Modeling and Evaluating Information Leakage Caused by Inferences in Supply Chains

Zhang, Da Yong

A Thesis
in
The Department
of
Concordia Institute for Information Systems Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Applied Science (Information Systems Security) at
Concordia University
Montreal, Quebec, Canada

December 2009

Your file Votre référence
ISBN: 978-0-494-67282-2
Our file Notre référence
ISBN: 978-0-494-67282-2

ABSTRACT

Modeling and Evaluating Information Leakage Caused by Inferences in Supply Chains

Zhang, Da Yong

While information sharing can benefit supply chains significantly, it may also have a "side effect", namely, information leakage. A limitation common to many existing solutions for preventing information leakage in supply chains is that they rely, either implicitly or explicitly, upon two unrealistic assumptions. First, what information is confidential is well known. Second, confidential information will not be revealed, if only it is not shared, regardless of how much other information is being shared. As it shall be shown in this thesis, those assumptions are not always true due to potential information leakage caused by inferences. Specifically, a conceptual model of such information leakage is proposed in this thesis. The model will enable companies in a supply chain to better understand how their confidential information may be leaked through inferences. On the basis of the proposed conceptual model, a quantitative approach is devised to evaluate the risk of information leakage caused by inferences when a given amount of information is shared. The quantitative approach will allow companies in a supply chain to measure and consequently to mitigate the risk of information leakage. Finally, a case study is discussed to illustrate how the proposed approaches work in practice.

Acknowledgments

I would like to thank all the people who ever helped and inspired me during my study at Concordia University.

Especially, I would like to thank my supervisors, Dr. Yong Zeng and Dr. Lingyu Wang. This thesis would not have been possible without their creative ideas, insightful comments and good advices. The support and encouragement of Dr. Zeng is invaluable to both my work and personal life. Dr. Wang's courses are my favorite at Concordia University, and discussions with him are always very helpful.

I am proud of being a member of the Design Lab at Concordia University, Dr. Wang's research group and the CRIAQ PLM2 project team. Working together with so many nice and talented people in these groups makes my life at Concordia University enjoyable and rewarding. Hongtao Li deserves special thanks for providing me materials used to extract the case study in this thesis. I also wish to thank anonymous reviewers for their comments on my papers submitted to conferences and journals.

My deepest gratitude goes to my parents, brother and sisters for their unconditional and constant love and support throughout my life, and for the encouragement they gave me when I was frustrated.

Table of Contents

List of Figures

List of Tables

Table of Acronyms

ACL	Area Control List
AFCCL	Automatic Firearms Country Control List
BIS	Bureau of Industry and Security
CCL	Commerce Control List
CGP	Control Goods Program
CPD	Collaborative Product Development
DAC	Discretionary Access Control
DDTC	Directorate of Defense Trade Controls
DPA	Defence Production Act
EAR	Export Administration Regulations
EBD	Environment Based Design
ECL	Export Control List
EIPA	Export and Import Permit Act
ICL	Import Control List
IP	Intellectual Property
ITAR	International Traffic in Arms Regulations
MAC	Mandatory Access Control
OFAC	Office of Foreign Assets Control
PDM	Product Data Management
PWGSC	Public Works and Government Services Canada

RBAC	Role Based Access Control
SCM	Supply Chain Management
SMC	Secure Multi-party Computation
USML	U.S. Munitions List
VPT	Virtual Project Team

Chapter 1

INTRODUCTION

1.1 Background

"A supply chain is the system of organizations, people, technology, activities, information and resources involved in moving a product or service from supplier to customer." [Wik] Supply Chain Management (SCM) can be defined as "a set of approaches utilized to efficiently integrate suppliers, manufacturers, warehouses, and stores, so that merchandise is produced and distributed at the right quantities, to the right locations, and at the right time, in order to minimize system wide costs while satisfying service level requirements" [SLKSL08].

Information sharing is widely accepted as an effective SCM approach that may significantly benefit supply chains [LST00, HLM03, Fia05, ZJ07], especially in reducing the bullwhip effect and decreasing supply chain costs [LPW97, MCdD07], increasing the efficiency of Collaborative Product Development (CPD) [ZSG04, LQ06], and facilitating decision makings such as product architecture, make-or-buy decisions [GSK06], supplier selection [MPHR98, dBLM01] and early supplier involvement [PHR05]. For this reason, it is common for a company in a supply chain to share a large amount of information with partners, which may then further share the information with third-parties.

On the other hand, information sharing in supply chains is also a double-edged sword:

it may have an adverse effect, namely, information leakage [LW00, Li02, Zha02, HT06a, HT06b, AG09]. In general, information leakage means that confidential information is unintentionally revealed to unauthorized parties. In supply chains, information leakage is a serious threat due to real incentives, that is, companies have strong motivations and more than enough capabilities to collect, analyze, acquire, and utilize information from others to gain a competitive edge [1].

1.2 Motivation

Information leakage in supply chains can take two different forms. First, confidential information may be mistakenly shared, resulting in the so-called *direct information leakage*. To avoid direct information leakage, companies need a precise answer to the question: *What information is confidential?* However, providing such an answer is usually more challenging than it looks, as we shall discuss in this thesis. Second, confidential information may also be unintentionally leaked in the form of *inferences*. An inference happens when confidential information can be inferred from other, seemingly non-confidential, shared information. This is possible due to the inherent engineering relationships between different pieces of information. To prevent damaging information leakage caused by inferences, companies need to answer the question: *What inferences are possible, and what is the risk of information leakage caused by such inferences?*

Unfortunately, answers to the above questions provided by most existing technical solutions are not fully satisfactory (the detailed review of existing solutions is given in Chapter 2). First, those solutions typically assume the classification of confidential information is already given, or can be trivially obtained. However, this is not always true. Different pieces of information in a complex engineering design, such as different parameters appearing in the same engineering formula, usually have an entangled relationship. Such a relationship

[1] Although information leakage may happen both inside a company and between different companies, we shall focus on the latter case in this thesis.

may blur the boundary between confidential and non-confidential information. Second, the possibility of potential information leakage caused by inferences is simply ignored in most existing technical solutions. However, the possibility and consequences of information leakage caused by inferences is not always ignorable, for example, in the defense industry or for information crucial to keeping advantages in competitive markets.

1.3 Contributions

In this thesis, information leakage caused by inferences in supply chains is systematically studied. The main contributions of this thesis include:

(1) A conceptual model is proposed for information leakage caused by inferences in supply chains. Instead of limiting our model to specific applications, the model mainly consists of abstract concepts, such as holder, inferrer, parameter, knowledge, channel, and so on. The model represents two types of generic knowledge, that is, the knowledge of parameters modeled as probability distributions and the knowledge of the relationship between parameters as a so-called logical dependency graph. The use of abstraction allows the model to be mapped to a broad range of real world scenarios. The model thus provides a powerful tool for companies in supply chains to better understand the nature of potential information leakage caused by inferences regardless of the underlying details of specific applications.

(2) On the basis of the conceptual model, a quantitative approach is devised to evaluate the risk of information leakage caused by inferences when a given amount of information is shared. The quantitative approach provides practical methods for companies in supply chains to take actions against potential information leakage caused by potential inferences, such as measuring and mitigating the risk of such information leakage.

(3) To illustrate the application of the proposed model and quantitative approach, a

software prototype is developed and an industrial case is studied.

1.4 Thesis organization

The rest of this thesis is organized as follows:

(1) Chapter 2 reviews related work based on an analysis of the security requirements in supply chains, using the design theory of Environment Based Design (EBD);

(2) Chapter 3 proposes a conceptual model of information leakage caused by inferences in supply chains;

(3) Chapter 4 devises a quantitative approach to evaluating the risk of information leakage caused by inferences in supply chains, and discusses the principles of mitigating the risk, especially by supplier selection;

(4) Chapter 5 presents a software prototype that implements the risk evaluation algorithm and a case study on a product in process industry;

(5) Chapter 6 concludes this thesis and points out future work.

Chapter 2

RELATED WORK

2.1 Security requirements in supply chains

2.1.1 Environment Based Design (EBD)

Environment Based Design (EBD) is a generic analysis and design methodology proposed by Zeng et al. in a series of papers [ZC91, ZG99a, ZG99b, Zen02, Zen04b, ZPA+04, Zen08, ZY09]. Figure 2.1 illustrates the process flow of EBD. It consists of three major steps: "environment analysis", "conflict identification" and "concept generation" [ZY09].

(1) "Environment analysis" defines the environment of the system with environment components and relationships between environment components;

(2) "Conflict identification" identifies conflicts between environment components or between their relationships;

(3) "Concept generation" generates new design concepts that resolve some conflicts in the environment of the system.

When new design concepts are generated and added to the environment, some conflicts will be solved. However, new design concepts may also cause new conflicts in the new environment of the system. EBD can be applied to analyze, identify and solve the new conflicts,

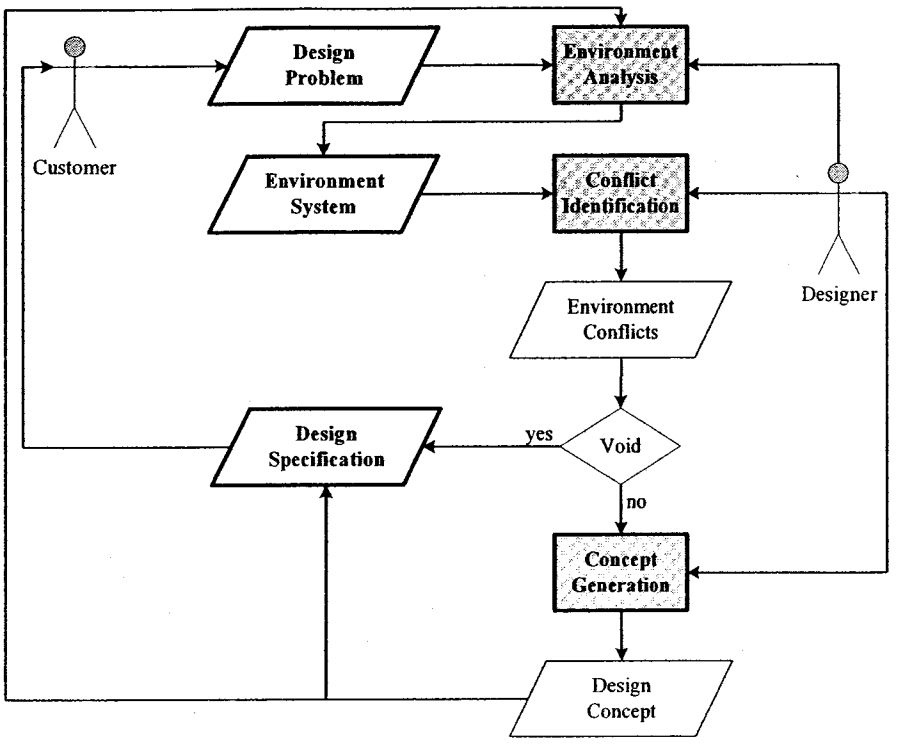

Figure 2.1: Environment Based Design: process flow [Zen04a]

following the steps of "environment analysis", "conflict identification" and "concept generation". Therefore, EBD is a recursive process. It will continue until there are no undesired conflicts [ZY09].

2.1.2 An EBD-based analysis

A supply chain, as a complex network, is difficult to analyze. Therefore, supply chains are usually studied from different structural perspectives, such as dyadic, serial, divergent, convergent and network [HLM03]. The dyadic structure is more suitable for being studied with analytical methods, and mathematical models of the dyadic structure can be extended to the divergent structure easily. In this section, a dyadic supply chain, which contains two supply chain partners, partner A and partner B, will be considered when analyzing the security requirements in supply chains. Logically, the analysis can be easily extended to multiple partners.

The security mechanisms that partner A is to deploy are considered as the system to design. According to EBD, the system consists of the structure of the security mechanisms, the environment of the security mechanisms and the interactions between the security mechanisms and its environment.

First, we identify objects and analyze their interactions within the environment of partner A's security mechanisms. According to EBD, environment objects can be generally divided into three classes, namely natural, built and human. The natural environment of partner A's security mechanisms includes objects such as time, space, etc.; the built environment of partner A's security mechanisms includes organizations, business processes, information systems, information, materials, parts, components, products, etc.; the human environment of partner A's security mechanisms consists of all workers who work for partner A or partner B. Some of the above environment objects are directly related to partner A's security mechanisms, such as business processes, workers, information systems and information.

(1) Business processes define what human objects and information systems are involved in a task and what information must be provided to execute the task. The execution of a business process requires accesses to information systems and information.

(2) Workers work in one or more business processes, access information systems and information, and may collaborate with each other;

(3) Information systems consist of software, hardware and networking. Information systems need to access information and communicate with other information systems;

(4) Information can be divided into two classes: shared information and non-shared information. As mentioned before, inferences may be conducted from shared information to non-shared information.

Based on the above discussion, some interactions directly related to partner A's security mechanisms are listed in Table 2.1.

