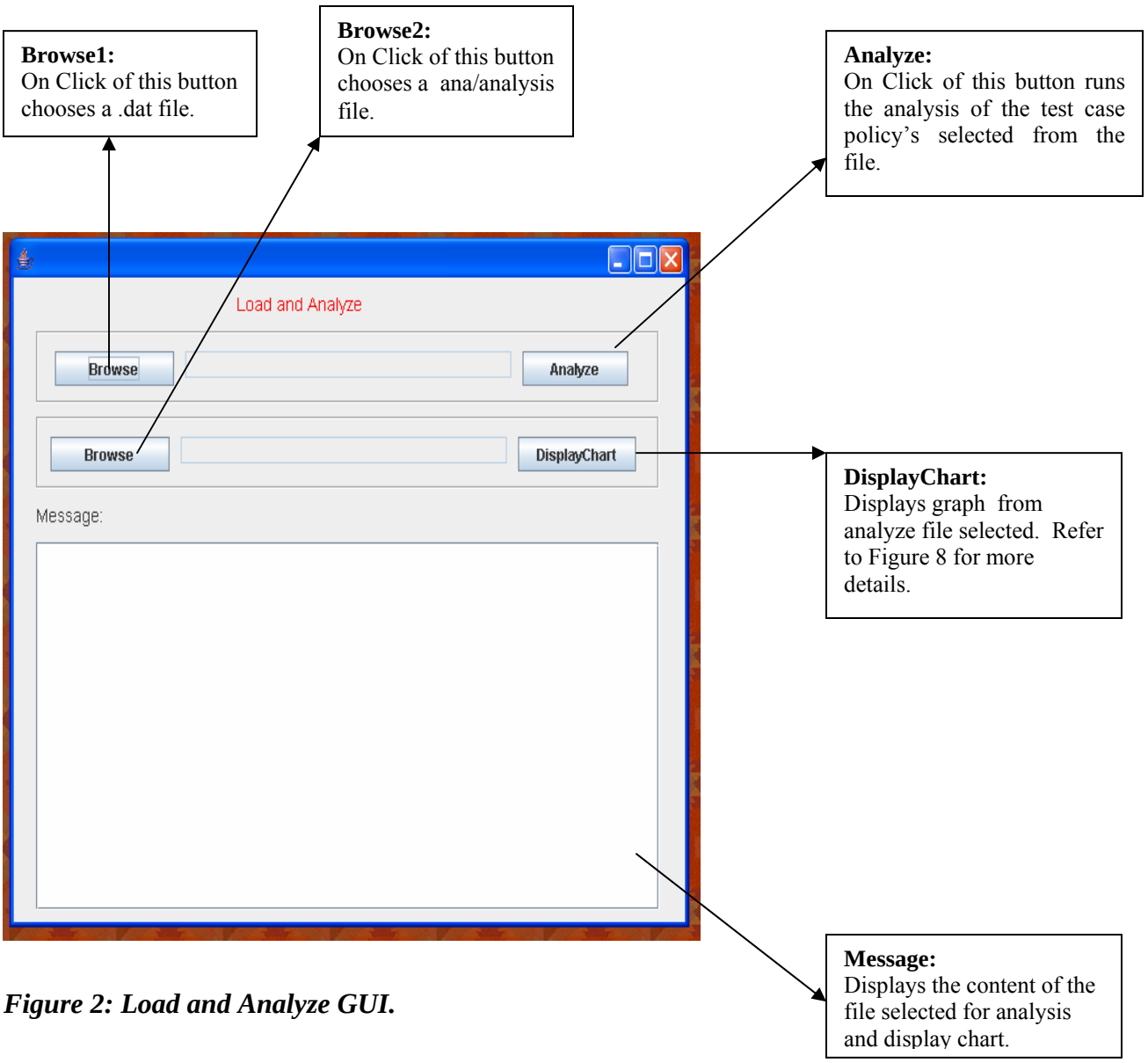


Figure 2: Load and Analyze GUI.

RunArm:

This GUI allows a user to select the preprocessed .arff file and select the options required for the association rules(for more about the options refer to figure 7). Buy clicking **Run ARM** button the **Apriori** association rule algorithm is applied to the selected file with options selected.

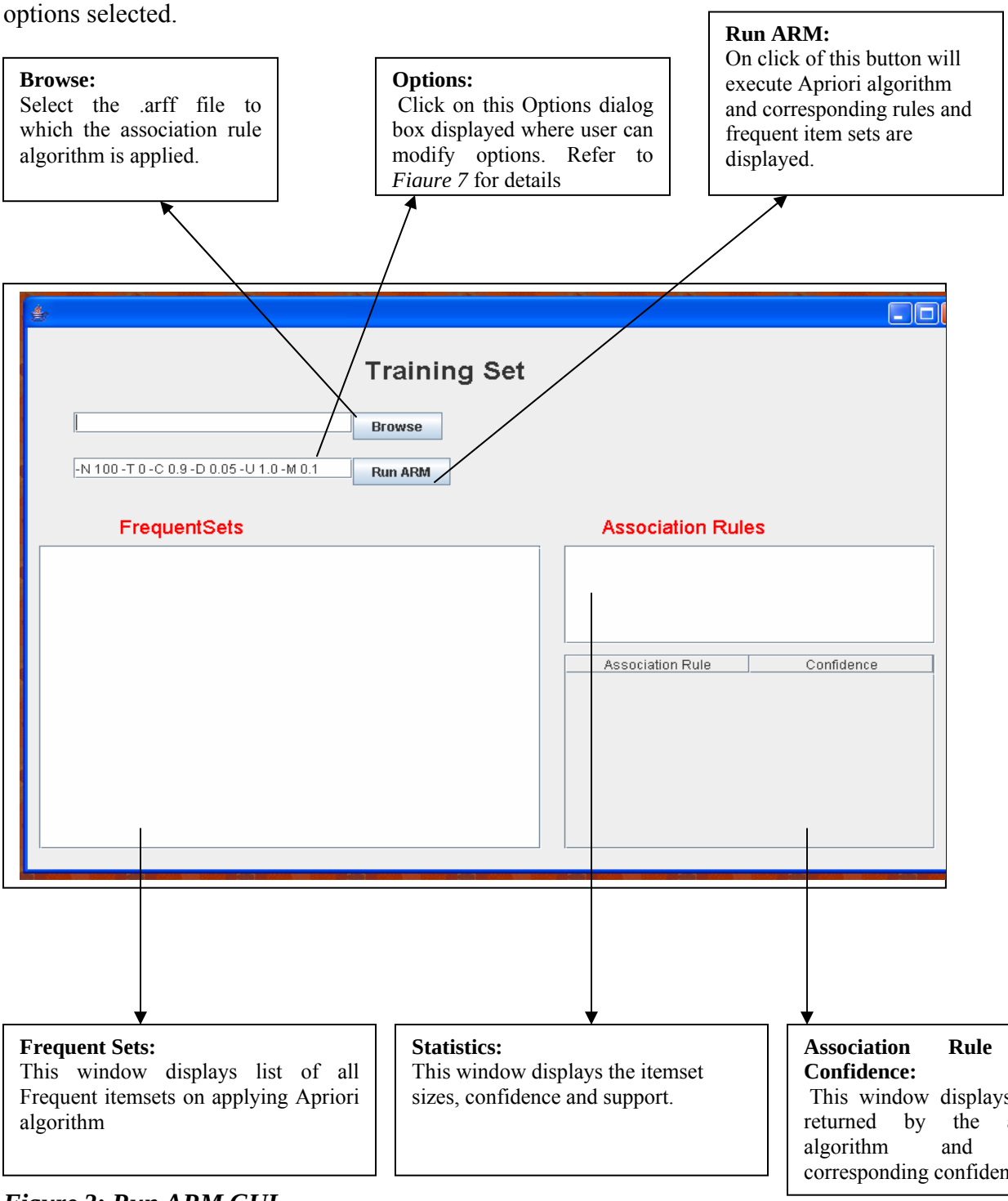


Figure 3: Run ARM GUI.

Main Process Data Set:

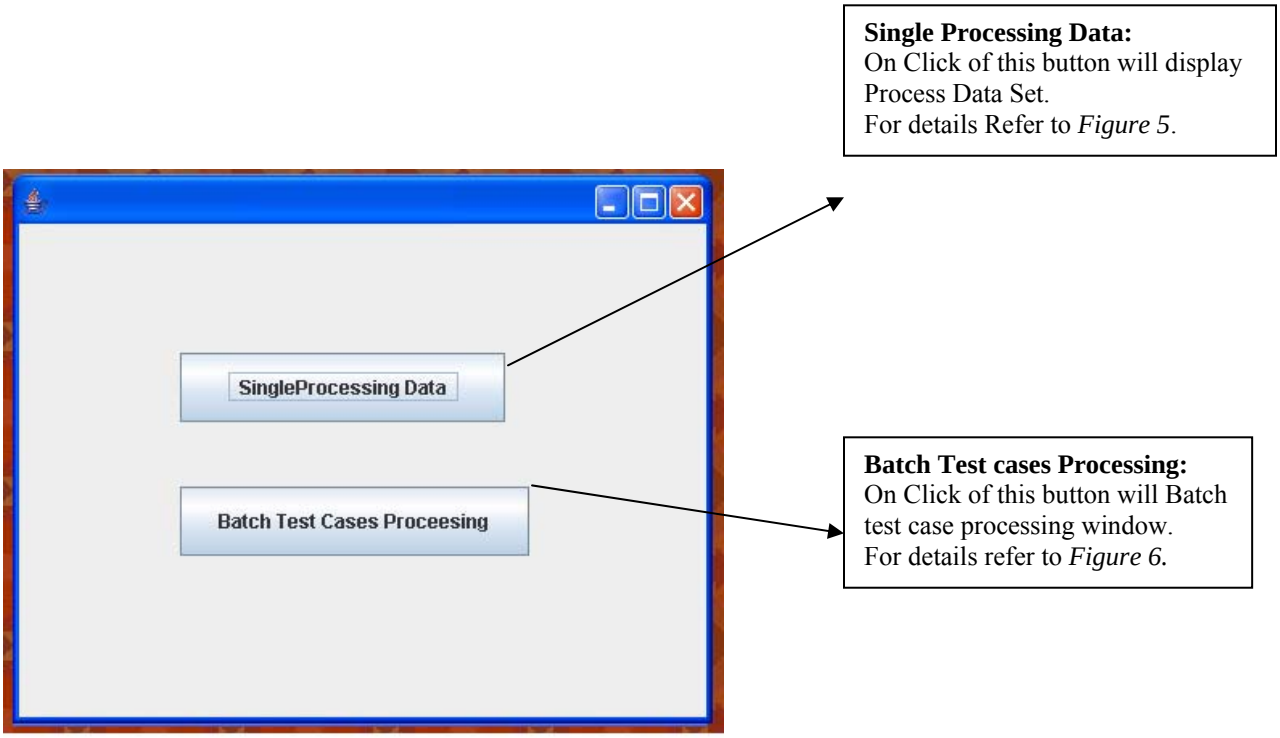
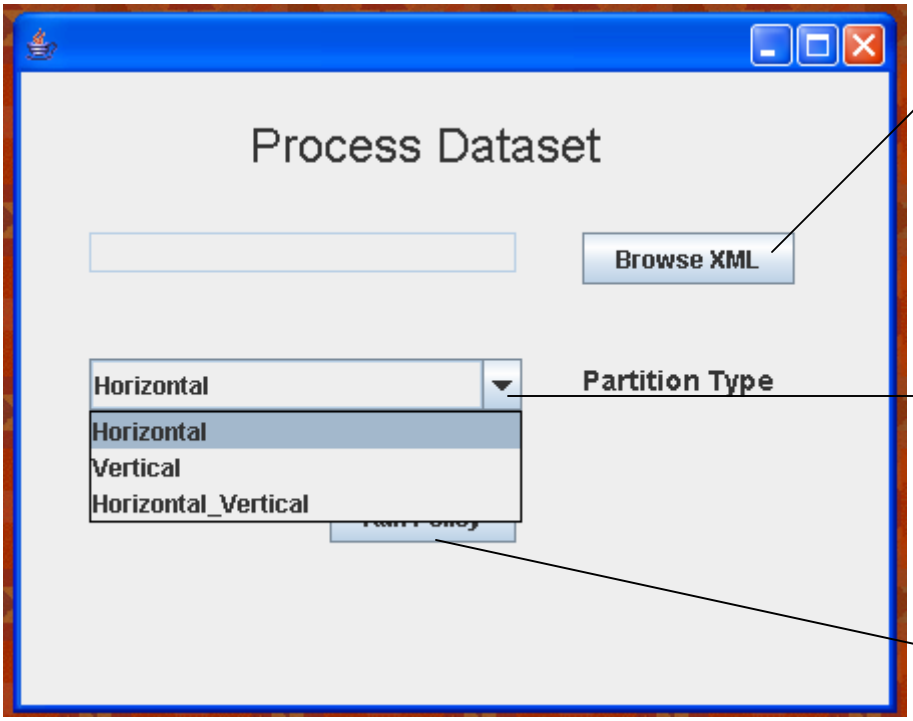


Figure 4: Main Process Data Set GUI.

Process DataSet:



Browse XML:
Click on this button select the xml file to be processed.
Refer to xml schema from section 1.4.

Partition Type:
Select the type of partition: Horizontal or Vertical or Horizontal and Vertical.
Refer section 1.2 for more details on options.

Run Policy:
Click on this button runs the policys based on xml file selected and partition type.

Figure 5: Process Dataset GUI.

Batch Test Cases Processing:

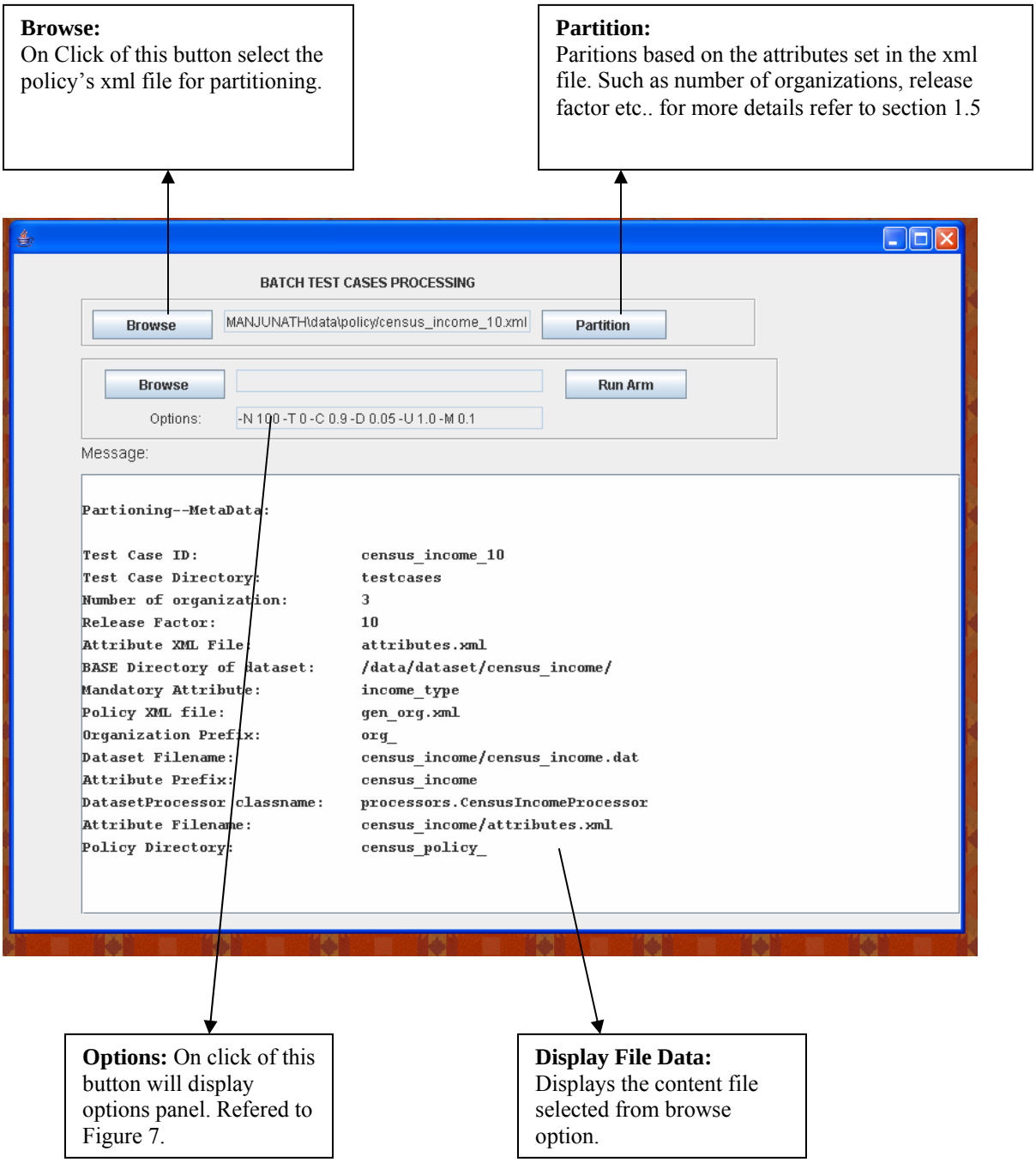


Figure 6: Batch Processing GUI.

Options:

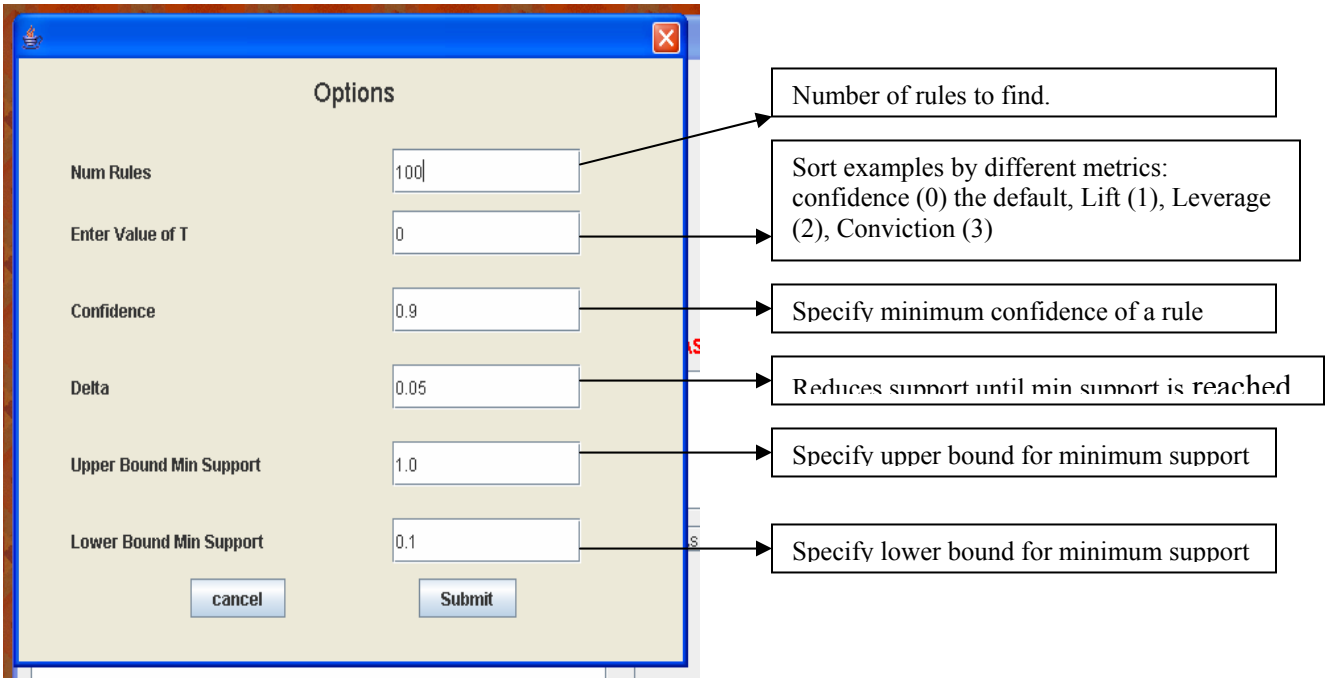


Figure 7: Options Panel

Acknowledgements: *We thank Manjunath Reddy and Srinivasan Iyer for their contributions to the project; in particular for the user interface implementation and Oracle expertise.*

Report #3

Design and Simulation of Agent-based Trust Management Techniques for a Coalition Environment

Srinivasan Iyer and Bhavani Thuraisingham

The University of Texas at Dallas

Published as Technical Report UTD-CS-45-06

Abstract

Effective communication among agents in large teams is crucial because the members share a common goal but only have partial views of the environment. Information sharing is difficult in a large team because, a team member may have a piece of valuable information but not know who needs the information, since it is infeasible to know what each other agent is doing. Information sharing is a main part of any system or organization. Even in the times of kings and Empires, Information sharing was unavoidable. There were many alliances between the Kingdoms, Espionages, miscommunications, treachery, deception, compromises, Victories and Defeats. The information sharing needs to be foolproof. Only the legitimate receiver should be able to get hold of the information. Even Kings had their own way of secured Information sharing. They had the royal seals to verify if the information is authentic. This paper mainly deals with intelligent software agents for information sharing with confidentiality and trust. It clearly defines an Intelligent Software Agent, background of Information sharing in intelligent agents and the trust in the agents. Some part of the information needs confidentiality. The information that is shared requires security policy enforced based on the level of confidentiality and trust level of individual agent.