Table 2.1: Interactions between environment objects

Environment Objects	Interaction
business process and business process	integration and collaboration
worker and worker	collaboration
worker and information system	access
information system and information system	communication
information system and information	access, processing and management
information and information	dependency

Second, we identify conflicts within the environment of partner A's security mechanisms. In this thesis, we focus on one of many conflicts within the environment of partner A's security mechanisms, the conflict between *"Protection"* and *"Access"* of A's non-shared information. In the dyadic supply chain, partner A shares a set of information with partner B, and tries to protect other information from accessing by partner B; partner B has accesses to the shared information, and tries to access the information protected by partner A.

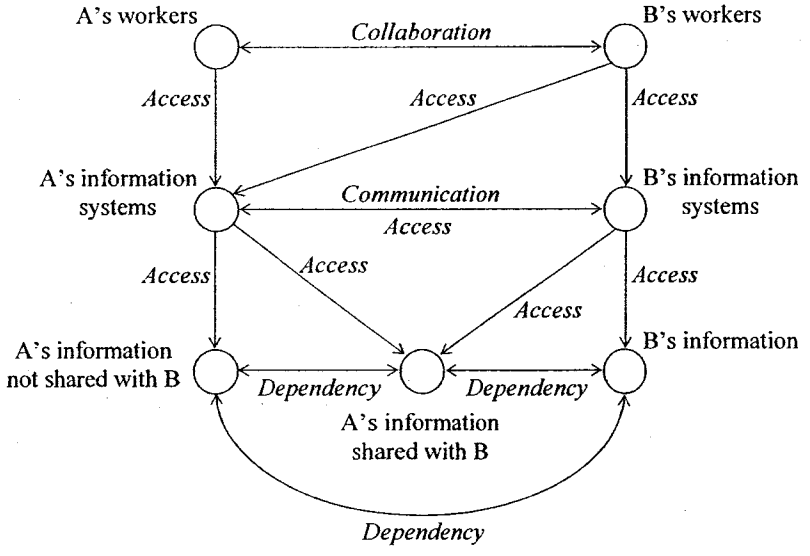

Figure 2.2: Routes to access partner A's information not shared with partner B

Workers, information systems and information may belong to partner A or partner B; and A's information can be divided into two classes: A's information shared with B and A's information not shared with B. Combined the above objects with the interactions in Table 2.1, as shown in Figure 2.2, there are at least four routes by which partner B can

access A's information that is not shared with partner B.

Route 1: *B's workers→A's information systems→A's information not shared with B*

B's workers may access A's information systems directly and access A's information not shared with B through A's information systems.

Route 2: *B's workers→B's information systems→A's information systems→A's information not shared with B*

B's workers may access A's information not shared with B through the communications between B's information systems and A's information systems. Another security issue for this route is communications security. The third parties may obtain accesses to unauthorized information by eavesdropping or intercepting the communications between B's information systems and A's information systems.

Route 3: *B's workers→A's workers→A's information systems→A's information not shared with B*

A's workers have accesses to A's information not shared with B. B's workers may obtain A's information not shared with B during collaborations with A's workers.

Route 4: *B's workers→B's information systems→B's information+A's information shared with B→A's information not shared with B*

Through this route, B's workers may obtain A's information not shared with B by inferences based on B's information and A's information shared with B, and the dependencies among B's information, A's information shared with B and A's information not shared with B. This route relies on the dependencies between information and information. The difference between this route and other routes is: it is not necessary for this route to collaborate with A's workers, communicate with A's information systems or physically access A's information not shared with B.

We consider the fundamental security requirements in supply chains, including confidentiality, integrity and availability, which are directly related to the environment objects

of information and information systems.

(1) ISO/IEC 27002 defines confidentiality as "ensuring that information is accessible only to those authorized to have access" [Cal06]. In general, encryption is the mechanism used to ensure confidentiality. Encryption methods can be roughly divided into two classes: symmetric-key cryptography and public-key cryptography. While symmetric-key cryptography uses the same key for both encryption and decryption, public-key cryptography uses a pair of keys, a public key and a private key. Some examples of popular symmetric-key algorithms include DES, triple-DES, AES, and RC4. Diffie-Hellman algorithm is the first public-key cryptographic algorithm, and RSA is another widely used public-key cryptographic algorithm. Digital signature is one branch of public-key cryptography used to ensure authenticity.

(2) Integrity is defined by ISO/IEC 27002 as "safeguarding the accuracy and completeness of information and processing methods" [Cal06]. In the context of communication, integrity is sometimes divided into two types: data integrity and origin integrity. Data integrity ensures that the data received are the same as the data sent; origin integrity ensures that the data are from the sender as it is claimed. Cryptographic hash functions (such as MD5 and SHA-1) and digital signature are two techniques that are most-commonly used to ensure integrity.

(3) ISO/IEC 27002 defines availability as the property of "ensuring that authorized users have access to information and associated assets when required" [Cal06]. In general, to ensure the availability of information and information systems in supply chains, accesses to information and information systems have to be restricted.

According to EBD, we will add the security mechanisms corresponding to the fundamental security requirements in supply chains to the environment, and then conduct "environment analysis" and "conflict identification" again on the new environment.

When the fundamental security mechanisms such as encryption, cryptographic hash

functions, digital signature and secure communication protocols are applied into the environment, the security properties of confidentiality, integrity, availability and communications security are enhanced. However, partner B still can access partner A's information not shared with partner B by the four routes discussed above.

Now we consider the interactions between the environment objects of workers and information systems. According to Route 1 and Route 2, partner B can access the information not shared with partner B through partner A's information systems or the communications between partner B's information systems and partner A's information systems. Therefore, partner A needs access control mechanisms to ensure that partner B can only access the information that it is authorized to access.

Generally speaking, access control ensures that only authorized users can access specific information and resources in specific contexts. Access control models can be categorized as two classes, discretionary and non-discretionary. Discretionary Access Control (DAC) is an access control policy, in which the owner of an object determines who is allowed to access the object and what operations are permitted to be performed on the object. Mandatory Access Control (MAC) and Role Based Access Control (RBAC) [SCFY96, FKS07] are non-discretionary access control models. In a MAC-based system, a user's read and write permissions on an object are determined by the system based on the user's clearance level and the object's sensitivity label; in a RBAC-based system, a user's permissions are determined by the system based on the user's roles.

There are some widely used security protocols, such as IPSec [KA98], SSL/TLS [MSS98], SSH [YL06], Kerberos [KN93] and PKI [HFPS99], that provide encryptions and authentications for security communications. Rouibah et al. report that in a collaborative product definition management system they developed, they deployed a security mechanism based upon CLAVISTM, a full-strength security framework, which includes state of the art standards such as PKI, TLS for online interaction and S/MIME for offline e-mail communication, as well as a certificate server, integrated with a LDAP server [ROA07]. It is proved that these mature techniques can be applied to supply chains with few changes.

Some laws and regulations, such as the International Traffic in Arms Regulations (ITAR) of U.S. and the Control Goods Program (CGP) of Canada [BG03, Che03, LL06], control accesses to defense articles, defense services and technical data related to defense articles.

With fundamental security and access controls implemented in the environment, partner B cannot access the information not shared with it through Route 1 and Route 2 any more. However, partner B still can access the information not shared with it through collaborations with partner A's workers (Route 3) and inferences (Route 4).

The security requirements for secure collaborations between workers and between supply chain partners include export control, Intellectual Property (IP) protection, information leakage prevention, and so on. To enhance these security requirements, not only technical methods but also legal, organizational and social methods have to been taken into account. Legal methods include export control laws and regulations, IP laws, business contracts, employment contracts and other relevant laws and regulations. Existing technical methods can be roughly divided into four classes, access control-based methods, sanitization-based methods, generalization-based methods and Secure Multi-party Computation (SMC)-based methods.

On the basis of above analysis, Table 2.2 lists some security requirements in supply chains, and corresponding security mechanisms.

This thesis will focus on one specific problem in secure collaborations between supply chain partners, namely information leakage caused by inferences in supply chains. Therefore, we will review work related to information leakage in supply chains in Section 2.2.1, and export controls and technical methods that may be used to prevent information leakage in supply chains in Section 2.2.2.

Table 2.2: Security requirements in supply chains

Layers	Environment objects	Security requirements	Security mechanisms
Secure collaborations between supply chain partners	Business processes	Export control, IP protection, Information leakage prevention	ITAR, CGP, IP laws, Business contracts, Security policies
Secure collaborations between workers	Workers	Export control, IP protection, Information leakage prevention	ITAR, CGP, Employment contracts, Security policies, Security procedures
Access control	Information, Information systems, Workers	Authentication, Authorization, Access control	ITAR, CGP, Security protocols, Access control models
Fundamental security	Information, Information systems	Confidentiality, Integrity, Availability, Communications security	Encryption, Cryptographic hash functions, Digital signature, Secure communication protocols

2.2 Information leakage in supply chains

2.2.1 Information leakage

In management science, there exists some work related to information leakage in supply chains and its effect on the supply chain's material flows and information flows. However, there is not yet a systematic research of information leakage caused by inferences in supply chains.

In a literature review to information sharing in supply chains, Lee and Whang [LW00] gave an example of information leakage in supply chains. In their example, a supplier supplied a critical part to two manufacturers who competed in the final product market; the supplier guaranteed that it would not leak information acquired from one manufacturer

to the other manufacturer. However, if a relation could be established between the actions that the supplier takes and shared information, the other manufacturer might infer the shared information to some extent by observing the behavior that the supplier reacted to the shared information.

As the first work on the effect of information leakage in supply chains, Li [Li02] examined the company's incentives to share information vertically in a divergent supply chain. In the supply chain, there were a upstream manufacturer and some downstream retailers. The retailers competed on the quantity of output and were endowed with private demand and cost information. Li's research showed that the leakage effect encouraged the retailers to share their cost information with the manufacturer while it discouraged them from sharing their demand information.

On the basis of Li's work, Zhang [Zha02] studied a divergent supply chain, in which two downstream retailers competed on either the quantity or the price of output. Zhang pointed out although no information would be voluntarily shared with the manufacturer, the retailers were willing to share information completely and get side payment for the information sharing when their information was statistically less accurate or they benefited more from the effect of information leakage.

Hoecht and Trott [HT06b] discussed one type of information leakage in outsourcing, in which consultants who worked with many clients might be influenced when working with a client and then use the best practice they acquired to other clients.

Anand and Goyal [AG09] researched the effect of demand information leakage on the material flows and information flows of a supply chain with horizontal competition. In their supplier chain model, there was horizontal competition between two downstream firms who had a common upstream supplier. While one firm was informed with demand information and the other firm was not, demand information may be leaked from the informed firm to the uninformed firm through their common upstream supplier. As a result of information leakage, the informed firm may try to block information flows within the supply chain to conceal its demand information. Finally, it will lead to operational losses because of material

flow distortion.

2.2.2 Information leakage prevention

2.2.2.1 Export controls

There are three main U.S. regulations that control exports: (1) the International Traffic in Arms Regulations (ITAR) [oSa]; (2) the Export Administration Regulations (EAR) [oCc]; and (3) the Office of Foreign Assets Control (OFAC).

The ITAR regulates the manufacture, temporary import and export of "defense articles", and the export of "defense services" and "technical data" appurtenant to defense articles and defense services [LL06]. Defense articles and defense services controlled by the ITAR are listed on the U.S. Munitions List (USML) [oSb]. The ITAR is administrated by the Directorate of Defense Trade Controls (DDTC) at the U.S. Department of State. According to ITAR, any exports and re-exports of defense article, defense service, or technical data must obtain the approvals from the DDTC. Moreover, registrations with the DDTC are required for both companies that engages in any munitions manufacturing or exporting activities and persons who engages in the manufacturing, brokering, importing, or exporting of defense articles or furnishing defense services. Licenses are also required for companies and persons who provide defense services or enter into technical assistance or manufacturing license agreements, even if no defense article or technical data is exported [LL06].

Canada has an exemption to ITAR, which allows unclassified articles, services and technical data to be exported from U.S. to Canada relatively freely [BG03, CN06]. The exemption was suspended by the US Government in 1999. After a series of legislative and regulatory amendments to Canada's export controls, a new Canadian ITAR exemption was implemented by the U.S government in 2001.

The EAR regulates the export and reexport of "dual-use" items that have both commercial and military or proliferation applications [oCc]. However, purely commercial items without an obvious military use are also subject to the EAR [oCc]. Items regulated by

EAR are listed on the Commerce Control List (CCL) [oCd]. The EAR is administrated by the Bureau of Industry and Security (BIS) at the U.S. Department of Commerce. A license from BIS may be required, depending on not only whether an item falls onto a category on the CCL but also the destination, the end-user and the end-use [oCc].

The OFAC at the US Department of the Treasury is responsible for administering and enforcing economic and trade sanctions against targeted foreign countries, terrorists, proliferation and others that threat to the national security, foreign policy or economy of the United States [Off].

The major export control laws and regulations of Canada contain the Export and Import Permit Act (EIPA) [oCb] and the CGP [WC].

The EIPA regulates the import, export and transfer of certain goods and technology based on four lists: the Area Control List (ACL), the Export Control List (ECL), the Import Control List (ICL) and the Automatic Firearms Country Control List (AFCCL) [oCb]. According to EIPA, permits are required for the export of all goods and technology to countries on the ACL, for the export of goods and technology on the ECL and for the import of goods and technology on the ICL [AC]. The export of automatic firearms to a country that is not on the AFCCL is prohibited by EIPA [BG03, oCb].

CGP is a program to strengthen Canada's defence trade controls through registration, prevention, deterrence and detection and to regulate the examination, possession or transfer of controlled goods and technology in Canada [WC]. It is authorized by the Defence Production Act (DPA) [oCa] and administered by the Public Works and Government Services Canada (PWGSC). According to CGP, any person who accesses controlled goods and technology within Canada must be registered, exempt from registration or excluded from registration [BG03].

2.2.2.2 Access control

Access control or authorization ensures that only authorized users can access specific information. To meet the security requirements of various systems, a large number of access

control models have been developed in the last four decades. Tolone et al. [TAPH05] conduct an excellent literature review of general access control models for collaborative systems. In this section, we will review only those access control models relevant to information sharing in supply chains.

Leong et al. [LYL03] proposed a mixed access control model for a workspace-oriented distributed Product Data Management (PDM) system. In their model, the proposed distributed PDM system is stratified into multiply workspaces, a security level is assigned to each workspace, users working in a workspace are granted with rights on product data based on the workspace's security level.

Role-based viewing is a new technique for collaborative 3D assembly design developed by Cera et al. [CKHR04, CBK+06] and Kim et al. [KCR+06]. It is achieved through the integration of multi-resolution geometry and RBAC model [SCFY96, FKS07]. In their model, geometric regions, features and constraints data of 3D assembly models are related with a set of roles. For a specific user, a 3D model is generated for viewing based on its roles.

As an access control model for collaborative design, S-RBDDAC [WABN06] combines RBAC and cryptographic methods to support RBAC with consideration of time, scheduling and value adding activity, policy delegation relation in a distributed context and fine-grained access control at dataset level. S-RBDDAC allows a collaborator to send a subset of the dataset it received to a third party with supply chain relationships. Permissions are granted to roles through key distribution and policy delegation. [WABN06]

Trust is also considered in some access control models. For example, Chen et. al. [CCC08] presented a trust evaluation method for Virtual Project Team (VPT). It can assist VPT members in determining whether resource holders have made appropriate decisions to share resources with other VPT members. Combining with access control, it can enable secure resource sharing, facilitate collaboration and enhance information transparency among members in a VPT [CCC08].

Access control-based methods cannot prevent information leakage caused by inferences

in supply chains because unauthorized information is not necessary for inferences.

2.2.2.3 Sanitization

Sanitization, or suppression, is a method of removing confidential information or sensitive data from documents, databases or other media so that they are able to be released. Sanitization is often used to reduce the document's classified level; suppression is used in the literature of privacy protection in database systems [Swe02].

In supply chains, sanitization can be applied to CAD data exchange between partners in collaborative product developments. CAD data contains not only data for the drawing but also design knowledge such as features, parametric data and geometric data. Design knowledge containing in CAD data is often very valuable and considered as a company's intellectual properties. Therefore, companies will sanitize their CAD data before exchanging them with their partners.

In particular, companies may produce their CAD data for components by heterogeneous CAD software packages in collaborative assembly design [SG01, SG02, CSF04, KWMN04]. To build the final assembly model of the product, companies will convert their CAD data in incompatible formats into neutral CAD data, such as in STEP format. From the perspective of preventing information leakage, such a conversion can be used as a sanitization method to remove design knowledge contained in native CAD data.

Sanitization may be an effective approach to mitigate the risk of information leakage caused by inferences in supply chains. However, there are still two questions to be answered: "*what information should be removed?*" and "*after sanitization, what is the risk of information leakage caused by potential inferences?*" In this thesis, we will address these issues.

2.2.2.4 Generalization

Generalization, which means "replacing (or recoding) a value with a less specific but semantically consistent value" [Swe02], is another widely used method for protecting privacy in database systems.

In supply chains, companies may have to share essential data for detail design in collaborative product development. Mun et al. [MHH09] proposed a skeleton model based method, which represents essential data such as design specifications in an intuitive and explicit manner while it does not reveal data related to intellectual property contained in CAD models. It can be considered as an application of generalization in the area of protecting information leakage in supply chains.

Similar to sanitization, generalization may be another effective approach to mitigating the risk of information leakage caused by inferences in supply chains. However, it also has two questions to answer: "*what information should be generalized?*" and "*after generalization, what is the risk of information leakage caused be potential inferences?*" In this thesis, we will address these issues and discuss a generalization-based mitigation approach.

2.2.2.5 Secure Multi-party Computation

SMC [Yao86, GMW87, LP02] protects confidential information by allowing users to perform joint computation on multiple datasets while not revealing information in these datasets.

Atallah et al. [AEDS03] introduced SMC into the area of preventing information leakage in supply chains. They proposed several SMC protocols for supply-chain interactions, such as capacity allocation under various policies, and bidding and auctions under both discriminatory and nondiscriminatory pricing. Their method is different from traditional information sharing. It enables supply-chain partners to cooperatively achieve desired system-wide goals without revealing any private information, even though the jointly-computed decisions require this information [AEDS03].

SMC-based methods address a different issue of information leakage from the one addressed in this thesis because information leakage caused by inferences in supply chains happens when information is shared between partners instead of when two partners perform joint computation.

Chapter 3

CONCEPTUAL MODELING

As discussed before, companies may infer confidential information from shared information due to the inherent engineering relationships between different pieces of information. In this chapter, we will model information leakage caused by inferences in supply chains with a conceptual model.

3.1 The information leakage model

Information leakage caused by inferences in supply chains is a complicated and challenging issue because of the complexity of supply chains and information sharing in supply chains.

First, a supply chain is a complex network, in which each partner plays one or more roles. While partners cooperate for some common interests, they have different business objectives and some of them are even (potential) competitors. Moreover, partners may have different security policies and deploy security mechanisms corresponding to their security policies, respectively. Therefore, when a supply chain partner shares information with another supply chain partner, the shared information may be leaked to third party companies by the second supply chain partner either deliberately or unintentionally.

Second, there are many kinds of information shared between partners in supply chains.

For examples, a manufacturer may provide technical know-how to its suppliers; companies may exchange CAD data in collaborative product development; a retailer may share its order information and demand forecast with the manufacturer. Furthermore, the relations between shared information and confidential information sometimes are complicated and may be described with formal methods such as algebra, logic and set theory, or with informal methods such as tables, graphes and even natural languages.

Third, every partner has its own confidential information and usually tries to prevent its confidential information from leaking to (potential) competitors. For examples, a supplier may own intellectual property rights on the components it supplies; a manufacturer may keep its cost information confidential; a retailer may protect its order information and demand forecast from other competitive retailers. At the same time, every partner may also collect, analyze, acquire and utilize information from its (potential) competitors.

We abstract a conceptual model from concrete cases of information leakage caused by inferences in supply chains. The conceptual model focuses on concepts and their relationships relevant to information leakage caused by inferences in supply chains, and we take it as the basis for our subsequent work. The benefits of such a conceptual model include:

(1) Since it is independent of implementation details, it is more suitable for theoretical, analytical or quantitative research of information leakage caused by inferences in supply chains;

(2) The results of such research can be applied into concrete cases of information leakage caused by inferences in supply chains.

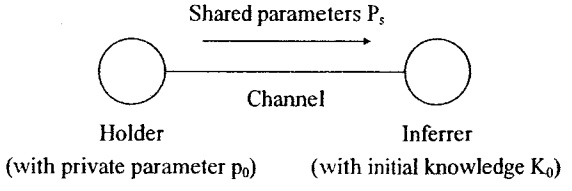

Figure 3.1: The conceptual model

Figure 3.1 shows the conceptual model of information leakage caused by inferences in supply chains, which consists of the following five key abstract concepts.

(1) Parameter: A parameter is an abstract information object that describes an attribute of a system. It may be a product design parameter or any other information object that can be described by a triplet (*name*, *actual value*, *working values*), in which *name* is an identifier of the parameter, *actual value* is the value that the parameter takes in the system and *working values* are the values that if the parameter takes, the system will still work well.

(2) Holder: The holder is the supply chain partner who holds the private parameter p_0 and tries to prevent it from revealing to other supply chain partners.

(3) Inferrer: The inferrer is a supply chain partner who tries to acquire a working value of the private parameter p_0 protected by the holder.

(4) Knowledge: Both the holder and the inferrer have their knowledge of parameters and the relations between parameters. We consider two types of knowledge in this thesis: knowledge of parameters and knowledge of the relations between parameters, and model them with probability distributions and logical dependency graph, respectively. We will discuss the two types of knowledge further in Section 3.2.

(5) Channel: The channel is the media between the holder and the inferrer, through which shared parameters P_s are transferred from the holder to the inferrer.

With the above conceptual model, the scenario of the information leakage can be derived as follows.

(1) The holder knows the actual value of a private parameter p_0. It tries to prevent the actual value of p_0 from revealing to the inferrer while it shares its knowledge of parameters P_s with the inferrer in some way.

(2) The inferrer does not know the actual value of p_0. It tries to acquire the actual value of p_0 by inferring on the basis of its initial knowledge K_0 and knowledge of P_s provided by the holder through parameter sharing.

(3) The holder models the inferrer's initial knowledge and knowledge obtained through inferences, and estimates and mitigates the risk of the information leakage.

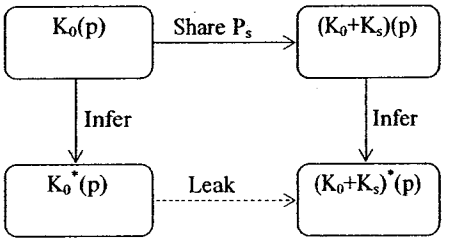

Figure 3.2: The information leakage model

The inferrer can obtain knowledge from three sources: initial knowledge, knowledge obtained through parameter sharing and knowledge obtained through inferences. If we denote the inferrer's initial knowledge as K_0, knowledge obtained through sharing parameters P_s as K_s, a comprehension of K_0 and K_s as $K_0 + K_s$, a comprehension of knowledge K and knowledge obtained through inferences on knowledge K as K^*, and knowledge of parameter p from knowledge K as $K(p)$, we can describe the relations among initial knowledge, knowledge obtained through parameter sharing and knowledge obtained through inferences as Figure 3.2.

(1) Initially, the inferrer has knowledge K_0;

(2) The inferrer's knowledge becomes K_0^* after it infers on K_0;

(3) When the inferrer obtains knowledge K_s through parameter sharing, it's knowledge becomes $K_0 + K_s$;

(4) The inferrer's knowledge becomes $(K_0 + K_s)^*$ after it infers on $K_0 + K_s$;

Definition 1 (Information leakage caused by inferences). *For a private parameter p, if* $(K_0 + K_s)^*$ *contains more knowledge of p than* K_0^* *does, we say information of p is leaked caused by inferences from the holder to the inferrer through sharing parameters* P_s.

3.2 Knowledge

3.2.1 Knowledge of parameters

As we discussed before, a parameter can be described by a triplet (*name*, *actual value*, *working values*). Because *name* is used only for the purpose of identifying the parameter, the holder and the inferrer may use different names for a parameter as long as the parameter is not shared. The *actual value* and the *working values* of a parameter are usually known to the holder; the *actual value* and the *working values* of the private parameter p_0 are unknown to the inferrer. In fact, the inferrer tries to acquire a *working value* of the private parameter p_0.

Even if it does not know the actual value of a parameter, the inferrer has knowledge of the parameter from out-of band channels, for example, its possible values and their probabilities. Therefore, the inferrer's knowledge of the parameter can be modeled as a probability distribution.