1. Introduction

Exciting emerging applications require hundreds or thousands of agents and robots to coordinate to achieve their joint goals. In domains such as military operations, space or disaster response, coordination among large numbers of agents promises to revolutionize the effectiveness of our ability to achieve complex goals. Such domains are characterized by widely distributed entities with limited communication channels among them and no agent having a complete view of the environment. Information relevant to team goals will become available to team members in a spontaneous, unpredictable and, most importantly, distributed way. The question addressed in this paper is when a team member senses some information, how it can decide which team member to

communicate that information to. In most applications for very large teams, broadcasting information is not suitable, desirable or feasible. Instead, the agent must attempt to target its information delivery to just the agents that need it. In a large team, each member has a limited model of what other members of the group know or even what many of them are doing. For example, a field agents involved in a military operation may observe many features of a battlefield on route to an assignment. Many of its observations will be relevant to the plans of other combatants but the field agents will not necessarily know which group members require the information.

This paper presents a system to sharing information that is applicable to large teams [1]. A key to the solution is imposing a static network topology on the members of the team where each agent requiring communication to be only along very few links in that network. The key observation underlying this solution is that each piece of information is interrelated and the sender of a piece of information can “guess" who might need some information based on previously sent messages. Thus, when an agent has a piece of information, it can determine which of its neighbors in the network is most likely to either need the information or know who does, based on related messages previously received. Secondly, investigate the influence of different types of team network topology on the efficiency of information sharing.

Trust negotiation is a very important part of any system or an organization. Without trust no transaction can be successful. If there are many systems interacting between them each one has to have trust with other in order to share data, alliances and deals to save the operation cost which is major part of any project. The negotiation is always conflicting since it is to compromise between two agents in order to achieve decision for conflicting distributed systems. The negotiation is taken based on the environment with two decisions to support self interest or the entire system. The decision tree is then formed based on the negotiation and the scenario is stored into the library incase if it is newly proposed. So that it can be used in the future without much of computation.

The Confidentiality of Information is a major threat in a system that is used to share information. Incase the confidential information is disclosed to an agent that is not entitled to that level of security, there is a chance of losing the vital information to an untrustworthy agent. If the trust level of the agent does not match with the security level of the information then the information is secured. The security policy of the information is distinguished into four types as unclassified, classified, secret and top secret.

2. Preliminaries

2.1 Definitions:

Agent: An agent defines a person or an organization that interacts with other person or organization on behalf of the owner.

Software Agent: It is not as simple as a real world agent. There are various definitions for a software agent. The closest definition would be the following "A software agent is a software with some inbuilt functionalities that interacts with other software agents and perform the allocated task based on the rules that govern them."

Intelligent Software Agent: It is a hybrid version of a software agent with some intelligence of its own. "[An Intelligent Software agent is] a piece of software that performs a given task using information gleaned from its environment to act in a suitable manner so as to complete the task successfully. The software should be able to adapt itself based on changes occurring in its environment, so that a change in circumstances will still yield the intended result." (Herman's 1997)

2.2 Functions:

Intelligent software agents should perform the following tasks continuously

1. Insight of changing environment
2. Action required for the change
3. Reason to the action taken
4. Solution for the problem
5. Draw Inferences and perform decision tree for future use.

3. Background and related work

Information sharing and Trust negotiation in intelligent agents have there root way behind from 90's. There are various researches going on Information sharing in Intelligent Software Agents lab of Carnegie Mellon (the Robotic Institute). One of such is Information sharing in Agents. They have alternative decision making systems and Bilateral Negotiations with outside options. In this paper for knowing the background of trust negotiation, will discuss some of the points from the bilateral negotiation with outside options.

The bilateral negotiations paper considers each trust negotiation as a thread. The model is composed of three modules: single-threaded negotiations synchronized multi-threaded negotiations, and dynamic multi-threaded negotiations. The single-threaded negotiation model

provides negotiation strategies without specifically considering outside options. The model of synchronized multi-threaded negotiations builds on the single-threaded negotiation model and considers the presence of concurrently existing outside options. The model of dynamic multi-threaded negotiations expands the synchronized multithreaded model by considering the uncertain outside options that may come dynamically in the future.

Most related work can be classified into one of several major categories. The first strand of research is based on a centralized model or distributed model where there are agents such as match maker, information broker or message broad who can response for all information communication [2,3]. These works has been shown to be able to greatly improve multiagent [4] system performance [5]. However, such work is inadequate for large team, since it is impossible or undesirable for all team members to share all their information all the time, i.e. because of the limit of required communication channels. The second major strand of research is relies on agents maintaining a shared model of each other or knowing exactly other members' actual internal state as STEAM[6], COM-MTDP [7] and CAST [8]'s mental model. However, as for centralized approaches, in large team there is insufficient bandwidth to support such an approach.

The information sharing problem can also be handled by setting up decentralized model. Both [9] and [10] did a communication decision model based on Markov decision processes (MDP). Their basic idea is an explicit communication action will incur a cost and they supposed the global reward function of the agent team and the communication cost and reward are known. Moreover, [11] put forward a decentralized collaborative multiagent communication model and mechanism design based on MDP which assumed that agents are full-synchronized when they start operating, but no specific optimal algorithm was presented. Unfortunately, there are no experimental results showing that their algorithm can work on large teams. Incomplete information theory is another way to solve the information sharing problems. [12] Presents a framework for team coordination under incomplete information based on the incomplete information game theory that agents can learn and share their estimates with each other. [13] Used a probability method to coordinate agent team without explicit communication by observing teammates' action and coordinating their activities via individual and group plan inference. Research on social networks began in physics [14, 15, 16], but since it has been applied in many areas though rarely in multiagent work.

4. System Architecture

The system model for information sharing among large teams can perform distributed information sharing without the cost of maintaining accurate models of all the teammates. First, impose a network topology on the team members analogous to the social networks that exist in

human societies. The key characteristic of this network model is that information exchange is based on peer to peer communication. Specifically limit agents to communicating directly with only a small percentage of the overall team.

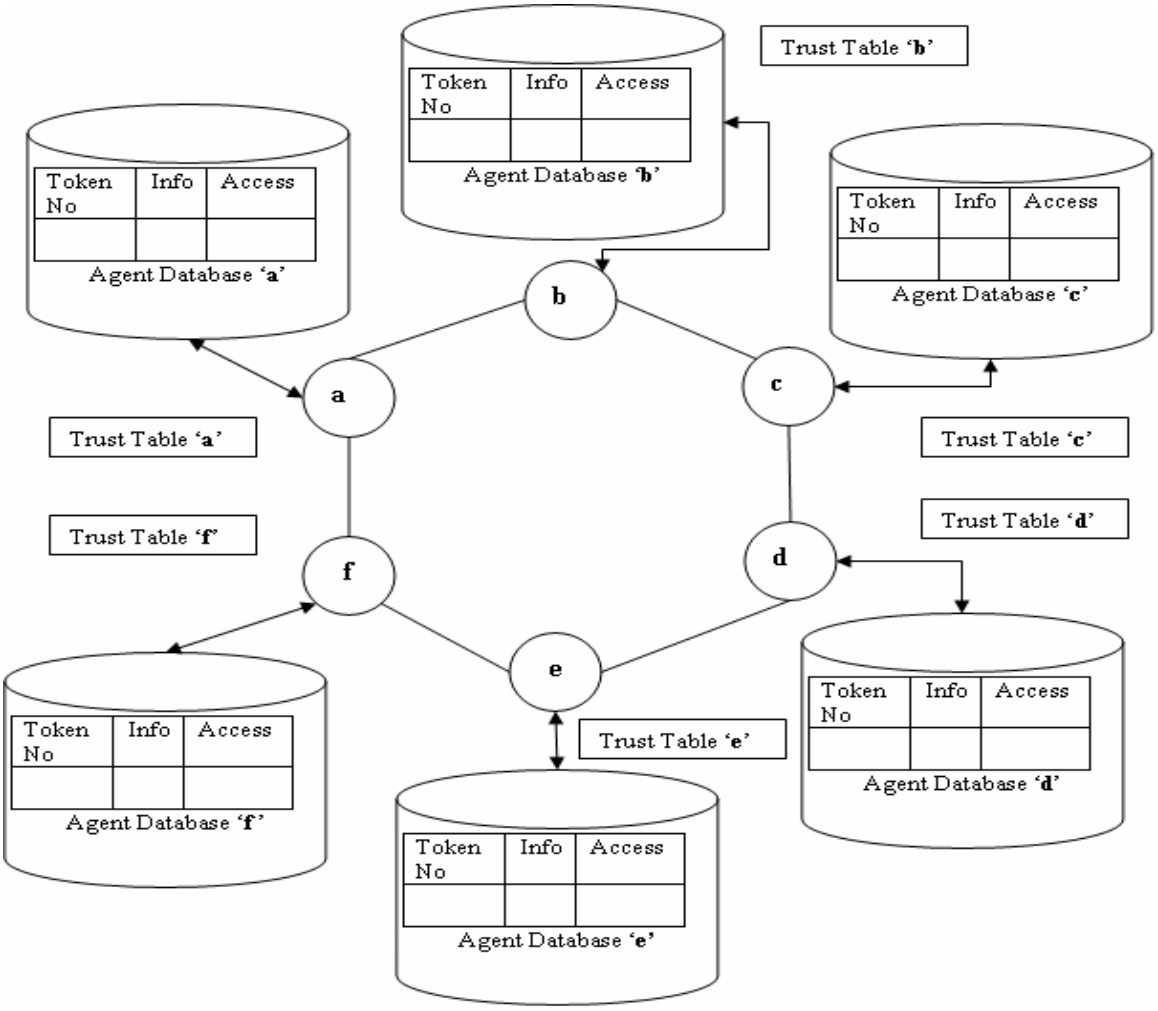


Figure1: System Model for agent information sharing with confidentiality and trust negotiation

Leveraging the team network, our basic approach like Figure1 is when an agent has a piece of information to communicate, it forwards that information to the direct acquaintance most likely to actually need that information or know who will. Then the acquaintance performs the same reasoning when it gets the information. After passing through hopefully, a small number of team members, information arrive at a team member that needs it. The intuition is that each agent attempts to guess which of its acquaintance either require the information or are in the best position to get the information to the agent that requires it. Even though members of large teams will not have accurate, up-to-date models of the team, our hypothesis is that the models will be accurate enough to deliver the information in a small number of "hops".

One agent is randomly chosen as the source of some information and another is randomly chosen as the sink for that information. A probability is attached to each link, indicating the chance that passing information down that link will get the information through the smallest number of links to the sink. The probability will increase as it reaches closer to the target. The chance of missing the target depends on the distance between the source and the sink. The number of "hops" to vary as the distance varies.

The challenge is to construct complex models for information sharing but only have reasonable models to improve agent's guessing. The key question is how to create models that allow the agent to "guess" correctly more often than not. To achieve this, we observe that each piece of domain knowledge is typically related to each other piece of domain information. For example, if agent '*a*' tells agent *b* about a plan to destroy an enemy base, when agent *b* gets the information that the base is fake, sending that information to agent *a* is a reasonable thing to do, since *a* likely either needs the information or knows who does. So it is reasonable to infer from an agent's formerly sent message that it may need the other kind of information to improve the performance as the above example. Thus, the previously received information can be interpreted as evidence to infer which acquaintance to send other information to. If an agent maintains a knowledge base about what it heard from its acquaintances, it can use that knowledge to determine where to route newly received information. The other challenge in the network is trust management. Consider the previous example. In case if agent *a* is not trustworthy then that information to destroy the enemy base might be fake. So trust negotiation is an important goal. In our system we can negotiate trust based on their acquaintance. For instance the source is acquainted with another agent in the network that is acquainted with the sink; the sink can get the trust level from its acquaintance. In the beginning the complex network will be formed with no acquaintances. Then once the connection is setup and each agent begin to acknowledge each other's neighbors then the trust levels are assigned to the agent based on their information. If there is a bad agent then it tends to spoil the entire system. The other agent sends the bad acquaintance that they have had with the corresponding agent.

In the system simulation there is also a security policy implementation that has a very important part in the sharing of the information to authorized agents rather than transferring the secret level data to lower access agents. The token and the information are linked with a security level. Each agent maintains its own level of confidentiality for any particular information. There may be instances where the same information with different clearance level can be stored in different agents. This also makes a possibility that if one agent rejects the request based on the trust level of the requesting agent, another agent can service the request based on the trust level or

acquaintance level that it has maintained with for that corresponding agent. The following example can explain the point. Agent '*a*' can have two or more acquaintances in this case it is two '*b*' and '*c*'. The trust level of '*a*' with '*b*' is in higher clearance level say secret level and with '*c*' it is in classified level. If there is a request from '*a*' sent for some information at secret level then '*c*' will reject the request and '*b*' will service the request. Similarly '*b*' and '*c*' have two different levels for the same information *i.e.* information '*x*' level secret in '*c*' and level classified in '*b*'. If '*a*' request for the same information then there is a chance that '*b*' will service the request.

5. Implementation of the system

5.1 Overview

The simulation of the intelligent agents sharing information is done using Java programming. The program mainly concentrates on two things. How much message is being transferred from each agent and the trust element within each agent? The summary of the simulation mainly has results on how much message each agent had in the beginning of the session? How much they shared with the other agents in the simulation and how much they received from the simulation.

The important feature of the simulation is that it also holds the history of the summary which makes easy to know the amount of data lost in each session. The agents can make use of the history of the summary to learn more about the other agents in the simulation and try to avoid the more data loss in the future session with the same set of agents. This also helps in knowing the nature of the agents involved, if they are ready to participate and send more messages or they are just waiting to get the most out of the other agents. Such agents are also blocked from the simulation by not sending messages to that particular link. This depends on the individual discretion of the agents. They also pass on the information to other agents in the simulation that such a neighbor is not willing to send any information and readily accepts all the information that is passed on to it or through it. Those dormant agents are like leeches that spoil the entire network. The algorithm is explained clearly in the next section of Implementation.

5.2 Algorithm for Information sharing with Confidentiality and Trustworthy Computing:

In this algorithm as in Figure 2, at the time of forming the coalition, the agents have the information about the direct acquaintances i.e. a neighbor and their trust level. If there is going to be a new neighbor the trust level is set to a minimum acquaintance level. Then each agent has its own set of information to be shared with other agents in the network. The information is linked with a significant token number and a security Policy. The moment a message is requested by some agent for some information, the token is received then the security policy of the

corresponding information is matched with the clearance level of the requesting agent. The clearance level mainly depends on the trust level of the requesting agent that is linked with the source agent. In this algorithm there are four such clearance levels. U→Unclassified, C→Classified, S→Secret and T→Top Secret. The trust levels are similarly split into four levels where in the minimum threshold is set for unclassified information. Each agent can read below or at their level.

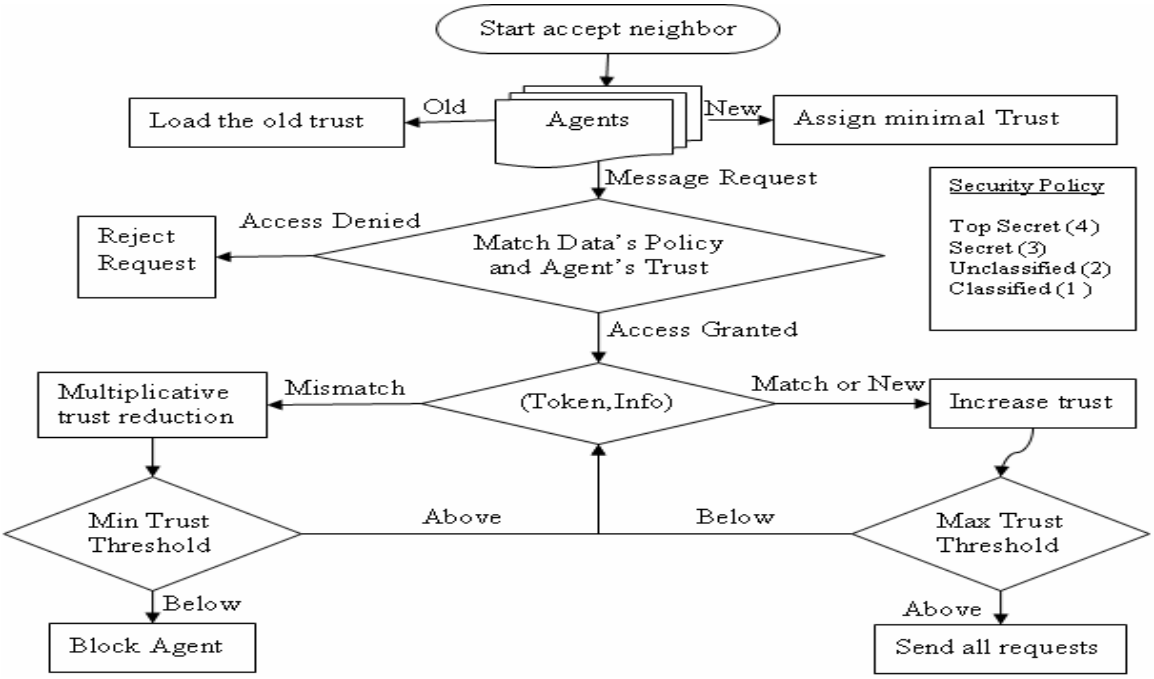


Figure 2: Flow diagram of the Information sharing with confidentiality and trust management.

There may be multiple copies of the information existing simultaneously in the network along with the same token number, yet the token and information pair is always unique. If the agent gets the same information with two different tokens or vice versa, then his discrepancy will lead to loss of trust. It will perform a multiplicative decrease in the trust level. Similarly if new information arrives trust level of the acquaintance is increased. There is a minimum and maximum threshold level for trust. If any acquaintance falls below the minimum threshold of the trust, then they are removed from the circle of trust, further communication is stopped and the rest of the acquaintances are notified about the bad agent. If the acquaintance's trust level goes above the maximum threshold then the agent sends all the messages requested by the acquaintance. The information sharing goes on until one agent gets the entire information it needs or to a fixed number of time where all the agents have the list of data lost and data gained. It also stores the

history of the direct acquaintance and its trust level which helps in future coalition with the same agent.

5.3 Specification of the algorithm

1. Form Communication link with other agents where the neighbors are the acquaintances.

2. If new neighbor set minimal trust level else load the existing trust level from the database.

3. If an agent request for some information. Check the trust level of the agent and the access or security level linked with the information

4. If the access is granted allow service the request based on priority. Else reject request.

5. Start sending and receiving messages (the tokens and the Information are linked).

6. If there is mismatch in messages multiplicative decrease of trust and if the trust goes below minimal trust after decreasing block agent and notify the network

7. If there is message (new or old with match) additive increase trust and also if the trust is above max threshold send the entire request one by one.

8. If any one agent has all information or end of session occurs end link store trust level, Message (Token and Information).

9. Calculate the amount of data lost or gained from each acquaintance

6. Experimental results

The simulation of the algorithm was implemented and there were many sets of results generated. The experimental results were very much helpful in understanding how the system works. In chart 1 the Information that was sent from each agent and the information gathered at each end is collected and the Net Gain is also calculated.