For example, there is a parameter p whose actual value is 0.8 and working values are within the range of $[0.7, 0.9]$. The inferrer has knowledge k_p of parameter p. Figure 3.3 shows some possible probability distributions as which k_p may be modeled. Figure 3.3(a) indiates that the inferrer is one hundred percent sure that the value of parameter p is 0.8; Figure 3.3(b) says that the inferrer knows there are three possible values and the most possible value is 0.8; with Figure 3.3(c), the inferrer knows that the value of parameter p is within the range of from 0.7 to 1.0 with a uniform probability distribution; with Figure 3.3(d), the inferrer knows that the possible values of parameter p follow the normal distribution

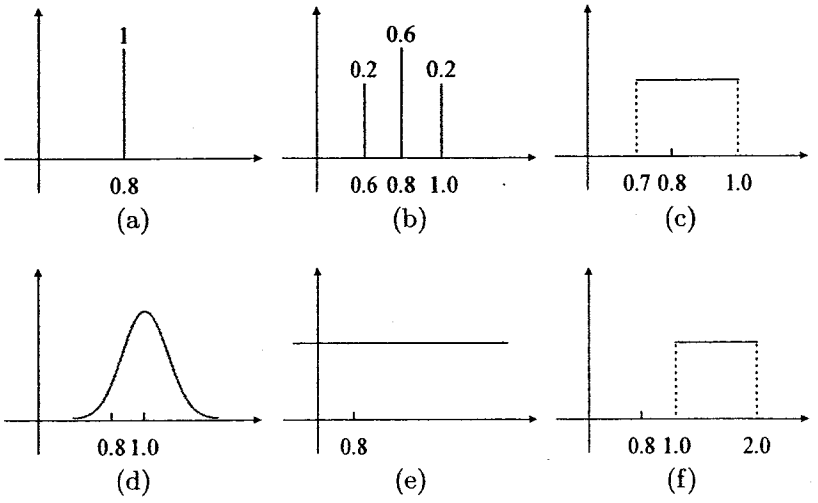

Figure 3.3: Examples of inferrer's knowledge of parameters

with mean 1.0; the inferrer may have "zero knowledge" of parameter p as shown by Figure 3.3(e); the inferrer may have totally wrong knowledge of parameter p, such as shown in Figure 3.3(f).

The probability distributions reflect the proximity of the inferrer's knowledge to the actual value. In probability distributions shown in Figure 3.3, it is easy to find that Figure 3.3(a) is the closest to the actual value; Figure 3.3(c) is more approximate to the actual value than Figure 3.3(e); Figure 3.3(f) contains the least information of parameter p.

In practice, it may be infeasible for the holder to collect the inferrer's knowledge of the parameter completely and accurately. Typically, the holder will describe the inferrer's knowledge of a parameter approximately with a simple or widely-used probability distribution, such as discrete distributions, continuous uniform distributions and normal distributions, following the physical characteristics of the parameter and the inferrer's information it collected.

3.2.2 Knowledge of relations among parameters

In this thesis, we introduce a graph-based method, *Logical Dependency Graph*, to describe the abstract logical relations between parameters. A logical dependency graph is different

from concrete dependencies between parameters described in algebra, logic or other formats; it is an abstract of concrete dependencies between parameters and it describes the dependencies between parameters in logic during the inference procedure. The holder needs a logical dependency graph to model what can be inferred.

Definition 2 (Logical dependency graph). *A logical dependency graph (or LDG) is a bipartite graph* [1] $G := (P, 2^P, LD)$, *where* P *is a set of parameters,* 2^P *is the power set of* P [2], $LD \subset P \times (2^P \backslash \{\emptyset\})$ *and* $\forall uv \in LD$, $u \notin v$.

According to Definition 2, a logical dependency graph G consists of parameters in P, parameter sets in 2^P and logical dependencies in LD, while every logical dependency is between a parameter from P and a parameter set from 2^P.

We take an example from the case study in Section 5.2 to show how to construct a logical dependency graph from a quantitative relation.

$$\begin{aligned} \boldsymbol{DryerOutletTemp} = \boldsymbol{DryerInletTemp} - (r_0 * TotalWaterRemovedPerCycle \\ /(60 * \rho_{ng} * RegenerationHeatingTime * (WaterContentOfRegenerationGas \\ * c_v/(\rho_{ng} * 1000000) + c_g)))/\boldsymbol{RegenerationFlowRate}, \end{aligned} \tag{3.1}$$

Equation 3.1 gives a quantitative dependency among some design parameters. *DryerOutletTemp* is a private design parameter; *DryerInletTemp* and *RegenerationFlowRate* are two design parameters that are unknown to the competitor (the inferrer in the case study); r_0, ρ_{ng}, c_g and c_v are physical constants from designer's handbooks; *TotalWaterRemovedPerCycle*, *RegenerationHeatingTime* and *WaterContentOfRegenerationGas* are operating conditions on the manual of the product.

[1] In graph theory, a bipartite graph $G = (U, V, E)$ is a graph that consists of two disjoint sets of vertices, U and V, and a set of edges E, in which every edge connects a vertex in U to one in V. [Weia]

[2] In set theory, given a set P, a power set of P is a set that consists of all sub sets of set P. Usually, the power set of set P is denoted as 2^P. [Weib]

First we construct the parameter set P of the logical dependency graph. In the inference procedure, what the inferrer concerns about are the parameters whose working values are unknown. Therefore, P usually does not contain public parameters and constants like $TotalWaterRemovedPerCycle$, $RegenerationHeatingTime$, $WaterContentOfRegenerationGas$, r_0, ρ_{ng}, c_g and c_v in Equation 3.1. For this example, $P = \{DryerOutletTemp, DryerInletTemp, RegenerationFlowRate\}$.

Then we construct the logical dependency relation LD between P and 2^P. Equation 3.1 indicates that if the actual values of $DryerInletTemp$ and $RegenerationFlowRate$ are known, the actual value of $DryerOutletTemp$ will be known. In other words, if parameter $DryerOutletTemp$ is the private parameter to be inferred, parameter $DryerInletTemp$ and parameter $RegenerationFlowRate$ need to be inferred first. Therefore, there is a logical dependency between parameter $DryerOutletTemp$ and parameter set $\{DryerInletTemp, RegenerationFlowRate\}$. Similarly, there are logical dependencies between parameter $DryerInletTemp$ and parameter set $\{DryerOutletTemp, RegenerationFlowRate\}$, and between parameter $RegenerationFlowRate$ and parameter set $\{DryerOutletTemp, DryerInletTemp\}$. For this example, $LP = \{(DryerOutletTemp, \{DryerInletTemp, RegenerationFlowRate\}), (DryerInletTemp, \{DryerOutletTemp, RegenerationFlowRate\}), (RegenerationFlowRate, \{DryerOutletTemp, DryerInletTemp\})\}$.

$$\boldsymbol{DryerOutletTemp} = \boldsymbol{CoolerInletTemp} + 10 \tag{3.2}$$

$$\boldsymbol{DryerInletTemp} = \boldsymbol{HeaterOutletTemp} * (1 - \boldsymbol{HeatLossRate}) \tag{3.3}$$

A logical dependency graph may describe logical dependencies extracted from multiple quantitative dependencies. Equation 3.2 and Equation 3.3 are two other quantitative dependencies from the case study in Section 5.2. We can construct the logical dependency graph with a procedure similar to the previous one. Figure 3.4 shows the logical dependency graph extracted from Equation 3.1, Equation 3.2 and Equation 3.3. In Figure 3.4 and the

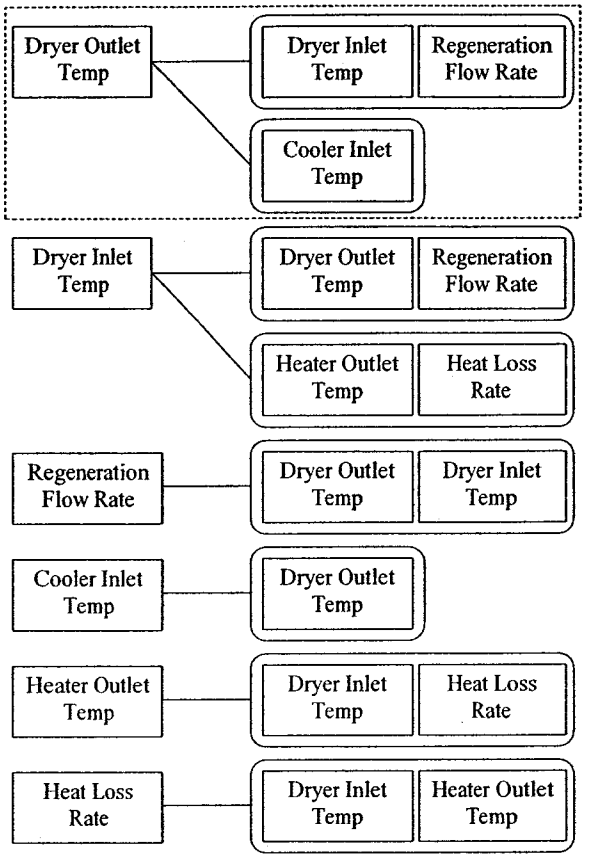

Figure 3.4: An example of logical dependency graph

rest of this thesis, we denote a parameter by a rectangle, a parameter set by a round corner rectangle and a logical dependency by a line connecting a rectangle and a round corner rectangle, and usually ignore isolated nodes in logical dependency graphes.

3.3 Inferences

3.3.1 Non-recursive inferences

In this section, we will model what knowledge can be obtained through inferences with given initial knowledge and knowledge obtained through parameter sharing.

First, we consider a case that includes four parameters and two logical dependencies as shown by the sub graph of the logical dependency graph in Figure 3.4 surrounded

by a dashed line. We denote the actual value of *DryerOutletTemp*, *DryerInletTemp*, *RegenerationFlowRate* and *CoolerInletTemp* as v_0, v_1, v_2 and v_3, respectively. The logical dependency between parameter *DryerOutletTemp* and parameter set {*DryerInletTemp*, *RegenerationFlowRate*} is an abstract of the quantitative dependency described by Equation 3.1. The logical dependency between parameter *DryerOutletTemp* and parameter set {*CoolerInletTemp*} is an abstract of the quantitative dependency described by Equation 3.2.

(1) If v_1, v_2 and v_3 are known, the value of *DryerOutletTemp* can be calculated by either Equation 3.1 or Equation 3.2. We denote them as v_0^1 and v_0^2, respectively. Then we have $v_0^1 = v_0^2 = v_0$.

(2) If v_1, v_2 and v_3 are unknown, the initial knowledge of parameter *DryerInletTemp*, *RegenerationFlowRate* and *CoolerInletTemp* can be modeled as probability distribution d_1, d_2 and d_3, respectively; and then the probability distribution of *DryerOutletTemp* can be calculated by either Equation 3.1 or Equation 3.2. We denote them as d_0^1 and d_0^2, respectively. If we assume that v_1 is within the range of d_1, v_2 is within the range of d_2 and v_3 is within the range of d_3, the ranges of d_0^1 and d_0^2 must intersect because there is at least one value, v_0, in their intersection.

We extend the above results to a general case and give Algorithm 1 for modeling what knowledge can be obtained through inferences.

The following gives a brief discussion of this algorithm.

(1) For parameter p_0, the inferrer's initial knowledge can be modeled as a probability distribution d_0^0. Algorithm 1 tries to exclude impossible values from the range of d_0^0 so that there is a higher possibility to acquire the actual value or a working value of parameter p_0.

(2) LD' is a set of logical dependencies in logical dependency graph G that are directly relevant to parameter p_0. If LD' is empty, it means there is no other parameter that

Algorithm 1 An algorithm for modeling what can be inferred

Input: A parameter p_0; a logical dependency graph $G = (P, 2^P, LD)$ and $p_0 \in P$; for $\forall p_i \in P$ and $0 \leq i \leq |P| - 1$, its initial probability distribution d_i^0;
Output: probability distribution d_0 of p_0;

1: $LD' \Leftarrow \{p_0 P_j | P_j \in 2^P, p_0 P_j \in LD \text{ and } j \geq 0\}$;
2: **if** LD' is empty **then**
3: $d_0 \Leftarrow d_0^0$;
4: **else**
5: **for all** $p_0 P_j \in LD'$ and $0 \leq j \leq |LD'| - 1$ **do**
6: **for all** $p_{jk} \in P_j$ and $0 \leq k \leq |P_j| - 1$ **do**
7: $d_{jk} \Leftarrow d_{jk}^0$; $\{d_{jk}^0$ is the initial probability distribution of $p_{jk}\}$
8: **end for**
9: $d_0^{j+1} \Leftarrow$ the probability distribution of p_0 inferred with $p_0 P_j$, d_{j0}, d_{j1}, ... and $d_{j(|P_j|-1)}$;
10: **end for**
11: **for all** $0 \leq l \leq |LD'|$ **do**
12: $r_0^l \Leftarrow$ the range of d_0^l;
13: **end for**
14: $r_0 \Leftarrow r_0^0 \cap r_0^1 \cap \cdots \cap r_0^{|LD'|}$;
15: **if** r_0 is empty **then**
16: $d_0 \Leftarrow d_0^0$;
17: **else**
18: **for all** $0 \leq m \leq |LD'|$ **do**
19: $d_0^{m'} \Leftarrow$ the part of d_0^m on r_0;
20: $d_0^{m''} \Leftarrow$ normalized $d_0^{m'}$;
21: **end for**
22: $d_0 \Leftarrow$ a comprehension of $d_0^{0''}$, $d_0^{1''}$, ..., $d_0^{|LD'|''}$;
23: **end if**
24: **end if**

has logical dependency with parameter p_0. In that case, the algorithm cannot do any further inference and it will return the initial probability distribution of parameter p_0, d_0^0.

(3) $p_0 P_j$ is a logical dependency between parameter p_0 and parameter set P_j. (i, j, k, l and m in Algorithm 1 are non-negative natural numbers, or i, j, k, l, $m \in N_0$.) Because there is such a logical dependency, a new probability distribution of p_0 can be inferred with the concrete dependency behind $p_0 P_j$ and probability distributions

of parameters in P_j. The inference at line 9 depends on the type of the concrete dependency behind $p_0 P_j$. For example, the inference at line 9 may be totally different for a dependency described in algebra and for a dependency described in logic.

(4) If the cardinality of LD' is $|LD'|$, there will have $|LD'|+1$ probability distributions of parameter p_0 after the block from line 5 to line 10. Each probability distribution has its range. At line 14, r_0 is assigned as the intersection of all of these ranges. r_0 is possible to be empty. As we discussed before, if r_0 is empty, there is at least one parameter whose actual value is not within the range of its initial probability distribution. If r_0 is empty, Algorithm 1 will return the initial probability distribution of parameter p_0, d_0^0.

(5) At line 19, every probability distribution of parameter p_0 is narrowed down to the new range r_0.

(6) At line 20, every probability distribution of parameter p_0 on the range r_0 is normalized. If $d_0^{m'}$ is discrete and its probability mass function is ${f_0}'(x)$, and the probability mass function of $d_0^{m''}$ is ${f_0}''(x)$,

$$\forall x \in r_0, {f_0}''(x) = \frac{{f_0}'(x)}{\sum_{y \in r_0} {f_0}'(y)}. \tag{3.4}$$

If $d_0^{m'}$ is continuous, its probability density function is ${f_0}'(x)$ and its range is between v_1 and v_2, and the probability density function of $d_0^{m''}$ is ${f_0}''(x)$,

$$\forall v_1 \leq x \leq v_2, {f_0}''(x) = \frac{{f_0}'(x)}{\int_{v_1}^{v_2} {f_0}'(y)dy}. \tag{3.5}$$

(7) At line 22, there will have $|LD'|+1$ normalized probability distributions of parameter p_0 on the range r_0. These probability distributions of parameter p_0 may be different because the initial knowledge of parameters may be not totally accurate. There are many strategies to comprehend these probability distributions of parameter p_0. One

strategy is using the arithmetic mean of their probability mass functions or probability density functions.

3.3.2 Recursive inferences

Algorithm 1 does not take full advantage of the logical dependency graph. For example, suppose to infer the probability distribution of parameter *DryerOutletTemp* with Algorithm 1 and the logical dependency graph is as shown in Figure 3.4. According to Algorithm 1, only logical dependencies between *DryerOutletTemp* and {*DryerInletTemp*, *RegenerationFlowRate*} and between *DryerOutletTemp* and {*CoolerInletTemp*}, and the initial probability distributions of parameter *DryerOutletTemperature*, *DryerInletTemp*, *RegenerationFlowRate* and *CoolerInletTemp* will be used. In fact, the probability distribution of parameter *DryerInletTemp* could also be inferred with its logical dependency with {*HeaterOutletTemp*, *HeatLossRate*} and the initial probability distributions of parameter *DryerInletTemp*, *HeaterOutletTempe* and *HeatLossRate*.

From the example above, it is easy to be found that a recursive algorithm for inferring the probability distribution of a parameter will be more effective. However, the recursive algorithm is a little more complicated than just changing line 7 of Algorithm 1 to "$d_{jk} \Leftarrow$ the probability distribution of parameter p_{jk} inferred recursively;" because of the existence of redundant recursions and infinite loops.

We use a tree-like structure to describe the recursions. Informally, first we add parameter p_0 as the root node of the tree-like structure; for each parameter or parameter-in-parameter-set p_i in the tree-like structure, if there is a logical dependency p_iP_j in the logical dependency graph, add parameter set P_j as a child node of p_i; and then continue the previous step. Figure 3.5 is an example of such a tree-like structure, which is constructed with logical dependencies in Figure 3.4 and gives partial recursions of inferring the probability distribution of parameter *DryerOutletTemp* with a recursive algorithm.

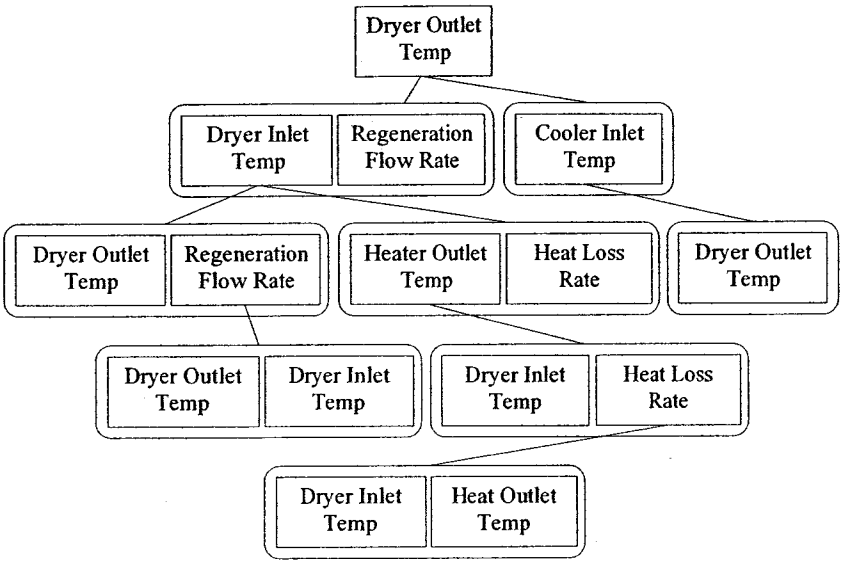

Figure 3.5: An example of the tree-like structure constructed with logical dependencies in Figure 3.4

A parameter may appear more than once in a tree-like structure. For example, parameter *DryerOutletTemp* appears four times and parameter *RegenerationFlowRate* appear two times in Figure 3.5. The multiple occurrences of a parameter during recursions may cause redundant recursions and infinite loops. For example, if the recursive algorithm infers the probability distribution of parameter *RegenerationFlowRate* with its logical dependency with {*DryerOutletTemp*, *DryerInletTemp*} at both its occurrences in Figure 3.5; one of the two inferences will be unnecessary.

Figure 3.6 gives a possible infinite loop when a recursive algorithm infers the probability distribution of parameter *DryerOutletTemp* with logical dependencies in Figure 3.4. Since there is a logical dependency between *DryerOutletTemp* and {*DryerInletTemp*, *RegenerationFlowRate*}, {*DryerInletTemp*, *RegenerationFlowRate*} could be added as a child node of *DryerOutletTemp*; since there is a logical dependency between *DryerInletTemp* and {*DryerOutletTemp*, *RegenerationFlowRate*}, {*DryerOutletTemp*, *RegenerationFlowRate*} could be added as a child node of the *DryerInletTemp*. Now we get the second occurrence of parameter *DryerOutletTemp*. If we repeat the previous procedure on the second occurrence of parameter *DryerOutletTemp*, we will get the third occurrence

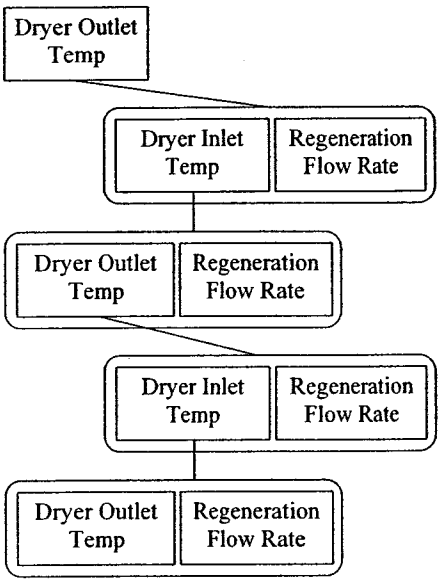

Figure 3.6: An example of infinite loop

of parameter *DryerOutletTemp* and we can repeat the previous procedure on the third occurrence of parameter *DryerOutletTemp* again. Apparently, it will not terminate.

Algorithm 2 is a recursive algorithm for modeling what knowledge of a parameter can be obtained through inferences, which implements recursive inferences and a strategy to avoid infinite loops and reduce redundant recursions. The strategy, we call it "one-time logical dependency", uses each logical dependency in the logical dependency graph only one time.

Proposition 1. *Algorithm 2 will always terminate.*

Proof. If we label the condition in line 2 as A and the condition in line 4 as B, the stopping criterion for the recursive algorithm will be $\neg A \vee (A \wedge \neg B)$. The stopping criterion will be reached. To prove this, we only need to prove that $A \wedge B$ is not always true. First, we assume A is always true. If $A \wedge B$ is true, line 6 will be executed and at least one logical dependency in LD' will be marked as "*used*". Because there are limited number of logical dependencies in G, there will be one point during the recursions that "LD' is empty" and $A \wedge B$ is false. Second, if A is not always true, $A \wedge B$ is not always true. Therefore, Proposition 1 is true. □

Algorithm 2 $rInfer(p_i, G, d_0, d_1, \ldots, d_{|P|-1})$: a recursive algorithm for modeling what can be inferred of parameter p_i

Input: A parameter p_i; a logical dependency graph $G = (P, 2^P, LD)$ and $p_i \in P$; for $\forall p_j \in P$ and $0 \leq j \leq |P| - 1$, its probability distribution d_j;
Output: probability distribution d_i of p_i;

1: $r_i \Leftarrow$ the range of d_i;
2: **if** there are more than one values in r_i **then**
3: $LD' \Leftarrow \{p_i P_k | P_k \in 2^P, p_i P_k \in LD, k \geq 0$ and $p_i P_k$ is marked with "*unused*"$\}$;
4: **if** LD' is not empty **then**
5: **for all** $p_i P_l \in LD'$ and $0 \leq l \leq |LD'| - 1$ **do**
6: mark $p_i P_l$ as "*used*";
7: **for all** $p_{lm} \in P_l$ and $0 \leq m \leq |P_l| - 1$ **do**
8: $d_{lm} \Leftarrow rInfer(p_{lm}, G, d_0, d_1, \ldots, d_{|P|-1})$;
9: **end for**
10: $d_i^l \Leftarrow$ the probability distribution of p_i inferred with $p_i P_l$, d_{l0}, d_{l1}, ... and $d_{l(|P_l|-1)}$;
11: **end for**
12: **for all** $0 \leq n \leq |LD'| - 1$ **do**
13: $r_i^n \Leftarrow$ the range of d_i^n;
14: **end for**
15: $r_i \Leftarrow r_i \cap r_i^0 \cap \cdots \cap r_i^{|LD'|-1}$;
16: **if** r_i is not empty **then**
17: $d_i \Leftarrow$ the part of d_i on r_i;
18: $d_i \Leftarrow$ normalized d_i;
19: **for all** $0 \leq h \leq |LD'| - 1$ **do**
20: $d_i^h \Leftarrow$ the part of d_i^h on r_i;
21: $d_i^h \Leftarrow$ normalized d_i^h;
22: **end for**
23: $d_i \Leftarrow$ a comprehension of d_i, d_i^0, ..., $d_i^{|LD'|-1}$;
24: **end if**
25: **end if**
26: **end if**

The following explains other aspects of Algorithm 2.

(1) Algorithm 2 invokes itself recursively at line 8. For the convenience of describing the invocation, we denote Algorithm 2 as $rInfer(p_i, G, d_0, d_1, \ldots, d_{|P|-1})$, where $rInfer$ is its name and p_i, G, d_0, d_1, ... and $d_{|P|-1}$ are its inputs.

(2) The inputs and output of Algorithm 2 are similar to Algorithm 1. The only difference

is that each logical dependency in G may be at one of two states, "unused" and "used". When Algorithm 2 is first invoked, the states of all logical dependencies in G are "unused".

(3) The states of logical dependencies in G are changed at line 6 and the probability distribution of p_j, d_j, is changed at line 23. During the recursions, the latest updated G and d_j are used when Algorithm 2 is invoked recursively at line 8.

When applying Algorithm 2 in inferring the probability distribution of parameter *Dryer-OutletTemp* with logical dependencies in Figure 3.4, the recursions are the same as shown in Figure 3.5.

Chapter 4

EVALUATION AND MITIGATION

In Chapter 3, we have modeled the inferrer's initial knowledge and its knowledge obtained from inferences from the perspective of the holder. What the holder concerns is what risk of information leakage it will take if the inferrer obtains such knowledge through inferences. In this chapter, we will discuss how the holder may evaluate and mitigate the risk of information leakage caused by inferences.