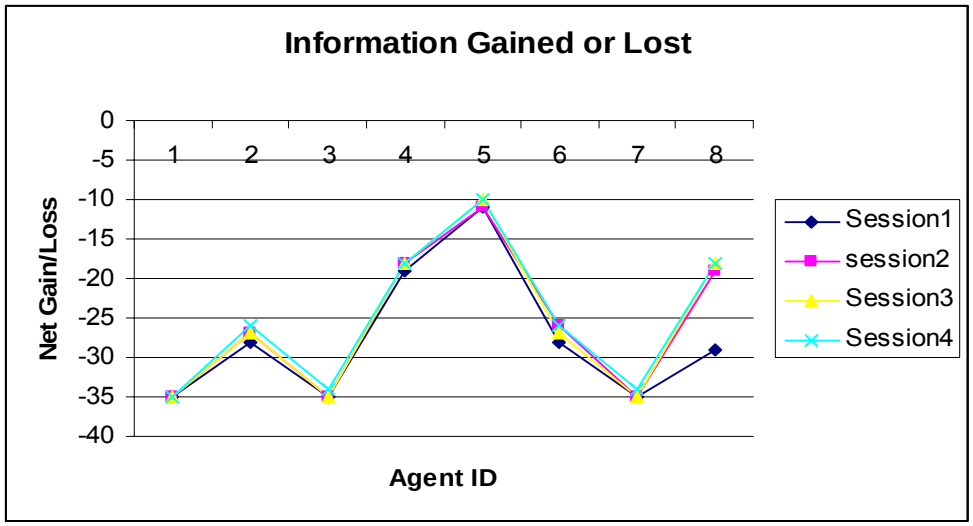


Chart 1: X axis → agents and Y axis → Net Gain/Loss

Let *T → Net Gain/Loss of Information for any agent.*

R →The message received from Agents by some agent ai.

S → The message sent to other Agents (a0, a1…… an) by agent ai.

O → The own message of each agent in the beginning of the session.

T = (R-(S+O))

Chart 1: The above chart mainly describes about the net gain of information by each agent in four continuous sessions of information sharing with trust computing.

The Chart 1 has four set of simulations that was done within 8 agents. The simulation revealed that as the session increases the gain also increases. This is because the agents come to know well about the other agents. Agents have the trust level of each agent in their database summary. The trust level increases the gain increases. Since most of the agents send 90% non negative messages the gain increases for each session. The chart1 also clearly shows that there is not a much of difference in gain of all the agents. They all share same level of trust in the beginning and the gain varies based on their trust level through the simulation. If they send one negative message their gain goes down. The neighbors stop sending messages if they are notified that some agent is below the threshold level of some other acquaintance. So the gain in sharing depends mainly on the trust level. The chart has net gain and loss on its Y axis. The series one to four indicate the simulation that was conducted on the eight agents in continuous session of information sharing. The chart clearly indicates the increase in gain as the session progresses. The summary of the experimental results contain the amount of message sent, received and the Net. It also has the recent trust level of all the neighbors. The newly received tokens are also copied in the summary.

7. Summary and Future Directions

The proposed algorithm has been implemented. The experimental results show that the information sharing is done as in peer to peer communication network. The amount of information lost and gained is stored at the end in the database. The number of messages sent to share a little amount of information through the network is high. The scalability also becomes an issue. If there are more neighbors the amount of message sent and managing the traffic of messages becomes a very big issue. The future work on this research can be implementation of the above system in which the guess and hops are calculated to the efficient way to share information among the agents.

A major issue we leave for future research is how to calculate the relationships between pieces of information which is highly relative with domain knowledge and expertise where our algorithm

should be applied. Furthermore, we do not investigate how information sharing works on negative relative messages where the relationship between pieces of information. Does the dormant agent gain more than the other active agents? Can the agents form a multicasting group which might help in communicating with a group of agents simultaneously? The multicasting group will save a lot of network resources by sending one message to a gateway agent and thereby pass it to the whole multicast group.

References

[1] P. Scerri, Y. Xu, E. Liao, J. Lai, M. Lewis, K. Sycara. Coordinating very large groups of wide area search munitions, Recent Developments in Cooperative Control and Optimization, Dordrecht, NL: Kluwer Academic Publishers.

[2] M. H. Burstein and D. E. Diller. A framework for dynamic informationflow in mixed-initiative human/agent organizations. Applied Intelligence on Agents and Process Management, 2004. Forthcoming.

[3] K. Decker, K. Sycara, A. Pannu and M. Williamson.Designing behaviors for information agents. Procs. Of the First International Conference on Autonomous Agents, Feb., 1997.

[4] P. R. Cohen, H. J. Levesque and I. Smith. On team formation.In J. Hintikka and R. Tuomela, editors, Contemporary Action Theory, Synthese,1998

[5] K. C. Jim and C.L. Giles. How communication can improve the performance of multi-agent systems. In Proceedings of Autonomous agents'01, 584-591, 2001.

[6] P. Scerri, Y. Xu, E. Liao, J. Lai, K. Sycara. Scaling Teamwork to Very Large Teams, AAMAS 04, Forthcoming, 2004.

[7] D. Pynadath and M. Tambe. The communicative multiagent team decision problem: analyzing teamwork theories and models. Journal of Artificial Intelligence Research, Vol.16, pages 389-423, 2002.

[8] J. Yen, J. Yin, T. R. Ioerger, M. S. Miller, D. Xu and R. A. Volz. Cast: Collaborative agents for simulating teamwork. In Proceedings of IJCAI'01, pages 1135-1142, 2001.

[9] P. Xuan, V. Lesser and S. Zilberstein. Communication decisions in multiagent cooperation: Model and experiments. In Proceedings of Autonomous Agents'01, 2001. [10] C.V. Goldman and S. Zilberstein. Optimizing information exchange in cooperative multi-agent systems. Proceedings of the Second International Conference on Autonomous Agents and Multi-agent Systems, 2003.

[11] C.V. Goldman and S. Zilberstein. Mechanism design for communication in cooperative systems. Game Theoretic and Decision Theoretic Agents Workshop at AAMAS' 03, July, 2003.

[12] H.H. Bui, S. Venkatesh and D. Kieronska. A framework for coordination and learning among team members. In Proceedings of the Third Australian Workshop on Distributed AI (DAI-97), pages 116-126, Perth, Australia.

[13] M.V. Wie. A probabilistic method for team plan formation without communication. Proceedings of the Fourth International Conference on Autonomous Agents, pages 112-113, Barcelona, Spain, June 3-7, 2000.

[14] R. Albert and A. Barabasi. Statistical mechanics of complex networks. Review Modern Physics, 74, 47,2002.

[15] M. E. J. Newman. The structure and function of complex networks. SIAM Review, Vol. 45, No. 2, pages 167-256, 2003.

[16] D. Watts and S. Strogatz. Collective dynamics of small world networks. Nature, 393:440-442, 1998.

Report #4

Research and simulation of game theoretical techniques for data sharing among semi-trustworthy partners

Ryan Layfield, Murat Kantarcioglu, and Bhavani Thuraisingham
Department of Computer Science, The University of Texas at Dallas
{layfield, muratk, bxt043000@utdallas.edu}

Published as Technical Report UTD-CS-46-06

Abstract

Our objective within our work has been to consider the interaction of participants within a loose coalition. In particular, we are interested in a scenario in which those involved have made a reluctant but necessary decision to trade information to achieve some goal. A great deal of work has already been done in the areas of secret sharing and protocol enforcement. However, even if agreements to exchange are kept, there is no guarantee what is shared is legitimate. The ultimate goal of this research is to create a behavior which works optimally against lying agencies while taking advantage of implicit trust. Our results at this point in the research suggest our algorithm is effective against basic opponents, though more refinement is needed. We discuss which behaviors worked for the players and why, with regards to the motivating factors for each strategy. Essentially we are using game theory to enforce honesty within a competitive coalition.

1. Introduction

A coalition consists of a set of organizations, which may be agencies, universities and corporations that work together in a peer-to-peer environment to solve problems such as intelligence and military operations as well as healthcare operations. Figure 1 illustrates an example coalition where three agencies have to share data and information. Members of a coalition, which are also called its partners, may be trustworthy, partially trustworthy or untrustworthy. Furthermore, coalitions may be dynamic in nature. That is, members may join and leave the coalitions in accordance with the policies and procedures.

Data from the various data sources at multiple security levels as well as from different services and agencies including the Air Force, Navy, Army, Local, State and Federal agencies belonging to coalitions have to be integrated so that the data can be mined, patterns and information extracted, relationships identified, and decisions made. The databases would include for example, military databases that contain information about military strategies, intelligence databases that contain information about potential terrorists and their patterns of attack, and medical databases that contain information about infectious diseases and stock piles. The agencies have to share information as much data as possible but at the same time maintain the security and integrity requirements. This concept has come to be called Assured Information Sharing. .

We have been examining policies and procedures for assured information sharing in a coalition where partners may be trustworthy, untrustworthy or partially trustworthy. In the case of trustworthy information sharing we conducted experiments to determine the amount of information lots by enforcing policies. In the case of untrustworthy information sharing we have examined approaches for both defensive and offensive operations. In the case of partially

trustworthy data sharing, we have examined game theoretic approaches so that the members of a coalition can extract information for their partners.

This paper describes our research on information sharing between partially trustworthy partners. It essentially considers interactions between participants within a loose coalition. In particular we apply game theoretic approaches for allowing players to determine when and who may lie to them. The organization of this paper is as follows. Section 2 gives motivation for our research. Related work is discussed in section 3.

This paper is broken up into 9 sections. Sections 1 and 2 setup the nature of the research and the real world motivation we have for it. Section 3 outlines related areas of work in game theory. Section 4 outlines the constraints placed by the assumptions we've made during the research, detailing the reasoning behind each choice. Section 5 outlines the mathematical and game theoretical principles within our model, while section 6 details how we plan to use these factors in our experiments. Section 7 gives the highlights of our results, providing as much data as time and space will allow. We conclude with section 8, showing our observations from experimentation, and section 9 cites our sources.

2. Background

Our motivating example can be found in the nature of international coalitions. Consider a set of countries which are interested in their own national security. At some point in the future, an attack will be carried out by a militant or criminal organization which will have far-reaching impact on the intended victim. The exact details of the situation are unknown, but each of these nations suspects that they may be the target.

Each nation has an agency which is responsible for dealing with such affairs and has the capability of stopping the attack once enough detail is known. At the initial discovery of the event, each agency has utilized their available resources and connections in an attempt to ascertain where and how the attack will take place. Regrettably, no single agency has been able to gather more than a fraction of the complete intelligence necessary, and their results have turned up no further leads on which to act.

As a specific example, consider recent events. This particular attack will be carried out by some unspecified group using what the agencies suspect is some kind of high-yield destructive device. The nature of the threat requires that each agency consider the lives of countless civilians, a considerable motivation.

The looming time table for the attack, a lack of further available data, and high stakes suggest that any single agency cannot complete the information alone. Understandably, all agencies would like the option of completing their investigation of the attack on their own. Involving an outside party raises the possibility of compromising national security.

In the face of such adversity, a number of agencies believing they are the target have decided to enter into a coalition. The framework for such an arrangement could be in place based on alliances, membership in worldwide organizations, or simply through political negotiation. Regardless of how it is formed, such an alliance provides the framework on which our research is based.

Due to political distrust among the countries involved, sharing information openly has no guarantee of gaining more data about the event. Therefore, cooperation in this situation is best achieved in a trade-based system. Consider the total data collection as a series of fragments. Assuming a protocol is agreed upon for making trades secure and no agency has incentive to deviate, it becomes possible for each agency to exchange pieces until enough information has been gathered to ascertain the nature of the attack. In ideal circumstances, all agencies would be inclined to trade data feely until the puzzle was complete and all could share the results.

However, the reality of politics dictates that it would benefit an agency to be the first to complete the puzzle. Doing so could potentially earn more funding for the agency from their

home country as praise for their work, or it could simply be a matter of beating others to it for leverage. Whatever the case may be, although protocols may be followed, it is within each agency's own interest to complete the puzzle first and ensure their opponents did not make the same progress.

The protocol in place only dictates how information is traded. It does not make any guarantee as to the legitimacy of what is being traded. As such, an agency can choose to fabricate information and offer it as if it were legitimate. Fabricating information is a relatively simple task; as long as it does not appear to be inconsistent with related information, an agency which chooses to exchange this false data will gain more from the trade than their opponent. This essentially permits an agency which wants to ensure certain opponents do not succeed to perform a 'legal' form of sabotage.

Since there is no guarantee that traded information is legitimate, an agency has three options following the trade. First, it can choose to accept the information as completely trustworthy, integrating it into their own collection immediately upon receipt. This is the easiest method available, and it is a viable option as long as their trade partner has proven trustworthy in the past.

For partners that are less trustworthy, the next option is to verify by asking another player to compare it with its own data. The result of such a query simply determines whether data matches the other participating agent's own holdings. The value of such a query is big, as it requires little work on either agency's part and it can yield relatively inexpensive results as long as participants are honest. The complete value of such a choice is based entirely on the holdings of the queried partner; there remains a risk that further analysis will be necessary.

The final option is internally verifying the data. By using the agency's own internal resources, it is possible to determine with complete accuracy whether or not the data is a fake. This is obviously a costly choice, as it requires a substantial devotion of time and manpower, but it does offer the guarantee that no further analysis will be required. It also has the benefit of not leaking the status of the agency to their competitors.

In simulations, it is easy to regard the situation as if all participants viewed each other equally. This greatly simplifies models and provides level footing for measuring interaction. Nevertheless, to adequately model the international environment, one must consider the history of the countries which each agency represents.

The relationship between any two countries is determined by what is frequently a complicated history of cooperation, distrust, victory, and defeat within a number of arenas. This relationship may be misinterpreted by both sides; country A may view country B as a long-time ally while recent discoveries by country B lead it to regard country A as a bitter enemy. Perception depends also on the context of the situation and the private motivations each country has to discover the information.

The end result of this scenario is the clear need for a model that can adequately account for each agency's view of its opponents and maximize the use of available information to make cost-effective decisions to ultimately complete the information puzzle. While this may seem as an ambitious task, a proper blend of game theory modeling and reasonable assumptions can yield a simulation in which we can model the complex interactions among them. Ultimately, we will be able to draw conclusions on how strategies can be changed and why certain agencies 'win' in these scenarios.

3. Related Work

Game theory has long been a staple of analysis within social and political sciences. Repeated games themselves represent a persistent and realistic view of how people interact. Robert Axelrod in his book "The Evolution of Cooperation" explores the findings of a contest he created using a variation of the Prisoner's Dilemma. In it, all participants were encouraged to

submit their own algorithms in an attempt to find the superior approach. Surprisingly, the winner of the contest was a retaliatory algorithm known as Tit-for-Tat, which essentially cooperated with another player unless there was a change of strategy, in which it simply mimicked its opponent. [1]

Several related areas of work have already considered the possibilities of game theory as applied to information sharing. In particular, a great deal of work has been done on peer-to-peer networks. Within these file sharing systems, independent players join and leave at their leisure, seeking to download a file or files with the help of other participants. Problems arise when a new participant joins the network and download a resource from other peers and never actually contribute to the group. This process, known as leeching, has been a large problem in piece-meal file sharing protocols such as the popular BitTorrent. The work of Gupta et. al [4] and Buragohain et. al. [2] both deal with this behavior by creating a system of incentives for further contribution.

4. Assumptions

4.1 Initial Assumptions

We make the assertion that all agencies begin with the same amount of 'pieces' of data. It is assumed that each piece has an equal value, thus ensuring every simulated agency begins with a fair amount of available pieces to trade. This is somewhat unrealistic, as many countries are reputedly much more effective at gathering information than others. However, a level playing field is essential to understanding how strategies are chosen, and an unequal distribution would favor the agent with more data than the rest.

We also assume that information never changes in its value as the game progresses. This is to simplify the model, as current research suggests modeling an economy without a centralized agency is currently far more difficult and has yet to be adequately explored. It is important to note that, in real scenarios, information availability makes some data much more valuable than others (i.e. the names of the attackers vs. whom they represent).

The next major assumption made is that each country has some predetermined amount of bias toward other countries. This is varied between sets of experiments, and it is obvious that some countries will have higher collective favor due to higher ratings than others. Every agency has some opinion of the other players, but none of these players have a way of knowing what that opinion is. Such data can only be determined through observation of interactions.

The cost associated with all data acceptance methods is consistent among all agencies. That is, the alternatives to simply accepting a piece each are assumed to have the same impact on local resources for each agency. Again, in real life, each agency will have some level of efficiency for multiple approaches, but for purposes of focusing on strategy, this cost will be consistent. Note this also implies each agency essentially has the same amount of strength regardless of the country they represent.

4.2 Interaction

Communication is an essential part of trade-based interactions. To reflect the modern technologies of privacy and encryption techniques, we assume that there exists a secure bidirectional information pipe between every agency over which messages can be sent. The pipe is reliable, always connected, and completely secure from any sort of outside interference or observation. This includes determining the type, content, and nature of messages, as well as when messages are being passed. In essence, no participant in the game can use information gathered from surveying or observing their opponents' interactions.

Given that this pipe is between all possible pairs of agencies, and that the pipe is always present, we also assume that no agency will be isolated from potential interaction. An agency must consider all other participants with the same criteria. Alliances between agencies to 'shut-

out' an opponent are not permitted in any form or fashion. This does not prohibit the sharing of information regarding reliability.

Another level of abstraction within this form of interaction is that no agency can directly observe any other agency's activities. While communication itself is protected, this stipulation includes learning what another agency is planning to do next, what information it currently has, and how close it is to gathering all of it. Data of this nature can still be derived by determining how they want to trade and cooperating with other participants to discover this information. How this is done will be discussed later.

When using the external verification method, we assume that all agents are completely honest in this protocol. The response from such a query cannot be fabricated. Additionally, any information shared in a query will always be with another agent whom through unspecified means has confirmed they indeed also have a piece. Regardless of what is shared, no agent may derive assumptions about other agents within this protocol, including the presence or absence of pieces within other agencies.

The issue of deviation from protocol is an important consideration that has already been briefly mentioned. This becomes a significant assumption in keeping the cooperation among agents viable, as the real life equivalent of backing out of a deal can both drastically upset the political environment and eliminate further legitimacy of established protocols. In some instances, it may even destroy the coalition itself. Exchanging false data is actually less risky by comparison, as there remains the possibility that the transmitter of the information simply picked it up from a previous trade believing it was legitimate.

One of the biggest issues in creating a workable simulation is avoiding the complications of an asynchronous environment. As indicated in the work of [9], while treating the agencies as a series of independent systems is a much more realistic interaction model, it opens up a new dimension of possible variables including the speed at which each agency operates at, how conflicting perceptions of time are handled, and issues of fairness. The accommodation of just a few of these factors would require the implementation of multiple potentially costly protocols within the model, all of which are beyond the concern of this research. Therefore, we assume that interactions happen in rounds, where each agency is interacting and trading simultaneously. There is a limit of one initiation of trade per agency per round, though one agency can be a part of multiple trades initiated with them.

4.3 Winning

We assume that once an agent believes it has acquired all pieces of information necessary to act that there exists some external action which can determine its success. Clearly, in reality, this would require an extraordinary commitment of resources to begin acting on the information to insure the event did not transpire. We therefore assume that when an agency decides to act that it can only do so once.

The resources available to each agency are assumed to be sufficient for the duration of the game regardless of strategy. They are consumed based on the verification strategy chosen at each round, and represent the investment of time and resources to accommodate the choice. However, in the interest of being efficient and recognizing the resources are indeed limited in real-life scenarios, each agency is motivated to minimize overall use.

The nature of 'winning' requires the number of pieces held by an agent to exceed some threshold of correctness. This threshold represents the allowance for error which occurs during any investigation, under the assumption that a fraction of errors are correctable through correlated data and associated information. Such a requirement also avoids situations in which two agencies, suspecting the other is on the verge of winning, refuses to tell the truth on a piece exchange necessary for either to win. [5]

The most important aspect of trying to win this game is that trust generated among agents will be carried into future relations countries which the agencies represent. For example should agency A choose to lie frequently to agency B in order to win the game, whether or not A succeeds, future relations between A and B will be severely degraded. In the event that A needs help from B, B will be much less likely to deal honestly in return. We thus assume that agency wants to also avoid losing considerable amounts of their reputation.

5. Modeling

The foundations for our proposed model begin with basic game theory. According to [7], game theory is best described as the study of mathematical models of conflict and cooperation between intelligent rational decision-makers. Essentially, we consider each agency as a player within a game whose goal is to gather all information that was initially distributed.

The lack of certainty regarding an opponent's actions and motivations rules out the use of traditional strategic modeling. Instead, we make use of Bayesian Modeling. According to the formal definition from [9], we consider a game as set of players N and a finite set of possible states Ω that any given player can be in. A state here is simply how the player views their opponents and what pieces have been collected up to this point.