4.1 Evaluation

Definition 3 (Probability of information leakage: $P(p)$). *For a parameter p, its probability of information leakage $P(p)$ is defined as the probability that the inferrer acquires a working value of parameter p.*

Assume that the actual value of parameter p is v_0, its working values are within the range of $[v_1, v_2]$, and the inferrer obtains probability distribution d_p of parameter p by

inferences. If d_p is discrete, its range is r_p and its probability mass function is $f_p(x)$,

$$P(p) = \sum_{v_i \in r_p \text{ and } v_1 \leq v_i \leq v_2} f_p(v_i) \; / \; \sum_{v_j \in r_p} f_p(v_j); \tag{4.1}$$

If d_p is continuous, its range is $[v_3, v_4]$, and its probability density function is $f_p(x)$,

$$P(p) = \int_{max\{v_1,v_3\}}^{min\{v_2,v_4\}} f_p(x)dx \; / \; \int_{v_3}^{v_4} f_p(x)dx. \tag{4.2}$$

Definition 4 (Conditional probability of information leakage: $P(p|\{(p_j, d_{p_j})\})$). *For a parameter p and a set of parameter-probability distribution pairs $\{(p_j, d_{p_j})\}$, conditional probability of information leakage $P(p|\{(p_j, d_{p_j})\})$ is defined as the probability that the inferrer acquires a working value of parameter p when $\forall p_j \in P_s$, p_j is shared with the inferrer as d_{p_j}.*

Algorithm 3 An algorithm for calculating conditional probability of information leakage $P(p|\{(p_j, d_{p_j})\})$

Input: A parameter p and its working values $[v_1, v_2]$; a logical dependency graph $G = (P, 2^P, LD)$ and $p \in P$; for $\forall p_i \in P$ and $0 \leq i \leq |P|-1$, its initial probability distribution d_{p_i}; $\forall p_j \in P_s$ and $0 \leq j \leq |P_s| - 1$, its probability distribution d_{p_j};
Output: conditional probability of information leakage $P(p|\{(p_j, d_{p_j})\})$

1: **for all** $p_j \in P_s$ **do**
2: **for all** $p_i \in P$ **do**
3: **if** (p_j is equal to p_i) **then**
4: $d_{p_i} \Leftarrow$ a comprehension of d_{p_i} and d_{p_j};
5: **end if**
6: **end for**
7: **end for**
8: $d_p \Leftarrow rInfer(p, G, d_{p_0}, d_{p_1}, \ldots, d_{p_{|P|-1}})$;
9: $P(p|\{(p_j, d_{p_j})\}) \Leftarrow$ calculate $P(p)$ with d_p by Equation 4.1 and Equation 4.2;

According to the conceptual model, the holder holds a private parameter p, whose actual value is v_0 and working values are within the range of $[v_1, v_2]$; the inferrer's initial knowledge k_0 can be modeled as a logical dependency graph $G = (P, 2^P, LD)$ and a set of parameter-probability distribution pairs $\{(p_i, d_{p_i})|\forall p_i \in P, d_{p_i}$ is a probability distribution

of p_i}; a set of parameters $P_s \subseteq P$ are shared with the inferrer, which can also be modeled as a set of parameter-probability distribution pairs $\{(p_j, d_{p_j}) | \forall p_j \in P_s, d_{p_j}$ is a probability distribution of $p_j\}$; the inferrer tries to acquire a *working value* of the private parameter p. Algorithm 3 gives an algorithm for calculating conditional probability of information leakage $P(p|\{(p_j, d_{p_j})\})$.

(1) Because $P_s \subseteq P$, for $p_j \in P_s$, there will be two probability distributions, d_{p_i} from initial knowledge and d_{p_j} from sharing P_s. At line 4, d_{p_i} is assigned with a comprehension of the two probability distributions. There may be many strategies to do such a comprehension. A natural and simple strategy is "trusting the latest information only", which means to ignore d_{p_i} and use d_{p_j} in the inference.

(2) At line 8, the probability distribution d_p of parameter p is inferred with Algorithm 2.

(3) At line 9, conditional probability of information leakage $P(p|\{(p_j, d_{p_j})\})$ is calculated with d_p and $[v_1, v_2]$ by Equation 4.1 and Equation 4.2.

Apparently, $P(p|\{(p_j, d_{p_j})\})$ is a real number within the range of [0,1]. If $P(p|\{(p_j, d_{p_j})\})$ is close to 1, it means that the inferrer may acquire a working value of parameter p very easily by inferences; if $P(p|\{(p_j, d_{p_j})\})$ is close to 0, it means that the inferrer may not acquire a working value of parameter p by inferences. We define a threshold t_0. If $P(p|\{(p_j, d_{p_j})\}) \geq t_0$, where t_0 is the threshold, we denote it as $\{(p_j, d_{p_j})\} \rightarrow_{t_0} p$; otherwise, we denote it as $\{(p_j, d_{p_j})\} \nrightarrow_{t_0} p$.

We consider a special but usual case of sharing: all parameters in P_s are shared with the inferrer as their actual values. In this case, we simplify $P(p|\{(p_j, d_{p_j})\})$ as $P(p|P_s)$, $\{(p_j, d_{p_j})\} \rightarrow_{t_0} p$ as $P_s \rightarrow_{t_0} p$. We define a threshold t_0. If $P(p|P_s) \geq t_0$, where t_0 is the threshold, we denote it as $P_s \rightarrow_{t_0} p$; otherwise, we denote it as $P_s \nrightarrow_{t_0} p$.

Definition 5 (Privacy context). *Let p be a private parameter and P_s be a set of parameters that the holder shares their actual values with the holder. If $P_s \rightarrow_{t_0} p$, we call P_s a privacy context to p.*

If $P_s \rightarrow p$ and $P_s \subset P_1$, it is certain that $P_1 \rightarrow p$ is true; if $P_s \rightarrow p$ and $P_2 \subset P_s$, it is not certain whether $P_2 \rightarrow p$ or $P_2 \nrightarrow p$ is true.

Definition 6 (Minimum privacy context). *If p is a private parameter and $\exists P_m, P_m \rightarrow p$, but $\forall P'_m \subset P_m, P'_m \nrightarrow p$, we say P_m is a minimum privacy context to parameter p.*

Based on discussions above, we can derive some obvious corollaries to answer the question of "*what information is confidential?*".

(1) If p is a private parameter, p should be confidential;

(2) If p is a private parameter, each of its privacy contexts should be confidential as a whole;

(3) If p is a private parameter, P_s is a privacy context to p and p_s is the only parameter in P_s, p_s should be confidential;

(4) If p is a private parameter and P_m is a minimal privacy context to p, at least one parameter in P_m should be confidential;

Definition 7 (Information Leakage Risk (ILR): $R(p|\{(p_j, d_{p_j})\})$). *For a parameter p and a set of parameters P_s shared with the inferrer, if $\forall p_j \in P_s$, p_j is shared with the inferrer as d_{p_j}, Information Leakage Risk (ILR): $R(p|\{(p_j, d_{p_j})$ is defined as*

$$R(p|\{(p_j, d_{p_j})\}) = P(p|\{(p_j, d_{p_j})\}) \times C(p), \tag{4.3}$$

where $C(p)$ is the consequence if parameter p is leaked to the inferrer.

There are many ways to define a consequence function of information leakage in supply chains. In our implementation, we define $C(p)$ as the following.

$$C(p) = \begin{cases} 1 & \text{if } p \text{ is confidential} \\ 0 & \text{if } p \text{ is not confidential} \end{cases} \tag{4.4}$$

The holder may define a threshold t_1. When $R(p|\{(p_j, d_{p_j})\})$ is greater than or equal to t_1, the sharing is considered "unsafe" or "too much"; when $P(p|\{(p_j, d_{p_j})\})$ is less than t_1, the sharing is considered to be "safe" or "not too much". This gives an answer to the question of "*what is the risk of information leakage caused by inferences?*".

4.2 Discussion on mitigation

According to Definition 7 and Equation 4.3, the risk of information leakage $R(p|\{(p_j, d_{p_j})\})$ is equal to the multiplication of the probability of information leakage $P(p|\{(p_j, d_{p_j})\})$ and the consequence of information leakage $C(p)$. Therefore, the risk of information leakage caused by inferences may be mitigated by reducing the probability and/or the consequence of information leakage. In this thesis, we will discuss in principle three approaches to decreasing the probability of information leakage on the basis of the results in Section 4.1 and point out their limitations for future improvements.

The first approach is generalization. Suppose that p is a private parameter, P_s is a set of parameters shared with the inferrer, and $P_s \rightarrow_{t_0} p$. If a parameter $p_i \in P_s$ is shared with the inferrer as a probability distribution d_{p_i} instead of its actual value and there are more than one value with the range of d_{p_i}, the new probability of information leakage caused by inferences may be lower than $P(p|P_s)$.

Table 4.1: Multiple settings of the blower

BlowerPower	*Delta P*	*BlowerPower*	*Delta P*
1.5	4	2.1	4
1.5	6	2.1	6
1.5	8	2.1	8
1.5	10	2.1	10
1.5	12	2.1	12

In some cases, parameters may be shared as probability distributions. For example, the blower is a component of the natural gas dryer discussed as a case study in Section 5.2. *BlowerPower* and *Delta P*) are two design parameters of the blower. The manufacturer

of the natural gas dryer may order the blower that can work under multiple settings as shown by Table 4.1 from a supplier. For the supplier, parameter *BlowerPower* may have two possible values, 1.5 and 2.1; parameter *DeltaP* may have five possible values, 4, 6, 8, 10 and 12.

The approach of generalization has a limitation: it works only if the parameter can be shared as a probability distribution.

The second approach is based on minimum privacy contexts. Suppose p_0 is a private parameter, P_s is a set of parameters and $P_s \rightarrow_{t_0} p_0$. If there is such a parameter set P_s', $P_s' \subset P_s$ but $\forall P_i \in P_s'$, $P_i \nrightarrow p_0$, the holder may mitigate the risk of information leakage by sharing only parameters in P_s' as their actual values with the inferrer. In theory, the holder can utilize minimum privacy contexts to find such a parameter set P_s'. If P_j is a minimum privacy context to p_0; then, we can obtain P_s' by $P_s' = P_j \backslash \{p_k\}$, where $p_k \in P_j$.

The minimum privacy context-based approach also has its limitations: first, it needs an efficient algorithm to find minimum privacy contexts; second, in practice, what parameters can be shared with a partner in a supply chain depends on many factors, such as business objects, business processes, product structure, the partner's capability etc. It is infeasible to simply divide a parameter set P_s into two parts, sharing one part but not the other.

Unlike the minimum privacy context-based approach, the approach of decompositions and allocations considers mitigating the risks of information leakage in product designs and partner choices in supply chains. Usually a product can be decomposed into systems; a system can be decomposed into components; and some components can be decomposed further into sub components. Product design parameters are used to specified or defined products, systems and components; isolated product design parameters are often meaningless without the products, systems or components that they specified or defined. If a partner is involved in a specific task or operation on a system or a component, it has to be shared with a certain set of design parameters of the system or the component. Therefore, to protect confidential product design parameters of a product, the manufacturer will try to find optimal product decompositions and allocations from tasks or operations on systems

or components to partners that makes the risks of information leakage lower than some threshold values.

The approach of decompositions and allocations are more suitable for protecting product design parameters than other parameters such as costs and inventories; and it is more suitable for the cases that the manufacturer holds confidential information because it is usually the manufacturer who has the initiative in product architecture and partner selection in a supply chain.

In next section, we will discuss mitigating the risk of information leakage with supplier selection. Mitigation with supplier selection can be regarded as an example of the decomposition and allocation approach.

4.3 Mitigation with supplier selection

4.3.1 Partitions

In this thesis, we use extended product structure tree to describe the relations among a product, its parts and components, and relevant tasks. There are two classes of nodes in an extended product structure tree, component nodes and assembly task nodes. A component node represents a product, a part or a component; an assembly task node, which is introduced into product structure tree for the purpose of simplifying issues relevant to assembly activities, represents the task of assembling its parent component. An edge connecting two component nodes represents a parent-child relationship between them. A component consists of its all child components. An edge connecting a component node and an assembly task node indicates that the component is assembled by the assembly task.

According to [ZG99b], a node can be defined as $n(k, i_k, j_{k-1})$, if the node is at the i_k-th position in the k-th layer and its parent node is at the j_{k-1}-th position in the $(k-1)$-th layer; all nodes construct a product structure tree recursively. Figure 4.1 shows a basic block of product structure tree [ZG99b]. Figure 4.1 shows an extended product structure

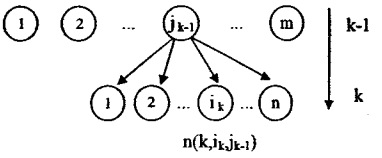

Figure 4.1: A basic block of product structure tree [Source: Adapted from [ZG99b]]

tree, which describes the relations among major components and assembly tasks of the product in the case study in Section 5.2. Table 4.2 lists all components and assembly tasks in the extended product structure tree.