In addition, there are a number of variables in this model that warrant player-specific information tracking. There is a set of actions A_i that each player can take with regards to their current state. Since they do not have complete information, they rely on a set of signals T_i that are observed with regards to the other participants, which is transformed by a signal function τ_i: $\Omega \rightarrow T_i$. This permits each player to interpret the data they perceive independently and draw from previous internal history. For each state we believe our opponents may have been in out of Ω, we assign a probability p_i that dictates our prior belief of other players' choices as indicated by a signal. Each interpreted signal is guaranteed to always have some probability, according to $p_i(\tau_i^{-1}(t_i)) > 0$ for all $t_i \in T_i$. This is implicitly a part of how we calculate previous strategies. Regardless of how we view it, our theory must ultimately be rooted in Bayesian games. [6]

This scenario is in essence a repeated game. That is, our signals in T_i are a function of a history of actions H, which is defined as $H = \bigcup_{i=0}^{n} A_i$, where A_0 represents the initial history [9]. Over the course of the game, the history of the game is built by choices made by each participant as they interact with others. Previous interactions are private and viewed and evaluated by each agency. The implicit signal function yields the probability of fake pieces being handed out given prior analysis.

As part of the game, we consider the objective the specific acquisition of pieces of information to complete the original 'puzzle'. How the puzzle itself is generated is irrelevant; we simply need a series of uniquely identifiable pieces. The operations on these pieces include making a real copy and creating a fake copy.

In strategic form, the basic representation of strategies in our game between two trade partners that can choose to lie to each other is depicted in Table 1.

P_2 / P_1	Tell Truth	Lie
Tell Truth	A / A	B / $A - L$
Lie	$A - L$ / B	$B - L$ / $B - L$

Table 1. Basic strategic representation.

where *A* represents payoff if the other player tells the truth, *B* represents potential gain if the player lies, and L represents the loss experienced by either player if they are lied to. This initial configuration is derived from the work of [7], and it gives a good starting point in terms of how the system could be modeled at it's most basic level. However, this is insufficient for our particular scenario.

There are essentially two types of players for each individual game between any two players: those that lie, and those that do not. Thus, {*truth,lie*} is the possible universe of strategies Ω. Which strategy an agency chooses varies from trade to trade, which implies a mixed strategy approach that we must accommodate for. For simplicity in our equations, $p^i_j(\ldots)$ represents the perception i has of j regarding some probability of an action or attribute. The result of $p^i_j(fake)$ is the perceived probability that player j will be lying to player *i*. Equation (1) represents this modified utility return for each possible choice of strategy for player 1. The probability $p^i_j(verify)$ represents the chance that player *j* will choose to verify the piece from player *i*, and M represents the loss of favor player *i* will incur in player *j*'s perception.

$$\begin{aligned} u(Truth_i, Truth_j) &= A \\ u(Truth_i, Lie_j) &= A - L(1 - p^i_j(\text{fake})) \\ u(Lie_i, Truth_j) &= B - M(p^j_i(\text{verify})) \\ u(Lie_i, Lie_j) &= B - M(p^j_i(\text{verify})) - L(1 - p^i_j(\text{fake})) \end{aligned} \tag{1}$$

This equation brings us closer to accommodating both agencies' motivations with regards to risk and consequence. However, it does not accommodate the fact that pieces must be chosen based on when and how we verify.

P_j / P_i	Tell Truth	Lie
Tell Truth	A A	$B - M(p^j_i(\text{verify}))$ $A - L(1 - p^i_j(\text{fake}))$
Lie	$A - L(1 - p^i_j(\text{fake}))$ $B - M(p^j_i(\text{verify}))$	$B - M(p^j_i(\text{verify})) - L(1 - p^i_j(\text{fake}))$ $B - M(p^j_i(\text{verify})) - L(1 - p^i_j(\text{fake}))$

Table 2. Refined Strategy with additive Opinion Weighting

Consider the value of *V* as a numerical representation of resource costs for verification. As stated before, there are three possible actions for receiving a piece: {*accept, external_verify, internal_verify*}. We assign a cost determined by function *c(…)* to each such that $0 \le c($accept$) < c($external_verify$) < c($internal_verify$)$. The choice of each based on cost alone would yield an equilibrium of always accepting, but based on choices by other players, this strategy would quickly compromise the ability to recognize players which choose to lie. The next logical step would then be to externally verify each piece and use this as an equilibrium, but this would be a waste of resources. The internal verification strategy yields a similar problem with a higher cost.

The investment of resources to verify a piece should be equally proportional to the perceived need to do so. At the same time, we can never fully trust another agent, as it may use a mixed strategy that builds up trust then lies consistently. Clearly, the equations we construct need to reflect this. We construct the probability of verification based directly on the probability that

A raw assumption that we should verify with the same probability that we expect to receive a fake piece would obviously drastically narrow the chance of catching it. Full trust

eliminates the chance of catching anything. Therefore, we introduce the constant λ to ensure that some minimum amount of verification occurs. Ideally, λ is equal to 0.10.

$$p_j^i(\text{verify}) = p_j^i(\text{fake}_j)(1-\lambda) + \lambda \qquad (3)$$

Building on equation (3), we now must consider what method to use. Instead of using a probability here, we instead use a threshold function based on the trustworthiness of the agent. This threshold will be defined as a constant σ, which represents a general assumption held by all agencies that there is a point at which external verification costs will exceed the value of internal verification, due to the fact that an agency may not always lie to other agencies. In other words, if one agency was lied to, the probability that another has been lied to is assumed to be reduced.

$$\text{choice}(\text{deal}_j) = \begin{cases} \text{accept} & r < p_j^i(\text{verify}) \\ \text{verify\_external} & r \geq p_j^i(\text{verify}) \text{ and } p_j^i(\text{fake}) \leq \sigma \\ \text{verify\_internal} & r \geq p_j^i(\text{verify}) \text{ and } p_j^i(\text{fake}) > \sigma \end{cases} \qquad (4)$$

$$\text{cost}(\text{verify}_j) = \begin{cases} \text{cost}(\text{verify\_external}) & p_j^i(\text{fake}) \leq \sigma \\ \text{cost}(\text{verify\_internal}) & p_j^i(\text{fake}) > \sigma \end{cases} \qquad (5)$$

$$\text{cost}(\text{deal}_j) = \text{cost}(\text{accept})(1 - p_j^i(\text{verify})) + \text{cost}(\text{verify}_j)\, p_j^i(\text{verify})$$

Note that external verification does not make any guarantee as the to the legitimacy of the piece. There are four possible values that can be returned by an external query: match, no match, verified match, and verified non-match. A match simply means the agent queried has an identical piece, though whether or not the data is fake remains questionable. A non-match implies the opposite unconfirmed result. A verified match means that the agent queried has a matching piece which has already been confirmed as legitimate. A verified non-match means that the piece submitted in the query is indeed fake.

The cost outlined in (5) is considered the expected average cost of trading with agency j and is applied to the strategy table found in Table 3 uniformly. The cost ultimately affects how each particular potential trading partner is evaluated. Due to the uniformity, it is omitted from the strategy table itself. The actual choice made once selecting an agent is determined by (4).

We consider the probability of an agent giving us a fake piece by comparing the number of pieces verified as valid versus those verified as invalid. This yields a simple ratio of verified good pieces over number of pieces verified from a source total. We consider V_i to be the set of all pieces verified as either fake or real (held or not), F_i as the set of all pieces verified as fake (note that $F_i \subseteq V_i$) and R_i^j as the set of pieces received during the game from agent j.

$$p_j^i(fake_j) = \begin{cases} \dfrac{count(F_i \bigcap R_i^j)}{count(V_i \bigcap R_i^j)} & \text{if } V_i \bigcap R_i^j \neq \emptyset \\ 0.5 & \text{otherwise} \end{cases} \qquad (2)$$

In the event that we have not verified any pieces from agent j, we assume neutrally that they will tell the truth 50% of the time.

For a given agency, once all strategy matrices for each player have been calculated for potential trade partners, we must decide on whom to deal with. From each matrix, we pick the move which yields the highest expected gain (what the agency gains sans what their opponent gains). The most dominant strategy value is selected, and the action is carried out.

6. Experiments

The goal of our experiments is to discover equilibriums among the players within the game. This will be done by running a number of selected scenarios, primarily to vary original trust, enough times to determine the majority winner among indeterminate variables (i.e. whom is selected when multiple choices are available with equal costs).

For each set of experiments, we begin by randomizing initial trust values among all participants. This reflects a randomly selected political environment which is independent of previous game results yet simulates how prior interactions have affected relations. The randomization also allows us to view whether or not having favor will yield a better strategic result.

Once each set has been generated, the scenario will be run 1000 times. Each iteration of the scenario will use the initially generated trust values and run for as many rounds as the game requires. We treat each game as a finite one due to the balance of interests in winning and political standing, and thus assume the iteration will terminate eventually.

During each iteration we collect a log file. This file is internally broken down by each round of interaction and trades. All chosen actions are recorded within each round, including whether or not a fake piece was given and subsequently discovered. Changes in trust are noted along with the new representing value.

The end of an iteration is noted within the log. The number of rounds taken along with the winner of the game are recorded, and a tally of how many times each player has won so far within the scenario is updated. In addition, tables in comma-separated value table are created from the in-game data, including number of times transactions between any two players occurred, and how often fake pieces were given. Most importantly, a comma-separated value table is generated on the fly which reflects how many pieces each agency has.

Through analysis of the tables, winners, and the strategies chosen, we hope to discover how well our game theory holds up against other behaviors. Each experiment is designed to present our theory with a particular challenge which we believe has merit with the real world. We recognize the fact that the nature and benefit of a behavior is largely dependent on the other chosen behaviors within the environment, and as such have setup three different configurations of each proposed experiment that reflect environments with similar ratios of strategies. Each configuration maintains a relatively equal ratio of participants.

In order to fully test our strategy calculations, we consider a number of possible opponent behaviors. While it is impractical to test all possible combinations, we consider scenarios in which we believe the constructed behavior outlined in this document is fully tested. These are listed in the following sections.

6.1 Pure Random

Competitive game theory requires a player to attempt to be as unpredictable as possible. In real life, an agency's strategy may be entirely unpredictable; they may not have a fixed strategy in how to interact, or they may simply not have as much of an interest in the benefits of finding the solution. Thus, the strategy chosen is done completely randomly, regardless of previous data.

6.2 Honest Random

Similar to pure random behavior, this particular variant focuses on sharing a real piece more often than a fake piece. The Pareto principle of 80/20 is used to determine when an agency chooses either strategy. This ratio was chosen based on the assumption that the principle, already found to have a wide variety of applications from analyzing social networks to predicting wealth distribution [10], represents a generic show of strategy. Coincidentally, this is precisely the ratio BitTorrerent seeks to maximize. [3]

This behavior represents a slightly refined approach to taking advantage of trust between other agencies within the coalition and tries to lie about one out of every five times it trades. Presumably, the other four times will be used to rebuild the reputation, while the lie may simply be blamed by the agency on bad data from other sources should they be inquired.

6. 3 Dishonest Random

Identical to Honest Random, but the emphasis is flipped to represent an agency which is much more likely to give fake pieces. This is basically a counterpart strategy that inverts the Pareto principle. This strategy represents an attempt by the agency to exploit the fact that it holds pieces necessary to win. It is impossible to avoid eventually trading with this agency for needed pieces.

6. 4 Benefit Random

Using the strategy table above, this behavior uses a weighted random behavior based on the netted value of a strategy. Although chosen randomly, an emphasis is placed on strategies that yield more valuable results. This makes a player unpredictable while allowing them to attempt to increase expected gains. Essentially, this forces a mixed strategy approach during the course of the game.

The benefit of such an approach is that it prevents other agencies from determinately predicting their moves. In theory, this makes it an excellent challenge to a pure game theory driven behavior, as it may 'fool' such an agency into believe one strategy will be used while invoking another. Essentially, this makes it a more refined version of the pure random approach, using the utility functions to drive where the strategy which will most likely benefit it.

6.5 Best Cost

This is the direct implementation of the work outlined in this paper. Using the strategy table outlined, each agency it can deal with is considered in a 1 on 1 game. A search is performed for a Nash equilibrium [NASH05], and if one exists, that strategy is used. Which game to play is based on the expected payoff of the dominant strategy coupled with a subtraction of the cost of performing a deal with that agency, which is based on equation 5. Our ideal results will prove this is the superior agency behavior.

This behavior represents an agency which is willing to make a determinate choice based on what it perceives as the most beneficial approach to solving the game before others do so. It runs the risk of being predictable, but should other agencies choose to use a similar approach, the nature of the strategy table will implicitly form an alliance within the coalition with little incentive to deviate. Those that lie or deviate will be less likely to be considered for trading.

7. Analysis of Results

7.1 Experiment 1 – Total Random vs. Best Cost

Assume that none of the agencies participating are rational, a common assumption in traditional game theory. We introduce a rational Best Cost behavior agency which is always in the minority. Our interest is to discover how well the agency can perform in the presence of completely random opponents.

Agency	Wins
TotalRandom_1	165
TotalRandom_2	180
TotalRandom_3	174
BestCost_1	511

Table 7.1.1. Configuration A: 4 Agent Game, 1000 Iterations

Agency	Wins
TotalRandom_1	97
TotalRandom_2	95
TotalRandom_3	99
TotalRandom_4	107
TotalRandom_5	94
TotalRandom_6	93
TotalRandom_7	107
BestCost_1	114
BestCost_2	107
BestCost_3	100

Table 7.1.2. Configuration C: 10 Agent Game, 1000 Iterations (3:7)

While the Best Cost behavior excels in a smaller environment, it appears that the ratio of like-minded agencies to totally random agencies is not beneficial. In fact, it appears the ability of an agency to win begins to fall drastically among the larger experimental sets. Analysis of winners within these situations reveals that they frequently start out by garnering a big lead early in the game. This appears to carry over in subsequent rounds as a form of inertia, though this is not necessarily a guarantee as evident in figure 7.1.1. Note that in this and future figures that the x-axis represents the round, while the y-axis represents pieces accumulated.

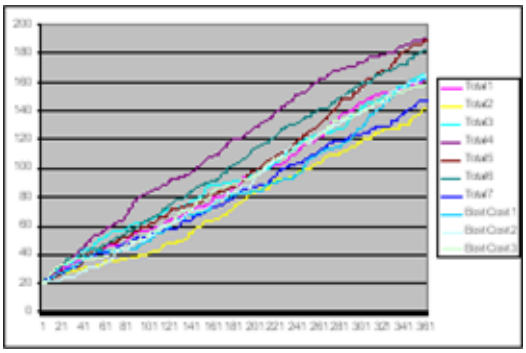


Figure 7.1.1. Iteration 4, Experiment 1, Configuration C

However, the Best Cost behavior does appear to still offer a minor edge over other more random players. Thus, in theory, such a behavior would offer a small benefit even when the strategies or motivations of opponents remain unknown.

7.2 Experiment 2 – Dishonest Random vs. Best Cost

Unlike the previous experiment, the results here reflect well on the ability of the Best Cost behavior to avoid interaction with those that are not beneficial to solving the problem. Since the algorithm allows recognition of those that do not give as many fakes, the three agencies are quick to interact with each other heavily. Thus, the only apparent reason this algorithm does not succeed in configuration A is due to a lack of other agencies sharing similar minded behavior.

Surprisingly, it appears that more Dishonest Random behavior agents are actually harmful to each other to the point of rarely being able to compete with the handful of Best Cost agencies.

Agency	Wins
DishonestRandom_1	286
DishonestRandom_2	315
DishonestRandom_3	288
BestCost_1	128

Table 7.2.1. 4 Agent Game, 1000 Iterations, Configuration A

Agency	Wins
DishonestRandom_1	116
DishonestRandom_2	128
DishonestRandom_3	115
DishonestRandom_4	118
DishonestRandom_5	110
BestCost_1	232
BestCost_2	205

Table 7.2.2. 7 Agent Game, 1000 Iterations, Configuration B

7.3 Experiment 3 – Benefit Random vs. Best Cost

The objective of this experiment is to determine whether or not a deterministic behavior with identical strategic priorities as that of a randomized behavior will succeed.

Agency	Wins
BenefitRandom_1	20
BenefitRandom_2	13
BenefitRandom_3	17
BestCost_1	956

Table 7.3.1. 4 Agent Game, 1000 Iterations, Configuration A

In this particular experiment, the success of the Best Cost behavior is due largely to two factors. First, all Random Benefit behavior agencies have a tendency to favor those that lie the least. Since the Best Cost strategy matrix rarely views a lie-based strategy as equilibrium with relatively

truthful agencies, the perceived deal cost for trading is substantially lower for all Best Cost behavior agencies.

Agency	Wins
BenefitRandom_1	0
BenefitRandom_2	0
BenefitRandom_3	0
BenefitRandom_4	0
BenefitRandom_5	0
BenefitRandom_6	0
BenefitRandom_7	0
BestCost_1	350
BestCost_2	326
BestCost_3	327

Table 7.3.2. 10 Agent Game, 1000 Iterations, Configuration C

Second, it's clear that these agencies have implicitly formed an alliance. As long as each agency is completely truthful, another agency which has lied just once will automatically be disqualified unless its competition is equally untruthful or it has pieces that are absolutely unavailable from any of the completely truthful agencies. Therefore, the Best Cost behavior agencies deal almost exclusively with each other.

These benefits lead to an incredible momentum among the one or more Best Cost behavior agencies. Figures 7.3.1 and 7.3.2 show exactly how much this momentum actually increases as the number of available like-minded agencies become available. However, it's important to note that only one of these agencies actually surges ahead of the rest significantly.

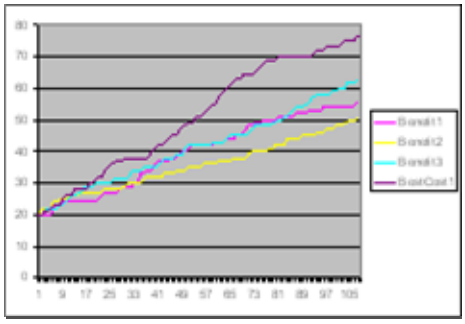


Figure 7.3.1. Iteration 887, Experiment 1, Configuration A

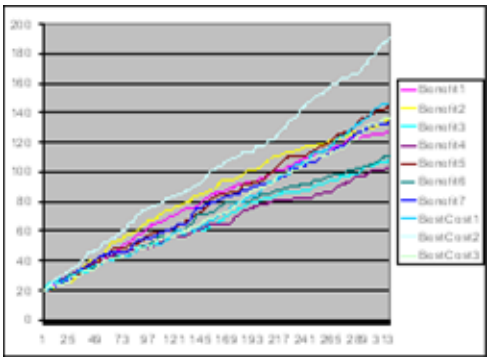


Figure 7.3.2. Iteration 887, Experiment 1, Configuration C

7.4 Experiment 4 – All Strategies

Perhaps the best test of any behavior is to consider a simulation in which the agencies participating have a diverse set of behaviors and interests of their own. Amidst all behaviors constructed and tested, it appears that the Best Cost behavior does indeed succeed at winning more frequently by at least a 3:1 ratio in all configurations. The competition is frequently close in each iteration, but once one of the Best Cost behavior agencies begins to gain considerable momentum, it tends to rapidly outperform the rest towards completion.

Agency	Wins
TotalRandom_1	36
TotalRandom_2	28
BenefitRandom_1	9
BenefitRandom_2	15
DishonestRandom_1	41
DishonestRandom_2	33
HonestRandom_1	32
HonestRandom_2	52
BestCost_1	374
BestCost_2	397

Table 7.4.1. 8 Agent Game, 1000 Iterations, Configuration B

8. Conclusions

Overall, it appears that the constructed behavior does perform better than other rudimentary approaches. However, the benefit provided appears to decrease in proportion to the number of agencies participating within the game. This phenomenon requires further investigation as to whether this is due to the underlying assumptions about rational players within game theory or simply a flaw within the algorithm itself.

An interesting factor that appeared within the game that was not previously considered was momentum of piece collection. When a round favors a particular agency in terms of successful trades of real pieces, the agency carried a leading 'edge' over opponents in terms of what is necessary and what is offered. Thus, it appears that agencies will often be able to surge ahead of the rest as long as enough pieces are in play.

However, in most of the experiments, the momentum appears to be short term at best, and towards the end the potential offers get very specific on the last piece necessary. In fact, towards the end, it appears that the winning agency often does so by only a single piece, often vying with another agency closely in the last twenty to thirty rounds. Recognition of this state could lead to a better version of the algorithm.

In order to improve the algorithm, one of our objectives is to consider past real world scenarios and attempt to analyze them in light of game theory. One of the variables we did not consider in our simulation which often plays an important role in real life is that of political influence. An agency with more influence has a greater chance to force allies to give up more data or be more honest in their dealings. However, this presents challenges in terms of managing a persistent public 'economy' and how to represent it, something which has presented a serious problem for game theory in the past. [7]

At this point in time, our research has yielded promising results. The algorithm appears to sufficiently recognize and beat other agencies with competitive behavior, though it has shown that it is not yet perfect. Investigation of these anomalies in our work is necessary to understand

the practical application of such work to the real world, but we believe that it is within reason to expect that such strategies could greatly benefit intelligence sharing within coalitions.

9. References

[1] Axelrod, Robert, *The Evolution of Cooperation.* Basic Books, New York, 1985.

[2] Buragohain, C., D. Agrawal, and S. Suri. "A game theoretic framework for incentives in P2P systems." Proceedings from the Third International Conference on Peer-to-Peer Computing, 2003.

[3] Cohen, Brian. "Incentives Build Robustness in BitTorrent" *In Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*, June 2003.

[4] Gupta, Rohit. and A. K. Somani. *"Game theory as a tool to strategize as well as predict nodes' behavior in peer-to-peer networks."* Proceedings from the 11th International Conference on Parallel and Distributed Systems, 2005.

[5] Halpern, Joseph and V. Teague, "Rational secret sharing and multiparty computation: extended abstract". *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing,* 2004.

[6] Harsanyi, John C., "Games with incomplete information played by 'Bayesian' players." *Management Science*, vol. 14, 1967.

[7] Myerson, Roger B., *Game Theory: Analysis of Conflict*. Harvard University Press, 1997.

[8] Nash, John, "Equilibrium Points in n-Person Games." *Proceedings of the National Academy of Sciences USA*, 36:48-49, 1950.

[9] Osbourne, Martin J. and A. Rubinstein, *A Course in Game Theory.* MIT Press, Cambridge, Mass., 1994.

[10] Scott, John. *Social Network Analysis : a Handbook.* Sage Publications, London, 1985.

Report #5.

Defensive Information Operations: DETECTING MALICIOUS EXECUTABLES USING ASSEMBLY FEATURE RETRIEVAL in an Untrustworthy Environment

Mohammad M. Masud, Latifur Khan, Bhavani Thuraisingham
Department of Computer Science, The University of Texas at Dallas,
Richardson, Texas
{mehedy, lkhan, bxt043000 }@utdallas.edu

Published as Technical Rreport: UTD-CS-47-06

ABSTRACT
As a part of defensive operations in an untrustworthy model, *we apply a novel, hybrid approach in detecting malicious executables. It is different from other approaches, because rather than considering only machine-bytes or assembly instructions of an executable, our approach combines byte-code with assembly-instructions to obtain an effective set of features. We are able to find useful assembly features using our two-phase Assembly Feature Retrieval (AFR) algorithm. We also apply an efficient technique to determine the assembly instruction(s) for any given n-gram (n-byte sequence of machine code), for any given executable. We also propose and implement a scalable solution to the n-gram feature extraction and selection problem. Our solution solves limited memory problem and applies efficient data structures to guarantee optimal running time. The AFR Model, a hybrid model for feature extraction, is presented. We apply our model to real instances of malicious executables. The results indicate that our model always performs better than other approaches that consider only byte-code or only assembly code to extract features.*

1 INTRODUCTION

In an untrustworthy model, defensive operations can be carried out in various ways. For example, a party may send malicious executables along with normal to other party and as a defensive operation the latter party has to detect this malicious executable. In this paper we investigate this issue.

Malicious executable is a code that harms computer systems. There are different kinds of malicious code such as: Worm, Virus, Exploit, Denial of Service (DoS), Flooder, Sniffer, Spoofer, Trojan and so on, which differ according to the way they attack computer systems and the malicious actions they perform. Some of them erase hard disks; some others clog the network, while some others sneak into the computer systems to steal away confidential and valuable information.

The traditional way of dealing with a known malicious executable is to apply signature based detection (unique telltale strings). But this approach has two serious problems. First, this approach involves significant amount of human intervention, and it may take long time (from days to weeks) to discover the signature. Thus, this approach is helpless against "zero day" attacks of computer worms. And second, this approach is hopeless against new attacks, i.e., it cannot detect new attacks. So, automated detection techniques are becoming more popular and are being developed by a number of researchers [1-8].

We propose a new model, called the Assembly Feature Retrieval (AFR) model that can automatically detect new malicious executables. Our approach is content-based, which means it

examines the content of the executables to determine its maliciousness. The principal idea of this approach is to extract interesting features from the content of known executables (i.e., the training set) and use these features to classify new executables (i.e., test data) using Data Mining. The main challenge of this approach is the extraction of relevant features that can efficiently distinguish between malicious and benign code. Our model applies a reverse engineering technique to extract features.

Malicious executables usually consist of byte-code, which are sequence of bytes that are specific to some operating systems and represents machine code for that system. If we could generate the exact replica of the high-level source code from which the byte-code has been generated, we would have acquired useful knowledge such as the structural, procedural and functional commonality among different malicious programs. This knowledge would then be efficiently applied to detect maliciousness of a program. But this is not possible at the state of the art. Auto-generation of high-level source code from a given byte-code, which is also known as de-compilation, is still an open research problem and far from being perfect. That is why some researchers are trying to detect maliciousness in executables by extracting features from the byte-code, like hex-string or *n*-grams of hex-dumps [9].

But there is an intermediate level between the high-level source code and the low-level machine code. This is the assembly instruction level. There are software tools available that can disassemble the machine executables into assembly instructions. By disassembling machine instructions, we are able to extract more meaningful and relevant features than we are able to extract from the byte-code. For example, if we use 4-grams as features, they may not be useful in distinguishing malicious code from benign ones. Because a 4-gram of a byte-code is a sequence of 4 bytes within the executable, which may represent part of a machine instruction, parts of more than one instruction, or just some data inside the code block. Thus, the information carried out by this 4-gram may be partial and incomplete. If we could decode this 4-byte block, and find out whether it is a part of any instruction(s), then we could have used that instruction(s) as features instead of this 4-byte, since the instruction(s) would carry more complete and meaningful information. This observation is the central motivation behind our model.

Our contributions to this research work are as follows. First, we build a new hybrid model for feature extraction, which utilizes the information obtained from disassembled executables as well as from byte-code, to extract useful features. We are able to find useful assembly features using our two-phase *feature retrieval* algorithm. Second, we apply an efficient address mapping technique, called *linear matching*, to find the assembly instruction sequences that represent an *n*-byte sequence of machine code (or *n*-gram, in short). Using this technique we can determine the assembly instruction(s) for any given *n*-gram, for any given executable. Third, we propose and implement a scalable solution to the *n*-gram feature extraction and selection problem. The problem is, given a set of malicious and benign executables, to find the best *n*-grams that can most efficiently distinguish between the malicious and the benign executables. Our solution not only solves limited memory problem but also applies efficient and powerful data structures to guarantee optimal running time. Thus, it is scalable to very large set of executables (in the order of thousands), even with limited main memory and processor speed.

Our model is novel and different from other content-based approaches because. Other approaches use only machine-code or only disassembled code to detect malicious executables whereas we use their (machine, assembly) combination for detection. We also extract other useful features, such as DLL usage and DLL function calls, from the disassembled executables and combine these features with assembly instruction features. We apply Support Vector Machine (SVM) on this feature set and perform cross validation to determine its classification accuracy. We compare the accuracy of our features with the accuracies of only byte-code or only assembly code features. We show that AFR model always extracts better features than the other two approaches.

The rest of the paper is organized as follows: section 2 discusses related works, section 3 presents and explains our model, section 4 shows the experiments and analyzes results, section 5 concludes with future directions. Attachments A and B describe algorithms and examples.

2 RELATED WORK

There are two main approaches to automate the detection of malicious executables: behavioral and content-based. Behavioral approaches analyze the behavior of messages like source-destination addresses, attachment types, message frequency etc. Examples of behavioral approaches are social network analysis [1, 2], and statistical analysis of outgoing emails [3, 4].

Some content-based approaches analyze the content of the message, and try to generate signature automatically. EarlyBird [5], Autograph [6], and Polygraph [7] are few examples of content-based worm detection system that generate signatures.

Some other content-based approaches extract features from the byte-code and apply machine learning to classify malicious executables. Stolfo et. al. [8] extracted DLL call information using GNU Bin-Utils [10] from the header of Windows PE executables. Also, they extracted string features using GNU *strings* program and used byte sequences obtained by hexdump as features. They report accuracy of their features using different classifiers.

The work that most closely matches our work is by Maloof et. al. [9]. They extracted *n*-gram features from the byte-code of executables and report high classification accuracy. Our approach is different from this work in that after extracting *n*-grams, we go one step further to extract assembly instruction sequences from the disassembled machine code. We show that this reduction increase efficiency of detecting new malicious executables.

3 THE ASSEMBLY FEATURE RETRIEVAL (AFR) MODEL

The Assembly Feature Retrieval (AFR) Model consists of different phases and components. Here we describe each of these in details.

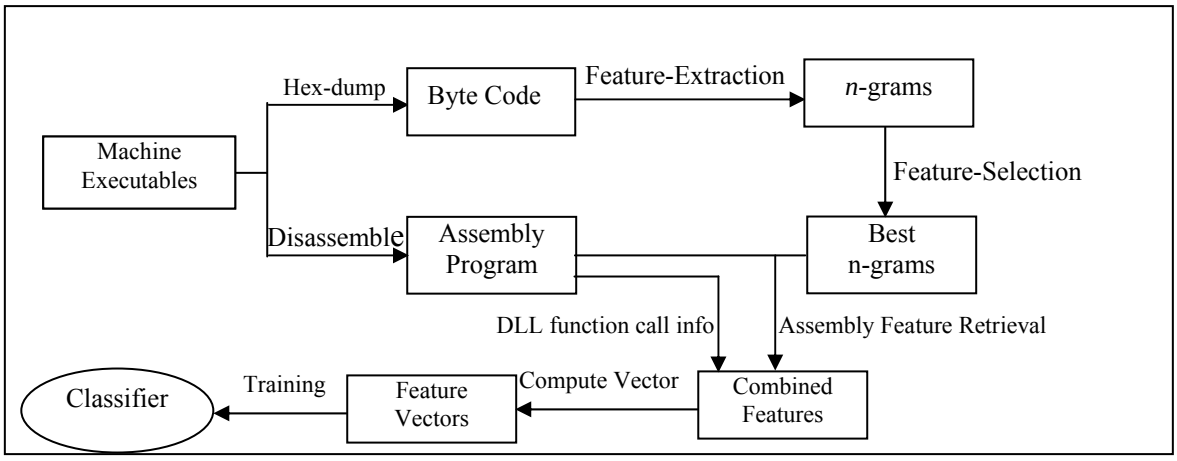


Fig. 1: Assembly Feature Retrieval Model (Training Phase)

Description of the Model

The Assembly Feature Retrieval (AFR) Model consists of two phases: the training phase and the testing phase. The training phase is shown in Figure 1, and explained in details in section 3.1. The

testing phase is shown in Figure 5, and explained in section 3.2. In the training phase we convert binary executables into byte-code files and Assembly Program files using UNIX Hex-dump utility, and a disassembly tool respectively. From the byte-code files, we extract *n*-gram features using our feature-extraction algorithm. Among large number of *n*-gram features, we select the best features using our feature selection algorithm. We then apply our AFR algorithm to retrieve assembly instruction sequences that best represent the selected *n*-gram features. These assembly sequences are then used as features in combination with the *n*-gram features, and a classifier is trained using the training set. In the testing phase, we extract the feature vector from the test file using the assembly features extracted in the training phase, and test this vector against the classifier. The classifier outputs the class prediction (benign, malicious) of the test file. Some of the major challenges we face in implementing the model are:

i.) *Too many features extracted:* The number of features generated in the feature extraction phase are so large that all of them cannot be stored in main memory. Again, because of large number of features, time to extract and store all features could be very high. But we have solved both these problems using very efficient data structures and scalable algorithms. Our solution is scalable to very large dataset. Our solution is explained in details in section 3.1.

ii.) *Byte-code features versus Assembly-instruction features:* We have found in our experiments that *n*-grams of byte-code features (i.e., *n*-byte sequence) show better classification accuracy than *n*-grams of assembly features (i.e., *n*-instruction sequence). But further research enabled us to adopt our hybrid approach, where we combine the byte-code features with assembly-instruction features using the AFR model. The resultant features show better performance than the other two. This approach is also explained in section 3.3.

The description of the major components of this model is given in the following sections. Although many examples in this section mention 4-gram features, they are also applicable for other values of *n*.

3.1 The Training Phase:

The Training phase is shown in Figure 1. In the training phase, we extract useful features from the training dataset, and train a classifier using SVM. The training dataset consists of real instances of malicious and benign executables. At first, we convert binary executables into byte-code files using UNIX Hex-dump utility, and into Assembly Program files using a disassembly tool called *pedisassem*. From the byte-code files, we extract *n*-gram features using our feature-extraction algorithm. Among large number of *n*-gram features that are generated, we select the best features using our feature selection algorithm. We then apply our AFR algorithm to retrieve assembly instruction sequences that best represent the selected *n*-gram features. These assembly sequences are then used as features. These assembly features are then merged with Dynamic Link Library (DLL) function call features. A classifier is then trained using these features. The major components of the training phase as explained below:

3.1.1 Hex-dump:

We apply the Unix hexdump utility (hexdump –x) to get the hexadecimal byte-code from the Executables. This utility outputs a sequence of byte-codes from the binary files in separate lines, where each line contains the address and eight columns of byte-code, each column containing a two-byte hexadecimal number.

Here is a sample two lines of output that hexdump generates:

00000f0 016e ... 6c2f 6e61 2f67
0000100 7453 … 1700 1800 000a

The first number in each line (e.g.: 00000f0) denotes the starting address of the line. Actually, the address is the offset from the starting of the executable. The next eight numbers represent the next 16-bytes of the binary. For example, the first two bytes of the first line are: 016e, where 01 is the upper byte (i.e., byte# 000000f1) and 6e is the lower byte (i.e., byte# 000000f0). We generate byte-code for each of the executables in the training set and save in separate disk files. We will refer to these files as *byte-code files*.

3.1.2 Disassembly

We have used a utility to disassemble the binary files and generate the assembly program corresponding to the binary. This open source software is called *PEDisassem*, which stands for Portable Executable Disassembler. It is written in java and can be obtained for free from [11]. It can generate assembly program from a Windows PE (Portable Executable) binary file, with detailed information like DLL usage, DLL function calls, resource information, and so on. Example 1 in Appendix 2 illustrates a sample disassembly generated by the disassembly tool. We generate assembly program for each of the executables in the training set and save in separate disk files. We will refer to these files as *assembly-program files*.

3.1.3 Feature extraction

Features are extracted from the byte-code files in the form of *n*-grams, where *n* = 2,4,6,8,10 and so on. Given a 11-byte sequence: *0123456789abcdef012345*,
The 2-grams (2-byte sequences) are: *0123*, *2345*, *4567*, *6789*, *89ab*, *abcd*, *cdef*, *ef01*, *0123*, *2345*.
So, the 2-grams are actually 2-bytes sliding window.
The 4-grams (4-byte sequences) are: *01234567*, *23456789*, *456789ab*,...,*ef012345*
Our *Feature Extraction Algorithm* (FEA) extracts all the *n*-grams for a given value of *n* from the byte-code files. This algorithm is explained in the following paragraphs.

3.1.4 Feature extraction algorithm (FEA):

Before going into details of the algorithm, we need to address the issue of limited memory, slow searching time, and how we solved it. For our 1435 executables dataset, there are 56 million 4-grams generated. If we store *n*-grams as strings, then the total memory required to store these 4-grams is around 448MB (= 56M x 8), since we need 2 bytes to store a 1-gram in string format. Also there are other information that need to be stored for each 4-grams, such as its positive frequency (i.e., total number of times it appears in the positive dataset), and its negative frequency (i.e., total number of times it appears in the negative dataset). We need these numbers to calculate information gain for each feature. So, we need at least 8 more bytes (two integers) to store these frequencies. Thus, the total memory requirement becomes at least 896 MB (= 56M x 16). Now, if these values are wrapped inside a class, then each object of the class may require more than 16 bytes, so the total memory requirement just to store the *n*-grams would go beyond 1GB. For higher grams, this requirement becomes higher. Unless we have a computer with tens of Gigabytes main memory, we are unable to store all these *n*-grams.

We have solved the memory problem using disk I/O. At first, some of the *n*-grams are kept in memory, until their total number exceeds a certain threshold. After that, they are written to disk and memory is cleared for storing another set of *n*-grams. If there are already some *n*-grams

stored in disk, we merge the content of the disk with the content of the memory and write the merged *n*-grams to disk. In order to ensure efficient merge operation, we store all the *n*-grams in lexicographically non-decreasing order, both in memory and disk.

The next issue is how to store the *n*-grams in memory. Although it seems trivial at a first glance, actually it is not. If we use a linear data structure like linked list or array to store the n-grams, the running time would be too high, because when an *n*-gram is extracted from the input file, we have to search whether it is already in the list. If it is in the list, we have to update the frequency of the corresponding element of the list; otherwise we have to add it to the list. Now, this search operation requires linear time. Even if all the *n*-grams could be stored in memory, it would require $O(N)$ searches to find one *n*-gram, where N is the total number of *n*-grams. So the total number of searches would be $O(N^2)$. From our previous example, given N=56 million, we have around 33.0×10^{14} searches in total. This would require several days on a 2GHz machine. So we must adopt a more efficient data structure to store the *n*-grams. We have used Adelson Velsky Landis (AVL) tree to store the *n*-grams in memory [12]. An AVL tree is a height-balanced binary search tree. It has the property that the absolute difference between the heights of the left sub-tree and the right sub-tree of any node is at most one. During insertion or deletion, if this property is violated, a balancing operation is performed and the tree regains its height balanced property. It is guaranteed that insertions and deletions are performed in logarithmic time. So, in order to search an *n*-gram in memory, we now need $\log_2(N) \approx 29$ searches. Thus, the running time is reduced about a million times (at most), which is of the order of a few minutes on the same machine! But in our experiments, it actually takes around two to three hours because of the overhead of disk I/O involved.

Now let us look back to the memory problem again. In order to merge the memory content with the disk content, we read n-grams from memory by traversing the tree in-order so that the sequence read out is sorted. The first *n*-gram read from the tree is compared with the first *n*-gram stored in disk. If they are the same, then their frequencies are added and the *n*-gram, along with its frequencies is written to a new disk file. If they are not the same, then the memory *n*-gram is written to new disk file if it is lexicographically smaller than the other and vice versa. The pointers are updated accordingly to read out the next *n*-gram. When all *n*-grams are written to the new file, the previous file is deleted and the new file is renamed to old file.

Algorithm 1 in Appendix 1 sketches the FEA. The **for** loop at line 3 runs for each byte-code file in the training set. The inner **while** loop at line 4 gathers all the *n*-grams for a file and adds it to the AVL tree. At line 7, we check whether tree size exceeds threshold value. If it exceeds, then either we save the contents of the tree in a new file (line 8) or merge the contents of tree with file (lines 9-11), and remove all the nodes from the tree (line 12).

3.1.5 Feature Selection Algorithm (FSA)

Since the number of extracted features is very large it is not possible to use all of them for training. To see why, first we notice that, if all the features are to be used, then we would need large amount of memory to store the feature vectors. Secondly, even if we were able to store the feature vectors in main memory, training time would be too long to be practical. Thirdly, any classifier would be confused with such a large feature set, because most of the features would be redundant or irrelevant. So, we are to choose a small, relevant and useful feature set from the very large set. We choose information gain as the selection criterion, because it is one of the best criteria used in literature for selecting the best feature(s) from a set of features. Our feature selection algorithm selects the best features from the all extracted features, using the information gain criterion.