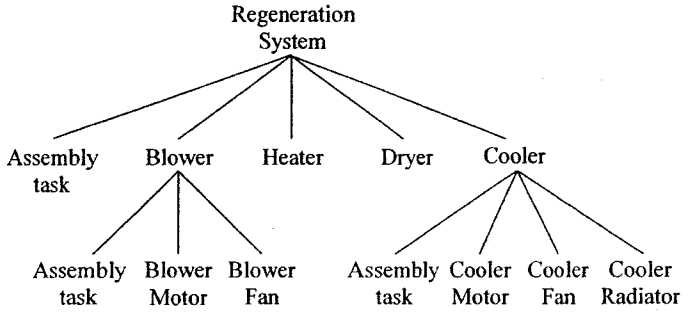

Figure 4.2: An extended product structure tree

Table 4.2: Components in Figure 4.2

Component	Description
n_0	Regeneration system
n_1	Assembly task of the regeneration system
n_2	Blower
n_3	Heater
n_4	Dryer
n_5	Cooler
n_6	Assembly task of the blower
n_7	Blower motor
n_8	Blower fan
n_9	Assembly task of the cooler
n_{10}	Cooler motor
n_{11}	Cooler fan
n_{12}	Cooler radiator

In this thesis, an extended product structure tree is denoted as T; all nodes of an extended product structure tree T is denoted as N_T; the root node of an extended product

structure tree T is denoted as $r(T)$. We define two functions $N(n)$ and $LN(n)$.

(1) $N(n) : N_T \rightarrow 2^{N_T}$, $\forall n \in N_T$, $N(n) = N_{T'}$, *where* $T' \subset T$ *and* $r(T') = n$;

(2) $LN(n) : N_T \rightarrow 2^{N_T}$, $\forall n \in N_T$, $LN(n) = \{n' \mid n' \in N(n)$ and n' is a leaf node$\}$.

Definition 8 (Partition). *T is a product structure tree and $r(T) = n_0$, a set of nodes N is called a partition of T, if it satisfies:*

(1) $N \subseteq N_T$;

(2) $\forall n_i, n_j \in N, i \neq j, LN(n_i) \cap LN(n_j) = \emptyset$;

(3) $\bigcup_{n_i \in N} LN(n_i) = LN(n_0)$;

Obviously, for any product structure tree T, partitions that satisfy condition 1, 2 and 3 in Definition 8 exists. The partitions of the product structure subtree shown by Figure 4.2 include $\{n_0\}$, $\{n_1, n_2, n_3, n_4, n_5\}$, $\{n_1, n_6, n_7, n_8, n_3, n_4, n_5\}$, $\{n_1, n_2, n_3, n_4, n_9, n_{10}, n_{11}, n_{12}\}$ and $\{n_1, n_6, n_7, n_8, n_3, n_4, n_9, n_{10}, n_{11}, n_{12}\}$.

4.3.2 Allocations

We denote the set of the manufacturer and its suppliers as S. $S = \{s_0, s_1, s_2, \dots\}$, where s_0 is the manufacturer. Table 4.3 lists all suppliers for the regeneration system of the natural gas dryer. Supplier's capabilities can be described in two ways: what components

Table 4.3: Suppliers of the natural gas dryer

Supplier	Description
s_0	Manufacturer
s_1	Competitor
s_2	Blower supplier
s_3	Heater supplier
s_4	Cooler supplier

a particular supplier can supply and what suppliers can supply a particular component. In this thesis, we define two functions, F_{sc} and F_{cs}, to describe supplier's capabilities.

(1) $F_{sc}(s) : S \rightarrow 2^{N_T}, \forall s \in S, F_{sc}(s) = \{n \mid n \in N_T, s \text{ can supply } n\}$. Table 4.5 gives a F_{sc} function.

(2) $F_{cs}(n) : N_T \rightarrow 2^S, \forall n \in N_T, F_{cs}(n) = \{s \mid s \in S, n \in F_{sc}(s)\}$. Table 4.4 gives a F_{cs} function.

Table 4.4: Supplier capability function

Component n	$F_{cs}(n)$
n_1	s_0
n_2	s_1, s_2
n_3	s_1, s_3
n_4	s_0
n_5	s_1, s_4

Table 4.5: Supplier capability function

Supplier s	$F_{sc}(s)$
s_0	n_1, n_4
s_1	n_2, n_3, n_5
s_2	n_2
s_3	n_3
s_4	n_5

For the purpose of simplicity, we assume that if a component is allocated to a supplier, all of its child components are also allocated to the same supplier.

Definition 9 (Allocation). *T is a product structure tree, N_T is the set of all nodes of T, S is a set of suppliers, F_{sc} is the supplier capability function, a mapping $F_a : N \rightarrow S$ is called an allocation, if it satisfies:*

(1) N is a partition of T;

(2) $\forall n \in N, \exists s \in S$, if $F_a(n) = s$, then $n \in F_{sc}(s)$;

It is not necessarily true that there is such a F_a in all cases; but with an additional condition $\exists$*partition* $N, N \subseteq \bigcup_{s_i \in S} F_{sc}(s_i)$, it can be easily proved that F_a exists.

Lemma 1 (Sufficient condition for existence of allocations). *If N is a partition of T and $N \subseteq \bigcup_{s_i \in S} F_{sc}(s_i)$, there exists at least one allocation $F_a : N \rightarrow S$.*

Proof. First, we construct a function $F : N \rightarrow S$. For each $n \in N$, since $N \subseteq \bigcup_{s_i \in S} F_{sc}(s_i)$, $n \in \bigcup_{s_i \in S} F_{sc}(s_i)$; so $\exists s_j \in S, n \in F_{sc}(s_j)$; let $F(n) = s_j$.

Then we prove that F is an allocation. (1) N is a partition of T; (2) $\forall n \in N$ *and* $s \in S$, if $F(n) = s$, according to the construction of F, we have $n \in F_{sc}(s)$. F satisfies conditions in Definition 9, so it is an allocation. □

If F_a exists, we say that F_{sc} is sufficient. The F_{sc} given in Table 4.5 is sufficient. Table 4.6 lists all possible allocations.

Table 4.6: Allocations

n	F_a^1	F_a^2	F_a^3	F_a^4	F_a^5	F_a^6	F_a^7	F_a^8
n_1	s_0	s_0	s_0	s_0	s_0	s_0	s_0	s_0
n_2	s_1	s_1	s_1	s_1	s_2	s_2	s_2	s_2
n_3	s_1	s_1	s_3	s_3	s_1	s_1	s_3	s_3
n_4	s_0	s_0	s_0	s_0	s_0	s_0	s_0	s_0
n_5	s_1	s_4	s_1	s_4	s_1	s_4	s_1	s_4

In practice, product structure trees, suppliers and supplier capability functions can be more complex than the one given in Figure 4.2, Table 4.3, Table 4.5 and Table 4.4. Therefore, we need an algorithm that finds all possible allocations for allocating components and tasks to suppliers while considering the product structure and supplier capabilities. Algorithm 4 gives such an algorithm .

The following is a brief discussion of this algorithm. A node may be redundant to allocation search if there is no supplier who can supply the components that the node represents. If a node $n \in LN(n_0)$ where n_0 is the root node of T, and $F_{cs}(n)$ is empty, n is impossible within any partitions. So we will remove node n and its sibling nodes (including their descendants) from T. The removal operation will make the parent node of node n a new leaf node. We will repeat the process until all nodes in $LN(n_0)$ and all new leaf nodes generated by removal operations have been checked.

Algorithm 4 Find All Allocations

Input: Product Structure Tree T; Suppliers S; Supplier capability functions F_{sc} and F_{cs};

Output: All allocations $\widetilde{F_a}$;

1: Remove redundant nodes from T;
2: Get all partitions $\widetilde{N}$;
3: $\widetilde{F_a} = \emptyset$;
4: $N_s = \bigcup_{s_i \in S} F_{sc}(s_i)$;
5: **for all** partition $N \in \widetilde{N}$ **do**
6: **if** $N \subseteq N_s$ **then**
7: Get allocations $\{F_a\}$ corresponding to N;
8: $\widetilde{F_a} = \widetilde{F_a} \cup \{F_a\}$;
9: **end if**
10: **end for**

We use a recursive algorithm to obtain all partitions. First, if n_0 is the root node of T, n_0 is a partition of T; then if n_0 has m child nodes, $\{n_1, n_2, \ldots, n_m\}$, we recursively obtain partitions of every subtree of T with root node $n_i \in \{n_1, n_2, \ldots, n_m\}$. If $\widetilde{N_i}$ is partitions of the subtree of T with root node n_i, $\forall N_i \in \widetilde{N_i}, \bigcup_{1 \leq i \leq m} N_i$ is a partition of T.

For each partition N, we use $N \subseteq \bigcup_{s_i \in S} F_{sc}(s_i)$ to decide if it will be used to generate corresponding allocations. For a particular partition N that satisfies the above condition, we can construct all allocations corresponding to it by N and supplier capability function F_{cs}.

In Lemma 1, we have proved that N is a partition and $N \subseteq \bigcup_{s_i \in S} F_{sc}(s_i)$ are the sufficient conditions for the existence of allocations. Based on Definition 9, we can prove that they are also the necessary conditions for existence of allocations. Therefore, it is easy to prove that Algorithm 4 is sound and complete.

Lemma 2 (Necessary conditions for existence of allocations). *If there exists an allocation $F_a : N \rightarrow S$, N is a partition of T, and $N \subseteq \bigcup_{s_i \in S} F_{sc}(s_i)$.*

Proof. First, if $F_a : N \rightarrow S$ is an allocation, N is a partition of T;

Then, if $F_a : N \rightarrow S$ is an allocation, $\forall n \in N, \exists s \in S, n = F_{sc}(s)$; since $F_{sc}(s) \subseteq \bigcup_{s_i \in S} F_{sc}(s_i)$, we get $\forall n \in N, n \subseteq \bigcup_{s_i \in S} F_{sc}(s_i)$; it means $N \subseteq \bigcup_{s_i \in S} F_{sc}(s_i)$. □

Theorem 1 (Necessary and sufficient conditions for existence of allocations). *The necessary and sufficient conditions for existing an allocation $F_a : N \rightarrow S$ are N is a partition of T and $N \subseteq \bigcup_{s_i \in S} F_{sc}(s_i)$.*

Proof. Based on Lemma 1 and Lemma 2, Theorem 1 is proved. □

4.3.3 The optimization problem

According to Definition 9, an allocation is a mapping from components or tasks to suppliers, satisfying the constraints of product structure and supplier capabilities. We can describe an allocation F_a with a binary matrix A. If there are m components or tasks and n suppliers, we can construct matrix $A = [a_{ij}]_{m \times n}$ as following.

$$a_{ij} = \begin{cases} 1 & \text{if } F_a \text{ allocats component or task } i \text{ to supplier } j \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

If a binary matrix A describes an allocation F_a as defined by Equation 4.5, we call it allocation matrix A, or more briefly allocation A.

If a set of components or tasks are allocated to a supplier, a set of relevant information has to be shared with that supplier. Therefore, given a private parameter and a supplier, we can calculate the risk that the private parameter is leaked to the supplier under a specific allocation based on Algorithm 3 and Definition 7. For an allocation, all the risks that private parameters are leaked to suppliers form a matrix R. If there are p private parameters and n suppliers, matrix R will have p rows and n columns. We can denote the risk matrix as $R = [r_{kj}]_{p \times n}$, where r_{kj} is the risk that private parameter k is leaked to supplier j.

Given a private parameter k and a supplier j, we can define a threshold t_{kj}. If the risk that private parameter k is leaked to supplier j is lower than t_{kj}, we consider the parameter sharing is "safe"; otherwise, we consider it is "unsafe". The thresholds for all combinations of private parameters and suppliers form a risk threshold matrix $T = [t_{kj}]_{p \times n}$.

We define $R < T$ if and only if $\forall\ k,\ j,\ r_{kj} < t_{kj}$. To mitigate the risk of information

leakage, we hope we can find an allocation A that satisfies the constraint of $R < T$.

If component or task i is allocated to supplier j, the total cost is c_{ij}. For all components or tasks and suppliers, the costs form a cost matrix $C = [c_{ij}]_{m \times n}$. From the perspective of cost, we hope we can find an allocation A that has the minimal cost. It can be described as $min \sum_{i,\ j} c_{ij} \times a_{ij}$.

Based on the above discussion, the optimization problem of partner selection can be described as finding an allocation A that satisfies

$$min \sum_{i,j} c_{ij} \times a_{ij} \tag{4.6}$$

$$R < T. \tag{4.7}$$

4.3.4 A generic process

Based on discussions in Section 4.3.1, Section 4.3.2 and Section 4.3.3, we can give a generic process of supplier selection to mitigate the risk of information leakage caused by inferences in supply chains.

Step 1: Find all allocations;

At this step, the constraints of product structure and supplier capabilities are considered. We can use Algorithm 4 to find all allocations.

Step 2: Find all safe allocations;

For an allocation, a private parameter and a supplier, we can calculate its risk of information leakage caused by inferences with Algorithm 3. For each allocation, we can construct a risk matrix R. By comparing risk matrices with the risk threshold matrix T, we can find allocations that are "safe" while the risk of information leakage caused by inferences is considered.