Information gain can be defined as follows: "It is actually a measure of the effectiveness of an attribute in classifying the training data" [13]. If we split the training data on this attribute values, then information gain gives the measurement of the expected reduction in entropy after the split. The more an attribute can reduce entropy in the training data, the better the attribute in classifying the data. Information Gain of an attribute A on a collection of examples S:

$$Gain\,(S, A) \equiv Entropy\,(S) - \sum_{V \in Values\,(A)} \frac{|\,S_v\,|}{|\,S\,|} Entropy\,(S_v)$$

(1)

Where *Values(A)* is the set of all possible values for attribute A, and S_v is the subset of S for which attribute A has value *v*. *Entropy(S)* is the entropy of *S*. In our case, each attribute (i.e., feature) has only two possible values (0, 1). If this attribute is present in an example (i.e., executable) than the value of this attribute for that example is 1, otherwise it is 0. So, in our case, information gain would be as in (2):

$$Gain(S, A) \equiv Entropy(S) - \frac{P_0 + N_0}{P + N} Entropy(P_0, N_0) - \frac{P_1 + N_1}{P + N} Entropy(P_1, N_1)$$

(2)

where,

P = total number of positive examples

N = total number of negative examples

P_0 = total number of positive examples having attribute value 0

P_1 = total number of positive examples having attribute value 1

N_0 = total number of negative examples having attribute value 0

N_1 = total number of negative examples having attribute value 1

Note that, $P_0 + P_1 = P$ and $N_0 + N_1 = N$

Entropy of a given set of examples is a measure of impurity or homogeneity of that data set. If we have total *p* positive examples and *n* negative examples, then entropy of this set of examples is given by (3).

$$Entropy(p, n) = -\frac{p}{n+p}\log_2\left(\frac{p}{n+p}\right) - \frac{n}{n+p}\log_2\left(\frac{n}{n+p}\right)$$

(3)

Now, the next problem is to select the best features that have the highest information gain. In our experiments, we choose best 500, 1000, 2000, 4000, 5000, and 10000 features. Since the total number of features is very large, sorting them would require $O(N\log_2 N)$ time and $O(N)$ memory. As stated earlier, we cannot afford large memories to store all the *n*-grams. But we have adopted efficient data structures and used them smartly, which enabled us to select best *S* features in only $O(N\log_2 S)$ time using $O(S)$ memory. For S=500 and N=56 million, the gain in memory usage is 112,000 times and the gain in running time is almost three times. Here, we use the Heap data structure as a priority queue. Heap is a balanced binary tree with the property that the minimum (maximum) element of any sub-tree remains at the root of the sub-tree. So, a Min-Heap always has the minimum element at its root. Heap is implemented using array. We sketch our FSA in Algorithm 2 of Appendix 1. In order to find the best *S* features, for *S* = 500, 1000, 2000, 4000, 5000 and 10000. We create Heaps of size *S*. The outer **while** loop in line 1 iterates for all the *n*-grams saved in file during feature extraction. Line 2 reads the next *n*-gram from file; line 3 calculates its information gain. The inner **for** loop runs for each Heap. We use a Min-Heap, so that the *n*-gram with the minimum information gain is always at the root of the Heap. The next *n*-

gram is inserted into a Heap if the Heap has less than *S* elements (line 7). Otherwise, we check whether its information gain is less than that of the *n*-gram at the root of the Heap (line 8). If it is less, then we discard this *n*-gram and read the next one from file (line 9). Otherwise, we replace the root with the new *n*-gram and restore the Heap property in logarithmic time (lines 10-11). So, each insertion takes at most $O(\log_2 N)$ searches. For the 10,000 size Heap, each insertion takes only 14 searches in the worst case. We continue to read all the *n*-grams and inserting (or discarding) it into the Heap. When all *n*-grams have been processed in this way, the Heaps contain the best *S* attributes according to information gain.

3.1.6 Dynamic Link Library (*DLL*) *calls*

The assembly program file contains information about DLL function calls. Using this information, we can get the DLL function names that have been called from the program. Each "called" function is used as a feature. So, we extract all the DLL function features from all the assembly program files and again apply a selection process to choose the best 500 function calls according to information gain. These features are combined with the features retrieved using AFR algorithm.

3.1.7 Assembly Feature Retrieval (AFR)

Assembly feature retrieval is at the heart of the model. It is described in details in section 3.3. The assembly retrieval algorithm retrieves appropriate assembly instruction sequence of each selected *n*-

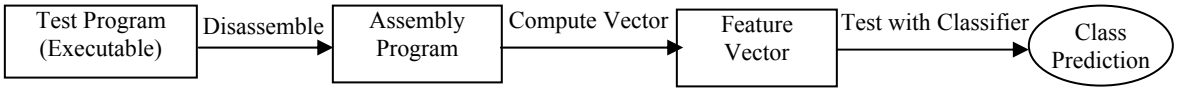


Fig. 2: The Assembly Feature Retrieval Model: Testing phase

gram feature, which are selected using the FSA. These sequences are then used as features.

3.1.8 Compute Vector

A feature vector consists of 0's and 1's. Each vector $Vi = \{V_{i,1}, V_{i,2}, \ldots V_{i,F}\}$, $i = 1 \ldots T$, where F is the total number of features, and T is the total number of training examples. So each element $V_{i,j}$ represents the value of feature j in example i. If this value is 0 then this feature is not present in this example and vice versa. We compute all these vectors by parsing the byte-code files as well as the assembly program files and searching the instruction sequences or byte sequences that this feature represents. These feature vectors are then used to train the classifier.

3.2 The testing phase:

The testing phase is sketched in figure 2. The first few components of the testing phase, i.e., Executable File, Disassemble, Assembly Program, Computer Vector, and Feature Vector are similar to those explained in training phase. We use SVM to test the example against the classifier obtained in training phase. The classification is one of {benign, malicious}

3.3 Assembly Feature Retrieval (AFR) Algorithm

The AFR algorithm is used to obtain assembly instruction sequences corresponding to the *n*-gram features. Before sketching the algorithm, we describe the problem with some examples and then explain how it has been solved.

The first problem is: given an *n*-gram feature how to find its corresponding assembly code. The code should be searched in all the assembly programs we generated in the training phase. The solution consists of several steps. First, we apply our *linear matching* technique: we use the offset address of the *n*-gram in the byte-code file to look for instructions at the same offset at the assembly program file. Based on the offset value, one of the three situations may occur:

i. the offset is before program entry point, so there is no corresponding code for the n-gram. We refer to this address as Address Before Entry Point (ABEP).
ii. there is some data at that offset, meaning this address points to data in code segment. We refer to this address as DATA.
iii. there is some code at that offset. We refer to this address as CODE.

If, for any *n*-gram, total number of CODE instances is no greater than *C* percent of the total number of instances, then we retain the *n*-gram feature as is, i.e., we do not replace it. The reason is that, although this *n*-gram represents very few assembly instruction sequences, it may contain some important encoded data. We call these *n*-grams as "essential *n*-grams". If total number of CODE instances is greater than *C* percent for any *n*-gram, then the *n*-gram is replaced with the assembly instructions. In our experiments, we set $C = 20\%$. There are multiple possible assembly code sequences for the same *n*-gram. We find all the occurrences of the *n*-gram to find all possible assembly code sequence that represents this *n*-gram. Example 2 in Appendix 2 illustrates this one-to-many mapping between *n*-gram and instruction sequences.

The second problem is to select the best sequence of instructions from all the instruction sequences. For ease of understanding, from now on we will refer to a sequence of consecutive assembly instructions as an "instruction-string" or only "string". We apply two different heuristics to find the best sequence:

i. The Most Frequent Substring (MFS) heuristic
ii. The Most Distinguishing String (MDS) heuristic

Before going into the details of the heuristics, we describe how instructions are represented in our approach.

3.3.1 Standard representation of instructions:

We adapt a standard representation for the instructions, so that we can easily store and compare it with other strings. We convert each instruction to this standard form as soon as we read it from memory.

a) Representing Instruction:
An Instruction I is a tuple, $I = \langle Q(I), P(I) \rangle$
Where, $Q(I)$: instruction name (character string) and
$P(I) = \{p_1(I),..., p_R(I)\}$; $0 \leq R \leq 2$: are parameters. So, each instruction may have zero, one or two parameters.

b)Representing Parameter:

Each parameter p_i may be one of four types:

i. *integer constant*
ii. *register variable*
iii. *memory variable*
iv. *port variable*

We use the function $T(p)$ to denote the type of parameter p. Each parameter p is also represented as a character string, which is one of the followings:

i. "*num*", if $T(p)$ is integer
ii. "*reg*", if $T(p)$ is a register
iii. "*mem*", if $T(p)$ is a memory
iv. "*port*", if $T(p)$ is a port

For example, the instruction: *mov eax, dword[eax+45]*

Will be represented by the string: "*mov.reg.mem*"

where, "*mov*" is the instruction name, "*reg*" is the type name for the register "*eax*" and "*mem*" is the type name for the memory variable "*dword[eax+45]*". We use a dot "." to separate different parts of the instruction.

c) Comparing instructions:

We define equivalence of instructions as follows. Two instructions x and y are equivalent if all of the following conditions are met:

i. $Q(x) = Q(y)$
ii. $R(x) = R(y)$
iii. $\forall i\ (T(p_i(x)) = T(p_i(y)))$

That is, two instructions are equivalent if they have the same names, same number of parameters and corresponding parameter types are the same. Actually, we can use standard string comparison functions to compare two instructions, since each instruction is represented as a string.

3.3.2 The Most Frequent Substring (MFS) heuristic:

Here, the best sequence is defined to be the sequence which has the highest frequency of occurrence in the given set of sequences. So, this problem reduces to finding the most common sequence among all sequences. It is possible that the best sequence is a substring of some sequence(s). Thus, this is a string matching problem where we want to find the Most Frequent Substring (MFS) that occurs among a set of strings.

One obvious hurdle in solving this problem is that a *null* string is also a substring of any string, so *null* will always be the MFS for any set of strings. So, we must set a *lower bound* to the number of characters in the MFS. If the lower bound is L, then the problem is to find the MFS with at least L characters. This problem can be solved by finding all possible Common Substrings (CSs) and picking the CS that has the highest occurrence. This problem can be formally defined as follows:

The best feature selection problem:

Let

$S = \{I_1, I_2, \ldots, I_N\}$: be the set of all instruction sequences. We will refer to each of these sequences as an *instruction string*. So, S is the set of instruction strings.

$S_{i,j} = \{I_i, \ldots, I_j\}$: be the subset of S containing all elements from I_i to I_j, $i <= j$

$I_i = \{I_{i,1}, I_{i,2}, \ldots, I_{i,j}, \ldots,\}$:be the ith instruction string, where $I_{i,j}$ is the jth instruction in the string

$X_{i,j,l} = \{I_{i,j}, I_{i,j+1}, \ldots, I_{i,j+l-1}\}$: a substring of the string I_i, starting from $I_{i,j}$ and having length l

$f(Y)$: frequency of occurrence of substring Y in the set of strings S. In other words, $f(Y)$ is the total number of strings in S of which Y is a substring.

The problem is to find the Most Frequent Substring MFS_L, having length >= L,

$$MFS_L = \underset{\forall i,j, i \le j \wedge l \ge L}{\arg\max} f(X_{i,j,l}) \quad (4)$$

A recursive solution to the best feature selection problem:

Let

$LCS(x, y)$ = Longest Common Substring of the strings x and y.

$$LCS_L(x,y) = \begin{cases} LCS(x,y), & if\ |LCS(x,y)| \ge L \\ \phi, & otherwise \end{cases} \quad (5)$$

That is, $LCS_L(x, y)$ is the longest common substring of x and y if its length >= L, and *null*, otherwise. Now, let us denote $MFS_L(i, j)$ to be the MFS of the subset $S_{i,j}$, having length >= L, which includes a substring of I_i. We define it recursively as:

$$MFS_L(i,j) = \begin{cases} I_i, \text{ if } i=j \\ LCS_L(MFS_L(i, j\text{-}1), I_j)), \text{ if } i<j \wedge LCS_L(MFS_L(i, j\text{-}1), I_j)) \neq \varphi \\ MFS_L(i, j\text{-}1), \text{ otherwise} \end{cases} \quad (6)$$

So, if the LCS_L between $MFS_L(i, j\text{-}1)$ and I_j is not null, then this LCS_L is the $MFS_L(i, j)$, otherwise, $MFS_L(i, j)$ is equal to $MFS_L(i, j\text{-}1)$. We assume that all sequences have at least L instructions. In order to determine the frequency of $MFS_L(i, j)$, we will define two more equations in (6) and (7):

$$f(LCS_L(x,y)) = \begin{cases} 1, & if\ LCS_L(x,y) \neq \phi \\ 0, & otherwise \end{cases} \quad (7)$$

That is, frequency of the LCS_L between x and y is 1, if this LCS_L is not null, and 0 otherwise. So the frequency of $MFS_L(i, j)$ can be defined as follows:

$$f(MFS_L(i,j)) = \begin{cases} 1, & if\ i = j \\ f(MFS_L(i, j-1)) + 1, & if\ i < j \wedge LCS_L(MFS_L(i, j-1), I_j) \neq \phi \\ f(MFS_L(i, j-1)), & if\ i < j \wedge LCS_L(MFS_L(i, j-1), I_j) = \phi \end{cases} \quad (8)$$

That is, frequency of $MFS_L(i, j)$ is one more than the frequency of $MFS_L(i, j\text{-}1)$ if we could find a non-null LCS_L between $MFS_L(i, j)$ and I_j, otherwise, it is the same as $MFS_L(i, j\text{-}1)$.

Now, we are ready to define our goal: the *MFS*. In fact, it is the MFS_L having highest frequency. In other words, the goal is to find:

$$MFS_L = \arg\max_{\forall i,j,i<=j} f(MFS_L(i,j)) \quad (9)$$

Solution to equation (9) involves invocations of the recursive equations (6) and (8). The solutions to equation (6) and (8) can be found using dynamic programming with two 2-dimensional arrays; one is $MFS_L[N\text{x}N]$ for storing all the MFS_L's and another is $f[N\text{x}N]$ for storing f 's. Then the solution to the equation (9) is the $MFS_L[i, j]$ for which $f[i, j]$ is the maximum. The entries of the tables Q_L and f can be filled up in a bottom-up fashion. The complexity of this algorithm would be $O(N^2K)$ where N is the total number of sequences and K is the worst case running time of *LCS* among any pair of sequences. Algorithm 3 in Appendix 1 sketches the dynamic programming to find MFS of a given set of strings S.

3.3.3 The Most Distinguishing Substring (MDS) heuristic:

The MFS heuristic selects the most frequently occurring substring among all the instruction strings. But there is no guarantee that MFS would be the best feature in distinguishing the malicious executables. In order to guarantee that we have such a feature in our feature set, we apply MDS heuristic. According to this heuristic, we select the instruction string that has the highest information gain. In order to find the MDS, we use a technique similar to the FEA (section 3.1.4) and FSA (section 3.1.5). We keep an AVL-tree for storing all the assembly sequences found using *linear matching*. When we find a new assembly instruction sequence, we insert it into the tree, updating P_i and N_i values (please see section 3.1.5) as necessary. When all sequences have been collected, we select the best 500 among them according to information gain criterion.

Algorithm 4 in Appendix 1 sketches the AFR algorithm. It consists of two phases. Phase I collects the assembly sequences corresponding to the *n*-grams, selected using FSA. Each byte-code file is scanned and *n*-grams are extracted. If the *n*-gram is among the selected ones (if found applying binary search), then we note the *offset* of the *n*-gram in the byte-code file and find all the assembly instructions at that offset in the assembly program file. These instructions are added to the list of instruction sequences corresponding to this *n*-gram. Phase II selects the best sequence using MFS and *MDS* heuristics. An illustrative example of the AFR algorithm is explained in Example 2, Appendix 2.

4 EXPERIMENTS

In this section we discuss out data set, experimental setup and results.

4.1 Data set

Our dataset consists of 597 benign and 838 malicious codes. The benign executables have been collected from different windows machines. The malicious executables have been collected from [16], which contains a large collection of malicious executables. We have used only a subset of this collection, under the names: “Email-worm”, “Net-worm”, “IM-worm” and “worm”. We have chosen only the Win32 executables.

4.2 Experimental setup

We implement our algorithms in Java with JDK 1.5. We use the libSVM library [17] for SVM. Our experiments are run on a Sun Solaris machine with 4GB main memory and 2GHz clock

speed. The disassembly and hex-dump are done only once for all machine executables and the resulting files are stored. We then run our experiments on the stored files.

4.2 Results

We have run our algorithm on the features selected by AFR model, as well as by Hex-gram only and Assembly-gram only model. We have applied different classifiers from different software suits: libSVM and WEKA. We have runn SVM using libSVM and ran Naive Bayes (NB) and Adaboost using WEKA. We report the results obtained for best 500 selected *n*-gram features only. Although we have run our experiments for best 1000, 2000 and larger number of selected *n*-gram features, we do not report them, because, increasing the number of selected features do not show any significant change in classification accuracy.

Table 1. Comparison of the classification accuracy between AFR model and other models. By byte-code 4-gram we mean 4 consecutive bytes of machine code, while assembly 4-gram means 4 consecutive assembly instructions

Classifier	AFR accuracy (%)	Byte-code 4-gram accuracy(%)	Assembly 4-gram accuracy (%)	Best Model
SVM[1]	97.3	95.2	90.6	AFR
NB[2]	91.5	87.6	84.0	AFR
Adaboost[2]	92.7	89.5	84.9	AFR
Avg.	**93.8**	**90.8**	**86.5**	**AFR**

[1]libSVM, [2]Weka

The results presented in Table 1 display the classification accuracies of different feature extraction models. The column under "AFR accuracy(%)" displays the accuracy of AFR model, which retrieves assembly features using 4-gram byte code features, combines these assembly features with *essential 4-gram* byte code features and DLL function call features. The column under "Byte-code 4-gram accuracy(%)" displays the accuracy of byte-code *n*-gram model. It reports the accuracy of the best 500 selected features. The row headed by SVM reports the accuracy of each of these models when SVM is used as classifier. For example, the first value of the row headed by SVM is 97.3%, which is the classification accuracy of AFR with SVM. It is 2.1% higher than its closest model, and 6.7% higher than the worst one. We see that accuracy of AFR is always the highest and at least 2.1% higher than its closest model. The average accuracy of AFR is also 3% higher than the second highest and 6.7% higher than the last one.

Table 2. Comparison of the classification accuracy using SVM between AFR model and other *n*-gram models for different values of *n*.

n	AFR accuracy (%)	Byte-code *n*-gram accuracy(%)	Assembly *n*-gram accuracy (%)	Best Model
4	97.3	95.2	90.6	AFR
6	97.6	95.5	87.4	AFR
8	97.1	94.6	88.3	AFR
10	97.2	95.1	73.6	AFR
Avg.	**97.3**	**95.1**	**86.5**	**AFR**

Table 2 reports the accuracies of different model for different *n*-grams using SVM. We choose SVM as the classifier, because as suggested by the results in Table 1, SVM provides the highest

accuracy on all models. Also, in table 2 we see that there accuracy of AFR is always highest irrespective of the value of *n*. The best accuracy for AFR is achieved with *n*=6, and it slightly decreases with incresed value of *n*. Accuracy of Byte-code *n*-grams is always lower than AFR, with 2.5% being the highest difference and 2.2% being the average difference. We also observe that the accuracy of Assembly *n*-gram becomes worse as the value of *n* is increased, with 73.6% being the worst value. The reason is that longer sequences of assembly code are rarer, and they cannot play any useful role in distinguishing between malicious and benign executables.

In summary, we find that our proposed hybrid model is always better than other *n*-gram models that rely only on byte code or Assembly instructions. This is because many *n*-gram features that are selected using the FSA (see section 3.1.5), bear only partial data, since they are part of one or more machine instructions. AFR retrieves the whole instructions and replaces the partial ones. In this way, AFR finds more complete and useful features. On the other hand, if anyone uses only assembly features, he may miss important encoded data that could be used as important feature. Thus, AFR model combines best features from both machine level bytes as well as assembly instructions and gains higher classification accuracy.

5 CONCLUSION

Our AFR model is a completely novel idea in malicious code detection. It extracts useful features from disassembled executables using the information obtained from machine executables. It then combines the assembly features with other features like DLL function calls and encoded machine-bytes. We have addressed a number of difficult implementation issues and provided very efficient, scalable and practical solutions. The difficulties we have faced during implementation are related with memory limitations and long running times. By using efficient data structures, algorithms and disk I/O, we are able to implement a fast, scalable and robust system for malicious code detection.

Preliminary results prove our claim that this model is going to produce very high accuracy. In future we would like to explore a number of different possibilities, such as: First, extracting useful assembly features directly from the assembly program without using byte-code features; Second, analyzing DLL function call pattern; Third, analyzing actions taken before or after each DLL call; and Finally, looking for a semantic signature or pattern instead of simple string patterns.

BIBLIOGRAPHICAL REFERENCES

[1] Golbeck, J., and Hendler, J. *Reputation network analysis for email filtering*. In CEAS (2004).

[2] Newman, M. E. J., Forrest, S., and Balthrop, J. *Email networks and the spread of computer viruses*. Physical Review E 66, 035101 (2002).

[3] Symantec Corporation. W32.Beagle.BG. Online, 2005. http://www.sarc.com/avcenter/venc/data/w32.beagle. bg@mm.html.

[4] Schultz, M., Eskin, E., and Zadok, E. MEF: *Malicious email filter, a UNIX mail filter that detects malicious windows executables.* In USENIX Annual Technical Conference - FREENIX Track (June 2001).

[5] Singh, S., Estan, C., Varghese, G., and Savage, S. *The EarlyBird System for Real-time Detection of Unknown Worms*. Technical report - cs2003-0761, UCSD, 2003.

[6] Kim, H. A. and Karp, B., *Autograph: Toward Automated, Distributed Worm Signature Detection*. in the Proceedings of the 13th Usenix Security Symposium (Security 2004), San Diego, CA, August, 2004.

[7] J. Newsome, B. Karp, and D. Song. *Polygraph: Automatically Generating Signatures for Polymorphic Worms*. In Proceedings of the IEEE Symposium on Security and Privacy, May 2005.

[8] M. Schultz, E. Eskin, E. Zadok, S. Stolfo, *Data mining methods for detection of new malicious executables*, in: Proc. IEEE Symposium on Security and Privacy, 2001, pp. 178--184.

[9] Kolter, J. Z., and Maloof, M. A. *Learning to detect malicious executables in the wild*. Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining Seattle, WA, USA Pages: 470 – 478, 2004.

[10] Cygnus. GNU Binutils Cygwin. *Online publication*, 1999. http://sourceware.cygnus.com/cygwin

[11] Windows P.E. Disassembler. http://www.geocities.com/~sangcho/index.html

[12] GoodRich, M. T., and Tamassia, R. *Data structures and algorithms in Java*. John Wiley & Sons, Inc. ISBN: 0-471-73884-0.

[13] Mitchell, T. Machine Learning. McGraw Hill, 1997.

[14] Weka: collection of machine learning algorithms for data mining tasks. http://www.cs.waikato.ac.nz/ml/weka/

[16] VX-Heavens: http://vx.netlux.org/

[17] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

ATTACHMENTS

I. Algorithms

Algorithm 1: The Feature Extraction Algorithm

```
Algorithm Extract_Feature (B)
B ={ B1 , B2, … , BK} : all byte-code files
1. T ← empty tree // Initialize AVL-tree
2. F ← new file  // Initialize disk file
3. for each Bi ∈ B do
4.     while not EOF(Bi) do  //while not end of file
5.         g ← next_ngram(Bi) // read next n-gram
6.         T.insert(g)  /* insert into tree and update
                    frequencies as necessary */
7.     If T.size > Threshold then //save or merge
8.         if F is empty then  F ←T.inorder
                    //save tree data in sorted order
9.         else Ft ← new file //initialize temporary file
10.            Ft ← merge(T.inorder, F.read)
               //merge tree data with file data and save
11.            F ← Ft // replace file with merged data
12.        T ← empty tree //release memory
```

Algorithm 2: The Feature Selection Algorithm

```
Algorithm Select_Feature (F, H, P, N)
F: the file containing all n-grams
H = {HS | HS is a heap of size S
          ∧ S ∈ {500, 1000, 2000, 4000, 5000, 10000}
P: total number of positive examples
N: total number of negative examples
1. while not EOF(F) do
2.     <g, p0, n0, p1, n1> ← next_ngram(F)
       //read n-gram with frequency counts
5.     gain ← Gain(p0, n0, p1, n1, P, N) // using (2)
5.     for each HS ∈ H do
7.         if HS.total_Nodes< S then HS.insert(g, gain)
8.         else if gain <= HS.root.gain then
               continue //discard lower gain n-grams
12.            else HS.root ← <g, gain> //replace root
14.        HS.restore   //apply restore operation
```

Algorithm 3: Finding the Most Frequent Sequence

```
Algorithm Find_MFS(S, L) returns MFS of S
S: {I1, … , IN}: the set of instruction strings
L: lower limit of string length
MFSL[NxN]: Table to hold MFSL[i,j] values
//see (6)
fL[NxN]: Table to hold fL[i,j] values //see (8)
1. for i=1 to N do MFSL[i,i]=Ii //initialize
2. MFS ← φ, max ← 0 //initialize
3. for l=2 to N-1 do //for all possible lengths
4.     for i=1 to l-1 do
5.         if LCSL(MFSL[i, j-1], Ij) ≠ φ then
6.             MFSL[i, j] ← LCSL(MFSL[i, j-1],
Ij)
7.             fL[i, j] ← fL [i, j-1] + 1
8.         if fL[i, j] > max then
9.             max ← fL[i, j], MFS ← MFSL[i,
j]
```

Algorithm 4: The Assembly Feature Retrieval

Algorithm Assembly_Feature_Retrival(*G, A, B*)

$G = \{g_1, g_2, ..., g_M\}$: the selected *n*-gram features

$A = \{A_1, A_2, \ldots, A_L\}$: all Assembly files

$B = \{B_1, B_2, \ldots, B_L\}$: all Byte-code files

M = #of selected features

L = #of training files

Define

$S_i = \{S_{i,1}, \ldots, S_{i,j}, \ldots\}$: the assembly instruction sequences corresponding to $g_{i, 1 \le i \le M}$

$S_{i,j} = \{I_{i,j,1}, \ldots, I_{i,j,k}, \ldots\}$: the *j*th instruction sequence of $S_{i, 1 \le i \le M}$; $I_{i,j,k}$ is the *k*th instruction of $S_{i,j}$

LCS(*X*,*Y*) : the longest common substring of the two sequences *X* and *Y*.

1. **for** $l = 1$ to M **do** $S_i \leftarrow$ empty //initialize lists

//phase I: sequence collection

2. **for** each $B_l \in B$ **do**

3. *offset* ← 0 //current offset in file

4. **while** not EOF(B_l) **do** //while not end of file

5. $g \leftarrow$ next_ngram(B_l) // read next *n*-gram

6. <*i*, *f*> ← binarySearch(*G*,*g*) // *seach g in G*

7. **if** *f*=**true then** // found

8. $j = |S_i| + 1$ //add another sequence

9. $S_{i,j} \leftarrow$ empty // initialize new sequence

10. **for** each instruction *r* found within the address range [*offset*, *offset* + *n*] of A_l **do**

11. $S_{i,j} \leftarrow S_{i,j} \cup r$ //add to the sequence

//phase II: sequence selection

12. **for** $i = 1$ to M **do** //for each S_i

13. *MFS* ← Find_MFS(S_i)

14. *MDS* ← Best sequence according to Info_gain

II. Examples

Example 1: Disassembly of an executable

```
Disassembly of File: OINFOP11.EXE

DateStamp=3F0FD005:Sat Jul 12 02:08:21 2003
Code Offset=00000400, Code Size = 0019E00
Data Offset=0001A200, Data Size = 0000800
Number of Objects = 0003 (dec), Imagebase = 30000000h
Object01:  .text   RVA: 00001000 Offset: 00000400 Size: 00019E00 Flags:
60000020
++++++++++++++++++ RESOURCE INFORMATION
Number of Resource Types =    4 (decimal)
Resource Type 001: REGISTRY
    … … …  (rest is omitted for brevity)
++++++++++++++++ STRING INFORMATION
Number of Strings =    1 (decimal)
Name: StringId_0007     "OffProv11"
++++++++++++++++++ IMPORTED FUNCTIONS
Number of Imported Modules =   11
Import Module 001: OInfo11.OCX
… … …  (some are omitted for brevity)
Import Module 011: WINSPOOL.DRV
++++++++++++++++++ IMPORT MODULE DETAILS
Import Module 001: OInfo11.OCX
Addr:00019BDC hint(0005) Name: GetOfficeData
… … …  (rest are omitted for brevity)
Import Module 002: KERNEL32.dll
Addr:00019E8A hint(0103) Name: GetCommandLineA
… … …  (rest are omitted for brevity)
++++++++++++++++++ EXPORTED FUNCTIONS
Number of Exported Functions =   10 Addr:300122C7 Ord:   1 (0001h) Name:
_ctime64
… … …  (rest are omitted for brevity)
++++++++++++++++++ Possible Strings Inside Code Block
:3000149C....NullString..Invalid DateTime
… … …  (rest are omitted for brevity)
++++++++++++++++++ ASSEMBLY CODE LISTING
//****** Start of Code in Object CODE Program Entry Point = 30011BAD
(OINFOP11.EXE File Offset:00000400)
:30001000 A89F0100               DWORD 00019FA8           ;; ....
… … …  (some are omitted for brevity)
:30001604 43                     inc ebx
:30001605 4F                     dec edi
:30001606 66666963654F62         imul sp, word[ebx+65], 624F
… … …  (rest are omitted for brevity)
```

Example 2: working of AFR algorithm, and example of one-to-many relation from *n*-gram to assembly instructions.

This is an example of the collection of assembly sequences corresponding to the *n*-gram "00005068". Note that this *n*-gram has 9 occurrences. The bolded portion of the op-code in Table 3 represents the *n*-gram.

Table 3. Assembly code sequences for "00005068".

No.	Op-code	Assembly code
$I_{1,1}$	E8B702**0000**	call 00401818
$I_{1,2}$	**50**	push eax
$I_{1,3}$	**68**28234000	push 00402328
$I_{2,1}$	0FB6800D02**0000**	movzx eax,byte[eax+20]
$I_{2,2}$	**50**	push eax
$I_{2,3}$	**68**CC000000	push 000000CC
$I_{3,1}$	8B805C04**0000**	mov eax,
$I_{3,2}$	**50**	dword[eax+45]
$I_{3,3}$	**68**01040000	push eax
		push 00000401
$I_{4,1}$	0FB6800D02**0000**	movzx eax,byte[eax+20]
$I_{4,2}$	**50**	push eax
$I_{4,3}$	**68**CC000000	push 000000CC
$I_{5,1}$	689010**0000**	push 00001090
$I_{5,2}$	**50**	push eax
$I_{5,3}$	**68**3C614000	push 0040613C
$I_{6,1}$	E8841C**0000**	call 004032CC
$I_{6,2}$	**50**	push eax
$I_{6,3}$	**68**207F4000	push 00407F20
$I_{7,1}$	8D801001**0000**	lea eax, dword[eax+110]
$I_{7,2}$	**50**	push eax
$I_{7,3}$	**68**07504000	push 00405007
$I_{8,1}$	25FFFF**0000**	and eax, 0000FFFF
$I_{8,2}$	**50**	push eax
$I_{8,3}$	**68**E8164100	push 004116E8
$I_{9,1}$	25FFFF**0000**	and eax, 0000FFFF
$I_{9,2}$	**50**	push eax
$I_{9,3}$	**68**600E4100	push 00410E60

These instruction sequences or "instruction strings" are collected in *Phase I* of the AFR algorithm. If the lower limit, $L = 2$, then the best instruction sequence according to MFS heuristic appears to be "*push push*", having frequency = 10. If $L = 3$, then there are multiple solutions with frequency = 2: "*and push push*" , "*call push push*", "*mov push push*" and so on. The instruction strings selected using MDS heuristics depened on the information gain of the particular string.

APPENDIX A:

ROLE-BASED ACCESS CONTROL AND USAGE CONTROL POLICIES FOR INFOSPHERES

Ravi Sandhu, Min Xu; George mason University

Bhavani Thuraisingham, The University of Texas at Dallas

To be published as a GMU Technical Report

1. THE PROBLEM

In the work discussed in report #2, we examined basic role-based access control policy for secure data sharing and conducted experiments to determine the amount of information that is lost due to enforcing security. While the access control policies utilized in this paper is a useful and flexible policy, the security community is moving towards a full-scale role-based access control model and more recently the usage control model. However none of these models have been examined for a coalition environment. The problem is to take advantage of the features offered by both RBAC and UCON and develop security models for the global infospheres. In this project we are examining the use of RBAC and UCON for Assured Information Sharing. We discuss some of the issues and challenges in this appendix

2.BACKGROND

RBAC: The seminal proposal on role-based access control by Sandhu et al [SAND96] introduced a general family of RBAC models called RBAC96. Subsequent work by Sandhu and his team, as well as other researchers in the community, established that RBAC is capable of expressing a wide range of policies of strong practical interest by using simple concepts. It has been demonstrated how to do conventional discretionary and mandatory access controls using RBAC, so RBAC truly encompasses previous access control models. Due to strong commercial interest by vendors and users of RBAC, the model evolved into a NIST/ANSI standard model first introduced in 2001 [FERR01] and formally adopted as an ANSI standard in 2004. The principal idea in RBAC is that users and permissions are assigned to roles, thereby users acquire permissions indirectly via roles (Figure 3).

UCON: The concept of Usage Control (UCON) was recently introduced in the literature by Park and Sandhu [PARK04]. In recent years there have been several attempts to extend access control models beyond the basic access matrix model of Lampson, which has dominated this arena for over three decades. UCON unifies various extensions proposed in the literature in context of specific applications such as Trust Management and Digital Rights Management. The UCON model provides a comprehensive framework for next generation access control. A UCON system consists of six components: subjects and their attributes, objects and their attributes, rights, authorizations, obligations, and conditions. The authorizations, obligations and conditions are the components of the usage control decisions. Another aspect that UCON extends traditional access control models is the concepts of continuity and mutability (Figure 4).

3. Related Work

Since RBAC was introduced by Sandhu and his colleagues several researchers have adapted this model for various applications. For example, Bertino et al [BERT05] have developed a temporal

authorization model based on RBAC. Osborne et al [OSBO04] have developed a model for XML documents based on RBAC. Thuraisingham has examined security for the semantic web based on an RBAC-like model [THUR05]. Applying RBAC for a coalition environment is yet to be carried out.

In the case of UCON model, Sandhu and his students have done pioneering work [PARK04]. For the first time there is now a model that encompasses all the other models. Sandhu et al have also extended UCON to handle temporal primitives. The development of UCON is still in the early stages and its application to a coalition environment has yet to be carried out.

4. Technical Issues

RBAC: RBAC is especially relevant to the protection of information in a local infosphere as well as in a global infosphere across a coalition. Administration of roles and cross-organizational roles, which are central to deployment of RBAC in infospheres are not addressed in the NIST/ANSI standard. Traditional approaches to RBAC administration often are heavyweight involving explicit actions by human administrators. These traditional approaches where a human is in the loop in every administrative decision are not scalable to the flexible and automated environment of an infosphere. Recently Sandhu and his students have introduced lightweight administration models for RBAC based on user attributes [ALKA02] and have also examined interaction of roles and workflow [KAND02]. One needs to develop administrative models for RBAC in infospheres with the goal of being as lightweight and seamless as possible without compromising security.

UCON: The new expressive power brought in by UCON is very germane to the automated and seamless security administration required in infospheres. For example, an authorization rule permits or denies access of a subject to an object with a specific right based on the subject and/or object attributes, such as role name, security classification or clearance, credit amount, etc. There may be different meanings attached to the authorization rules enforced by different local infospheres. These differences have to be reconciled. UCON is an attribute-based model, in which permission is authorized depending on the values of subject and object attributes. In a global infosphere, the challenge is to export policies that depend on the attribute values and the roles. UCON model also consists of obligations and conditions. For example, playing a licensed video file by organization A requires a user to click a notice and register in the organization's web page. Such an action can be required before or during the playing process. Mutability in UCON means that a subject or object attribute value may be updated to a new value as a result of the access. The impact of these features in a global infosphere is yet to be examined.

5. Our Approach: Role Based Access Control and Usage Control for Infospheres

Secure information sharing, within and across infospheres, requires the enforcement of persistent access control, whereby access controls on information objects persist even as these objects reside on computers outside the immediate control of the information source. Persistent access control is a form of dissemination control (DCON) where the access policy to be enforced is inextricably linked with the object as it is moved from place to place in cyberspace. There are two major challenges in achieving this goal.

- How to enforce access controls on objects as they are physically resident on multiple computers, including end-user client computers?
- What kinds of policies are appropriate for these situations and how should they be specified?

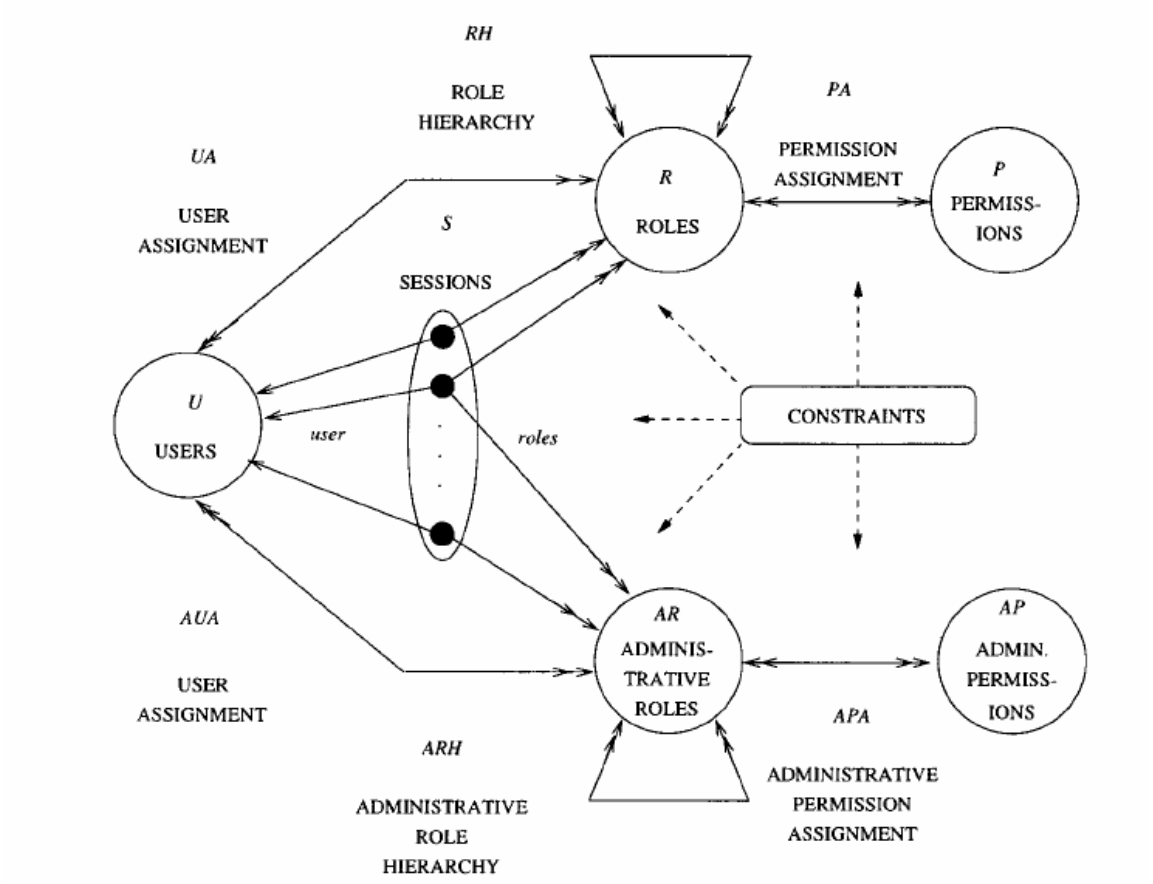


Figure 1. -Role-based Access Control (RBAC96

The first of these challenges is addressed by emerging trusted computing technologies (including the Trusted Computing Group's Trusted Platform Module, Intel's LaGrande Technology and Microsoft's Next Generation Secure Computing Base), which are anticipated to see widespread use in the near future. Recent work by Ravi Sandhu and his student Xinwen Zhang [ZHANG05] in this arena has demonstrated the enforcement of persistent access control both by ensuring that the object can be accessed only on a suitably trustworthy platform and by a suitably authorized user. Trust in the platform is established by integrity measurement and attestation protocols. Trust in the user is based on the user's identity and the user's attributes on the basis of a suitable public-key infrastructure. These technologies are expected to be widely available commercially in the next two to three years.

In comparison progress on the second challenge has been much slower, partly because until recently commercially viable technologies for persistent access control were not available. Given the recent push to bring these technologies to market the question of how to effectively use them to facilitate controlled information sharing in a coalition environment has become much more compelling. This second challenge is directly addressed in this project.

This project is developing a series of models for information sharing in coalitions based on Ravi Sandhu's pioneering work on role-based access control (RBAC) and usage control (UCON). The space of information sharing policies is extremely rich and varied [THOM04]. We partition

this space in two distinct dimensions so as to build these models in a systematic manner. The first dimension distinguishes whether or not the information content in a disseminated object can be changed as the object is further re-disseminated. There are two alternatives here as follows.

- **Read-only information sharing:** In this alternative the content of an object cannot be changed as it gets disseminated. The information content remains as it was when the object was created by its source.
- **Read-write information sharing:** In this alternative the content of an object does change as it gets disseminated. There are a number of sub-cases depending on how the information can change. One possibility is to add annotations and notes to the base content which itself does not change. Another possibility is to redact material in the process of downgrading the security level of the content. Further, the content may be modified by replacing portions of the original content with new content.

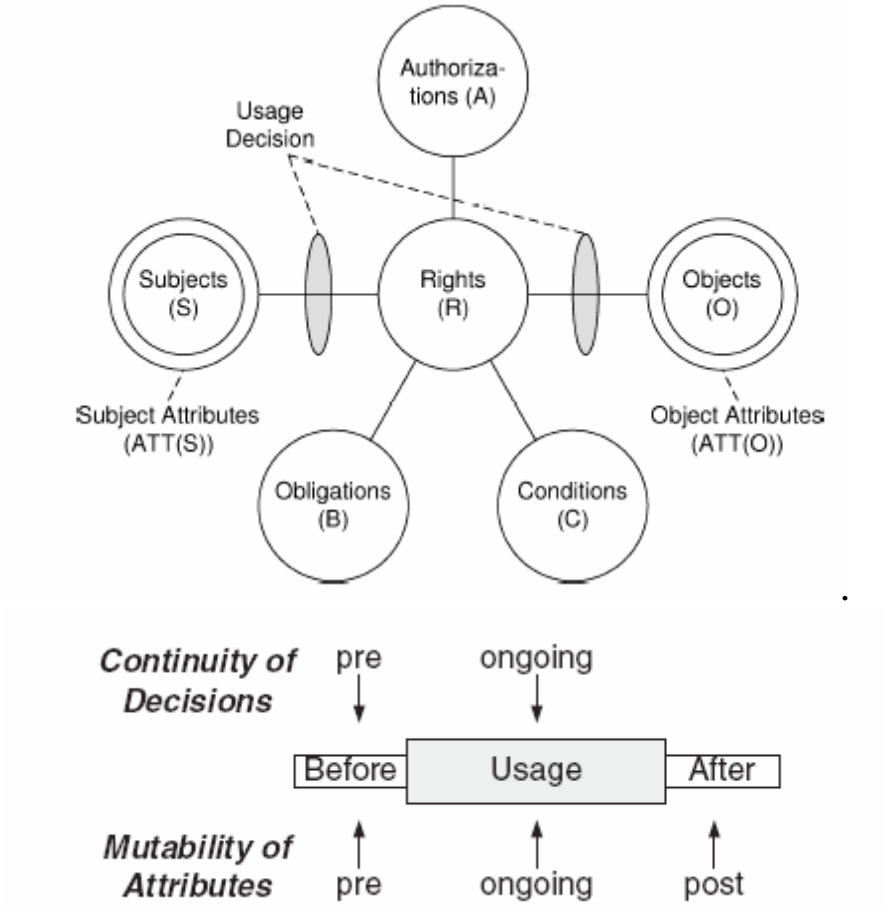


Figure 2. UCON Components

While the read-only certainly has practical applications the main purpose for treating it separately is to follow the dictum of "walk before you run." By focusing first on the read-only case it is possible to understand the issues that arise here clearly before taking on the more difficult task of dealing with writes. This incremental approach has been very productive in previous research on security models by Ravi Sandhu and seems to be the best approach for constructing models in a complex space.

The second dimension for partitioning the space of information-sharing policies is based on the scale of dissemination. In this dimension we are studying the following alternatives.

- **Small scale:** In small-scale dissemination the number of individuals who can access an object is of a small magnitude such as 10. This scale of dissemination is appropriate for the most sensitive content. At this scale it hardly seems appropriate to have very complex models. Dissemination can occur in the simple form of individual to individual (or point to point) dissemination. Some form of basic originator control where the intent of the source if the object is carried through a series of individual disseminations is the most appropriate policy. Nonetheless there are significant issues that arise. These include issues of revocation, cascading revocation, off-line access, limits on access (number of times, duration, etc.), prohibition of access (often expressed as negative rights) and transfer-only dissemination (in contrast to copy dissemination). These issues remain to be systematically addresses even in this small-scale context.

- **Medium scale:** In medium-scale dissemination the number of individuals who can access an object is of a larger magnitude such as 100's or 1000's. At this scale dissemination is best accomplished by models based on user and object attributes such as security labels, roles and other appropriate properties. The issues raised in small-scale dissemination continue to be significant here as well. In addition issues of role-to-role dissemination and delegation also arise.

- **Hybrid scale:** Hybrid scale offers a novel combination of the above two cases proposed for the first time in this project. The fundamental idea is that truly sensitive information needs to be confined to a few individuals so that actual dissemination must be small scale. Nevertheless it is impractical to achieve small-scale dissemination entirely by individual-to-individual dissemination. This is especially so in highly dynamic and mission critical applications such as the ones that the military faces. Information needs to be available to appropriate individuals when they need it. Deciding who precisely these individuals are in advance is unrealistic. Our proposal is to distinguish potential from actual dissemination. Potential dissemination is based on roles and security labels just as in the medium scale case. Actual dissemination, however, is based on the count of individuals who actually see the content. Thus a mission plan may be available to all officers of a certain rank of a coalition partner, but actual access may be limited to a small number, say, two or three. Morever during a combat situation these limits may be relaxed so actual access is available to a larger number, such as ten or fifteen. Conversely, occurrence of combat may limit the number even further to the one officer of appropriate rank who is on duty at that moment. The main goal is to enforce a small-scale of actual dissemination without pre-specifying the actual individuals who make the access, while at the same time allowing for automatic adjustments in these policies as circumstances in the real world change. The combination of RBAC and UCON is particularly powerful for expressing such hybrid policies.

Combining these two dimensions we get six combinations to investigate. This project is systematically investigating this space using a combination of RBAC and UCON to develop a series of novel models in this arena.

6. Progress

After examining both RBAC and UCON, we have selected some of the appropriate features from both models and developed a model call RBUC (Role-based usage Control). RBUC is illustrated in Figure 3. RBUC integrates RBAC and UCON to provide flexible access control for coalition environment, not only has RBAC flexibility , but also has great UCON features, such as use control, continuity of decisions and mutability of attributes.

Applicability of RBUC for a coalition environment is illustrated in Figure 4. The coalition partners maybe trustworthy), semi-trustworthy) or untrustworthy), so we can assign different roles on the users (professor) from different infospheres, e.g.

- professor role,
- trustworthy professor role,
- semi-trustworthy professor role,
- untrustworthy professor role.

We can enforce usage control on data by set up object attributes to different roles during permission-role-assignment, e.g.
professor role: 4 times a day,
trustworthy role: 3 times a day
semi-trustworthy professor role: 2 times a day,
untrustworthy professor role: 1 time a day

More details will be given in a forthcoming report at GMU and will be discussed in our FY07 annual report.

Figure 3. RBUC for Coalitions

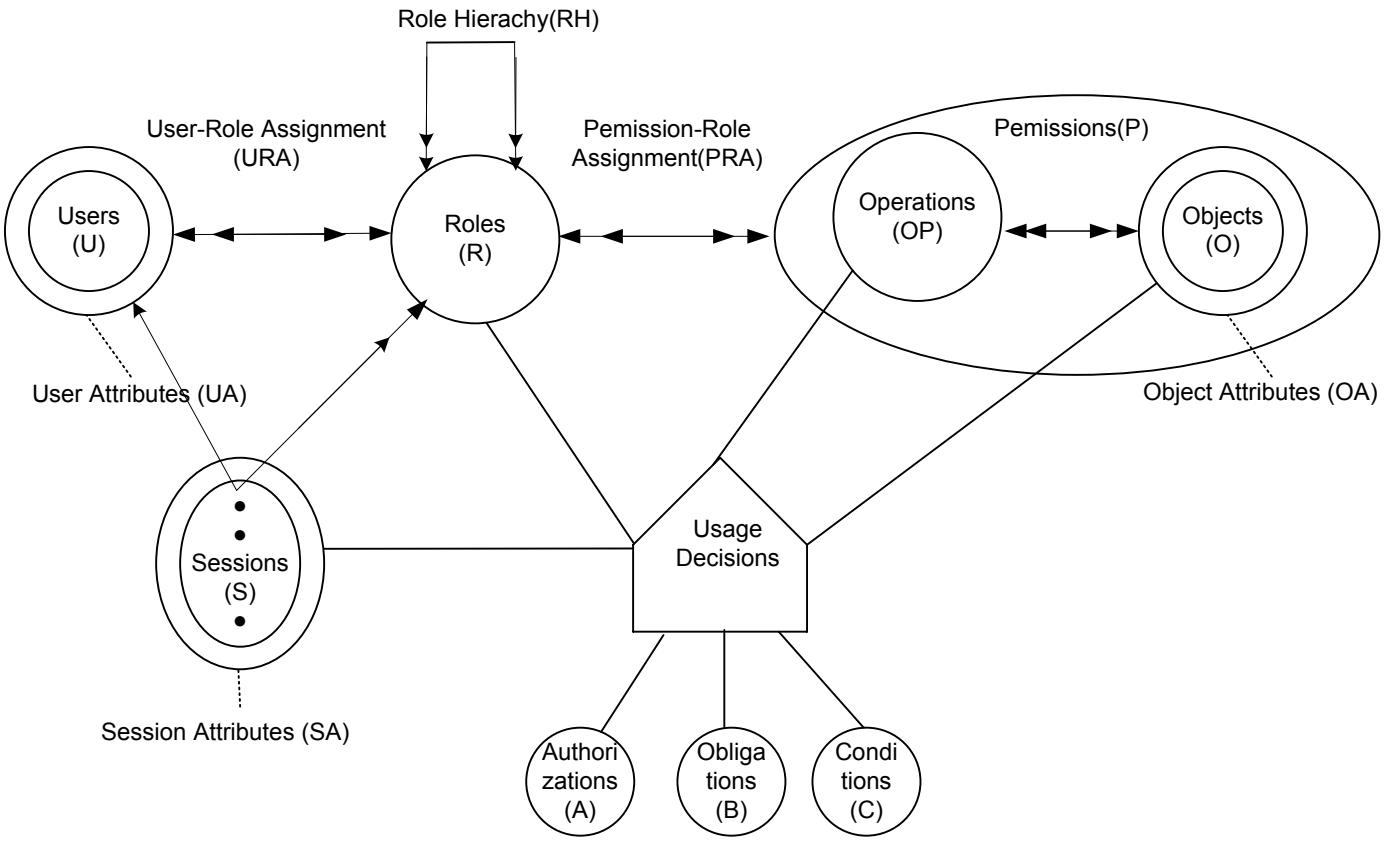
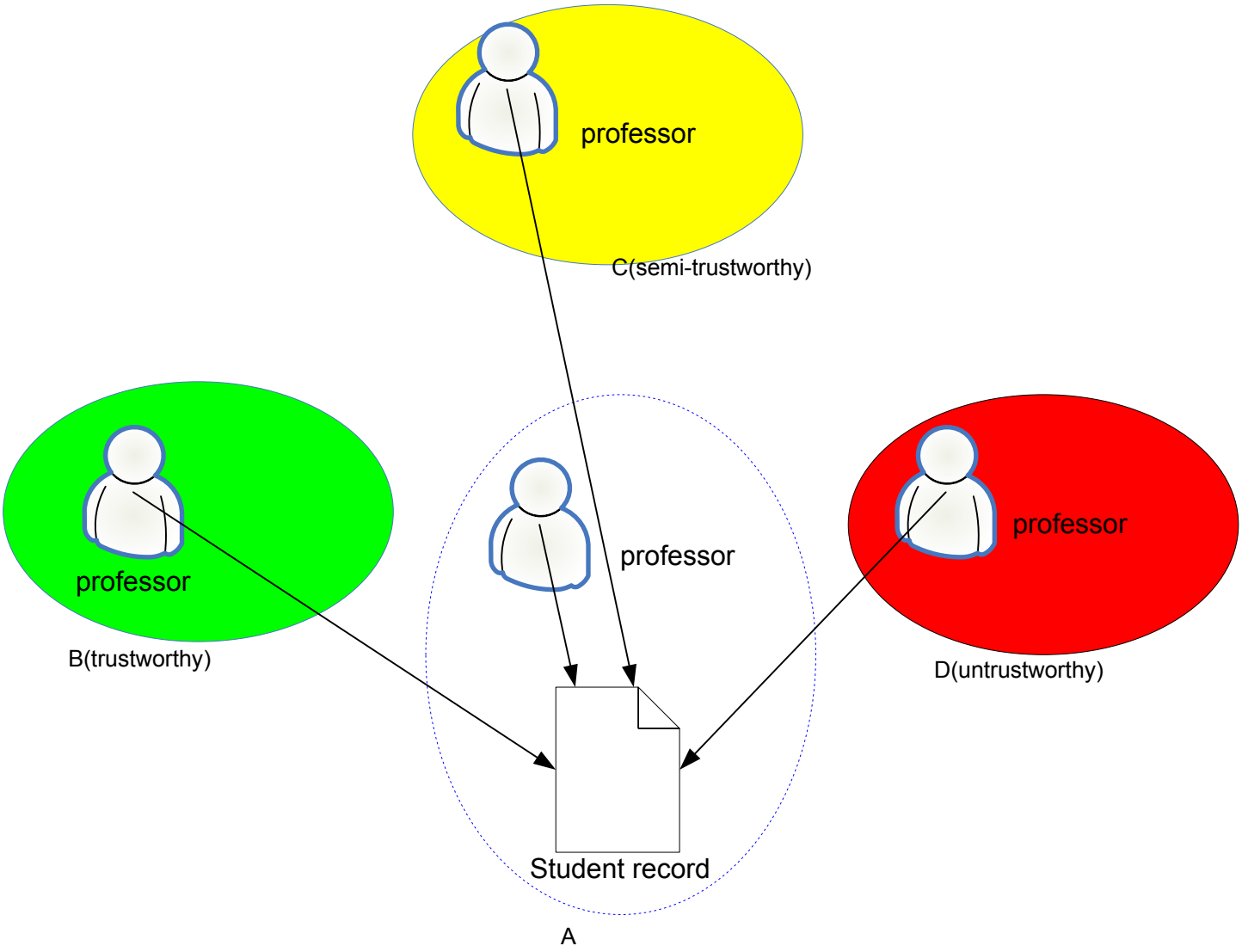


Figure 4. RBUC for Coalitions



REFERENCES

[ALKAS02] Mohammad Al-Kahtani and Ravi Sandhu, "A Model for Attribute-Based User-Role Assignment." *Proc. 17th Annual Computer Security Applications Conference*, Las Vegas, Nevada, December 9-13, 2002, pages 353-362.

[FERR01] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn and Ramaswamy Chandramouli. "Proposed NIST Standard for Role-Based Access Control." *ACM Transactions on Information and System Security*, Volume 4, Number 3, August 2001, pages 224-274.

[KAND02] Savith Kandala and Ravi Sandhu, "Secure Role-Based Workflow Models." Database Security XV: Status and Prospects, (D. Spooner, editor), Kluwer 2002.

[LIU05] Alexander Liu, Cheryl Martin, Tom Hetherington, and Sara Matzner, A Comparison of System Call Feature Representations for Insider Threat Detection, Proceedings of the 2005 IEEE Workshop on Information Assurance, United States Military Academy, West Point, NY June 2005

[PARK04] Jaehong Park and Ravi Sandhu. "The $UCON_{ABC}$ Usage Control Model." *ACM Transactions on Information and System Security*, Volume 7, Number 1, February 2004.

[PITK99] James Pitkow and Peter Pirolli. Mining longest repeating subsequences to predict World Wide Web surfing. In *Proc. of 2nd USENIX Symposium on Internet Technologies and Systems (USITS'99).*Boulder, Colorado, October 1999.

[SAND96] Ravi Sandhu, Edward Coyne, Hal Feinstein and Charles Youman, "Role-Based Access Control Models." *IEEE Computer*, Volume 29, Number 2, February 1996.

[THUR05] B. Thuraisingham, Security Standards for the Semantic Web, Computer Standards and Interface Journal, March 2005.

[THUR05d] B. Thuraisingham, *Managing Cyber Threats*: Issues and Challenges, Kluwer (editor: V. Kumar et al), Kluwer, 2005.

[THOM04] Roshan Thomas and Ravi Sandhu, "Towards a Multi-Dimensional Characterization of Dissemination Control." *Proc. 5th IEEE International Workshop on Policies for Distributed Systems and Networks*, New York, June 7-9, 2004, pages 197-200.

[ZHANG05] Xinwen Zhang and Ravi Sandhu, "Peer-to-Peer Access Control Architecture Using Trusted Computing Technology." Proc. 10th ACM Symposium on Access Control Models and Technologies (SACMAT), Stockholm, June 1-3, 2005.

Appendix B

Data and Applications Security Research at the University of Texas at Dallas

Bhavani Thuraisingham

(October 2004 – Present)

Research is proceeding in three main areas: Assured Information Sharing, Secure Geospatial data management, and Surveillance/Biometrics. This research is the work of my students under my supervision and direction.

Publications: Several journal publications including in IEEE Transactions on Systems, Man and Cybernetics, Very Large Database Journal, Computer Systems Science and Engineering, Multimedia Tools; and Conferences including ACM SACMAT, IEEE ISORC, and IFIP Data Security. Patent applications are under discussion. Book based on our research on Data Mining Applications is under contrct.

Area 1: Assured Information Sharing:

In the area of assured information sharing (funded mainly by AFOSR), the goal is for organizations to share data and at the same time enforce policies. We are investigating confidentiality, privacy, trust, integrity, provenance, standards and infrastructure aspects. In particular, we are examining three scenarios. In the first scenario we assume that the partners of a coalition are trustworthy (e.g, US, UK, Australia). However each partner may want to enforce various security policies. We are investigating the use of Ravi Sandhu's RBAC and UCON policies for such a scenario, carrying out data mining and conducting experimental studies as to the amount of information that is lost by enforcing policies. We are also investigating ways to transfer our technologies to programs such as DoD's NCES (Network Centric Enterprise Services)

In the second scenario, we assume that the partners are semi-trustworthy. In this case, we want to play games with the partners and extract as much information as possible without giving out information about ourselves. We are using results from game theory to formulate strategies for such a scenario and have obtained some interesting simulation results. In the third scenario we assume that the partners are untrustworthy. Here, we apply data mining to defend our systems from virus and worms and at the same time try to probe into our partners systems, We are examining the use of honey pot technologies and are conducting both defensive and offensive information operations.

In addition to the above areas of focus we are conducting complementary research funded by Texas Enterpriwse Funds including in privacy preserving data mining, trust management and negotiation, secure semantic web, data provenance management, real-time dependable data management, risk-based access control applying markov models, grid-based infrastructures, and secure ERP systems for coalition data sharing.

Area 1: Secure Geospatial Data Management:

In the area of secure geospatial data management (funded by Raytheon Corporation), we are developing technologies for geospatial semantic web and data mining. We are

specifying extensions to GML for access control policies as well as developing ontologies for geospatial data. Using these ontologies we are conducting data mining. In addition, we are also developing geospatial web services. Finally we are developing a new language called GRDF (Geospatial Resource Description Framework) and Secure GRDF for a geospatial semantic web. While we are developing various pieces of technologies, our goal is to work through standards organizations such as OGC (Open Geospatial Consortium) and corporations such as Raytheon to transfer our research to standards and operational programs. We are members of both OGC and USGIF. We also members of UTD's Geosciences program.

Area 3: Surveillance/Biometrics/Sensor/RFID etc.

In the area of surveillance and biometrics (in collaboration with Unisys Corporation) our goal is to develop technologies for detecting suspicious events as well as to maintain privacy. We developed a surveillance system to detect suspicious events. We identified normal events and used data mining techniques determined whether an event is suspicious. We are investigating the use of encryption techniques to ensure privacy of the individuals. Finally we developed access control models for surveillance data.

In the area of biometrics, we are developing technologies to support our surveillance work including novel algorithms for face recognition and fingerprint detection. In the areas of RFID, our goal is to manage the various RFID tags. Essentially we are designing a system that manages these tags that may migrate and ensures security and privacy. We are also investigating identity assurance and in particular federated identity management. I have some Masters level students conducting research and simulation studies in secure sensor networks.