Step 3: Optimize on operational cost;

We can calculate the operational cost of each "safe" allocation, and then find the allocation with the minimum cost. At this step, the problem of supplier selection to mitigate the risk of information leakage caused by inferences in supply chains becomes a classic cost optimization problem, which can be solved by many numerical or non-numerical methods.

With the above three steps, the manufacturer can find one or more allocations from components and tasks to suppliers that satisfy the constraints of product structure and supplier capabilities with low risks of information leakage caused by inferences and minimum operational cost. These allocations will make it possible for the manufacturer to mitigate its risks of information leakage caused by inferences before sharing information with its suppliers. A complete example for supplier selection will be given in Chapter 5.

Chapter 5

IMPLEMENTATION AND CASE STUDY

5.1 The software prototype

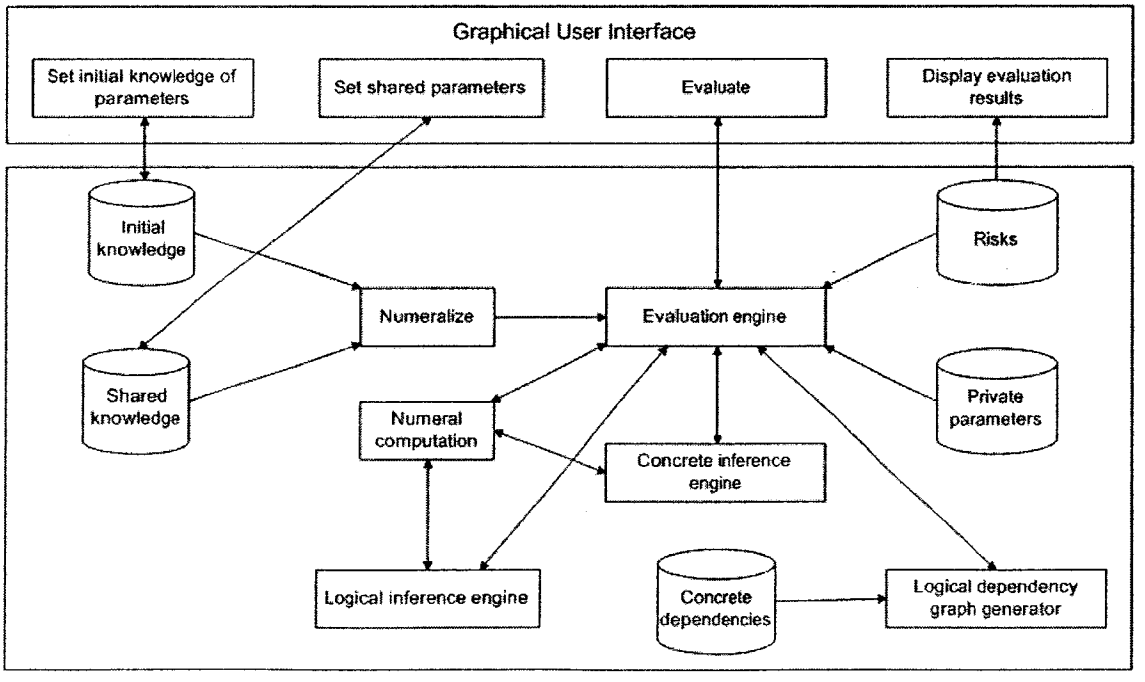

Figure 5.1: The software framework of the software prototype

We developed a software prototype to calculate the probability of information leakage caused by inferences. Figure 5.1 is the framework of the software prototype.

With the software prototype, we can model the inferrer's initial knowledge of parameters as shown by Figure 5.2. Now the software prototype supports three types of probability distributions, namely discrete distributions, continuous uniform distributions and normal distributions.

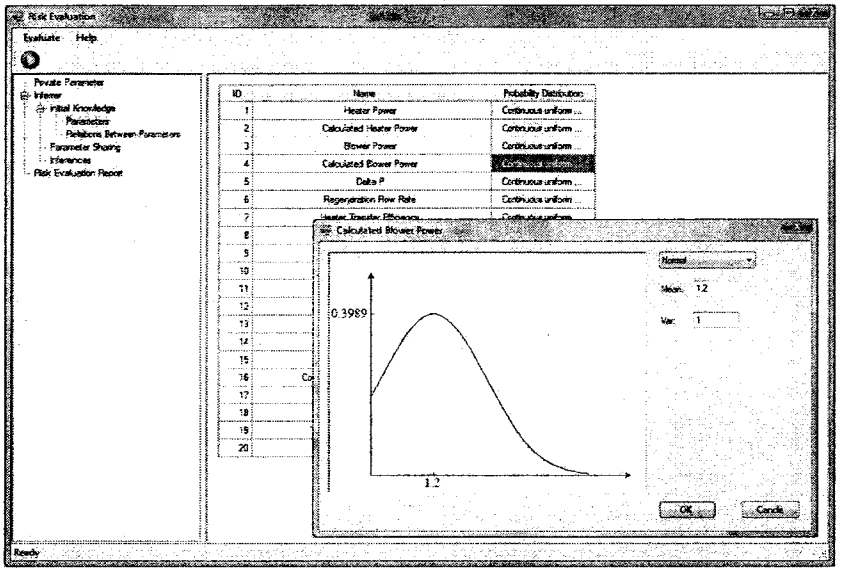

Figure 5.2: Modeling the inferrer's initial knowledge of parameters

As shown by Figure 5.3, the software prototype allows the user to choose what parameters the holder will share with the inferrer and with what probability distributions these parameters are shared.

The software prototype can calculate the probability distribution of a private parameter that may be obtained through inferences, which is an implementation of Algorithm 2, and the probability of information leakage caused by inferences, which is an implementation of Algorithm 3. The calculations are based on the inputs of concrete design parameters and their relations, using numerical computation methods. Figure 5.4 is a screenshot of the software prototype that shows the result of risk evaluation.

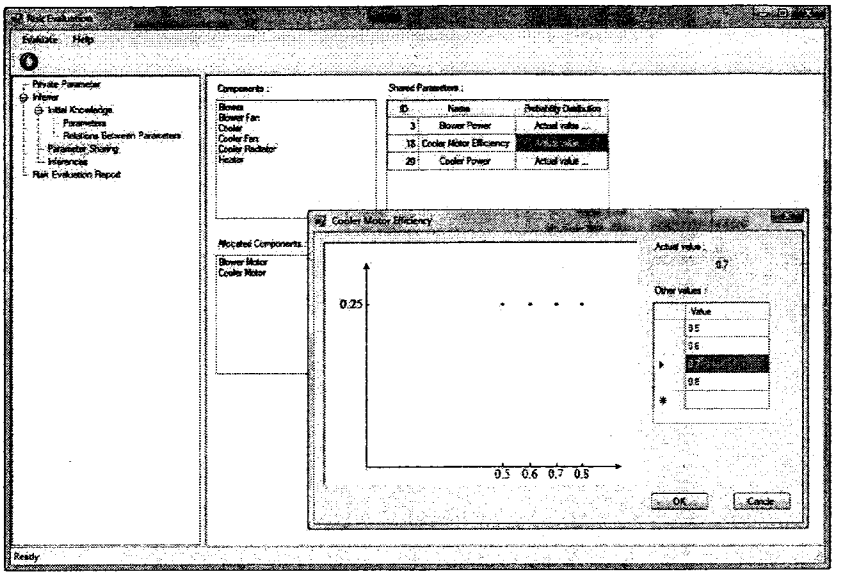

Figure 5.3: Setting parameter sharing

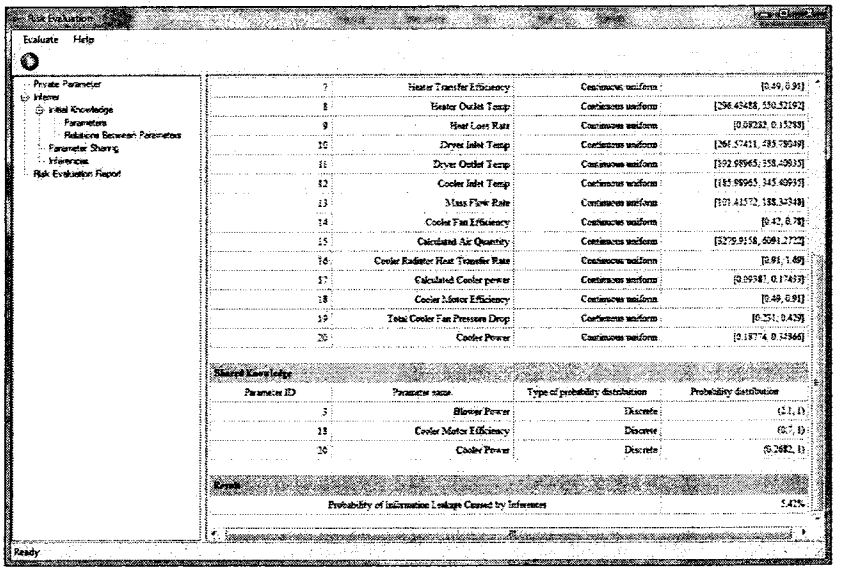

Figure 5.4: A screenshot of the prototype that shows the result of risk evaluation

The software prototype can also display logical dependency graph (as shown by Figure 5.5) and the recursive inference process (as shown by Figure 5.6).

The software prototype is developed in C Sharp. It consists of seven packages, namely *RiskEvaluation*, *Forms*, *Graphic*, *LogicalDependencyGraph*, *TreeLikeStructure*, *Distributions* and *NumeralizedDistributions*. A UML package dependency diagram is shown in

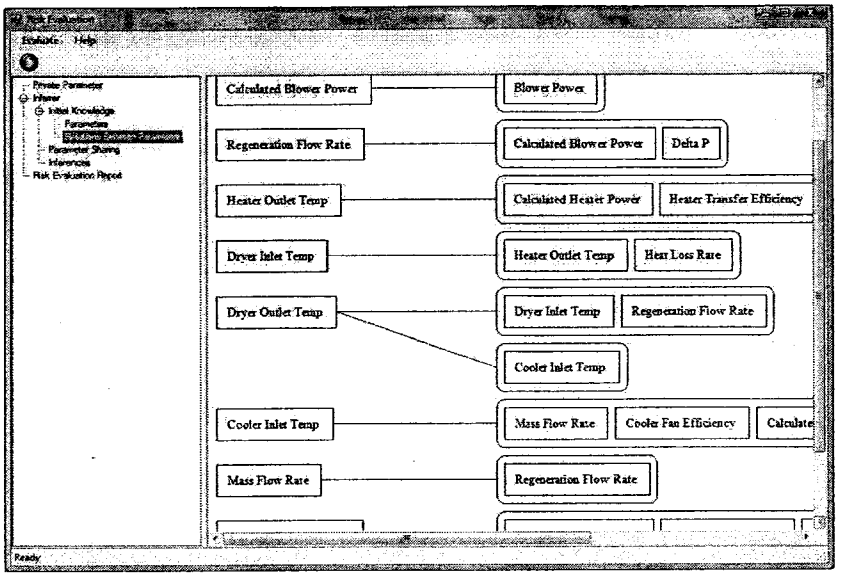

Figure 5.5: A screenshot of the prototype that shows logical dependency graph

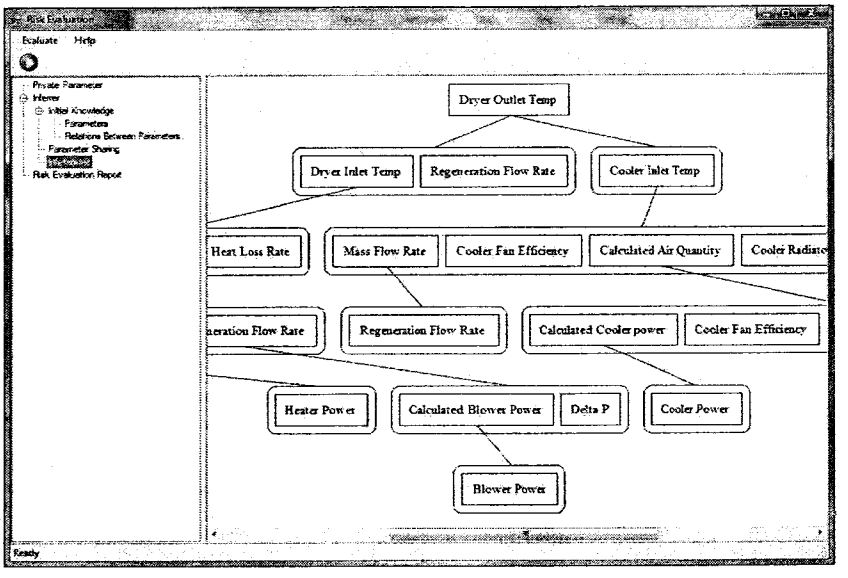

Figure 5.6: A screenshot of the prototype that shows the recursive inference process

Figure 5.7.

Package *RiskEvaluation* is the core of the implementation, which contains three classes:

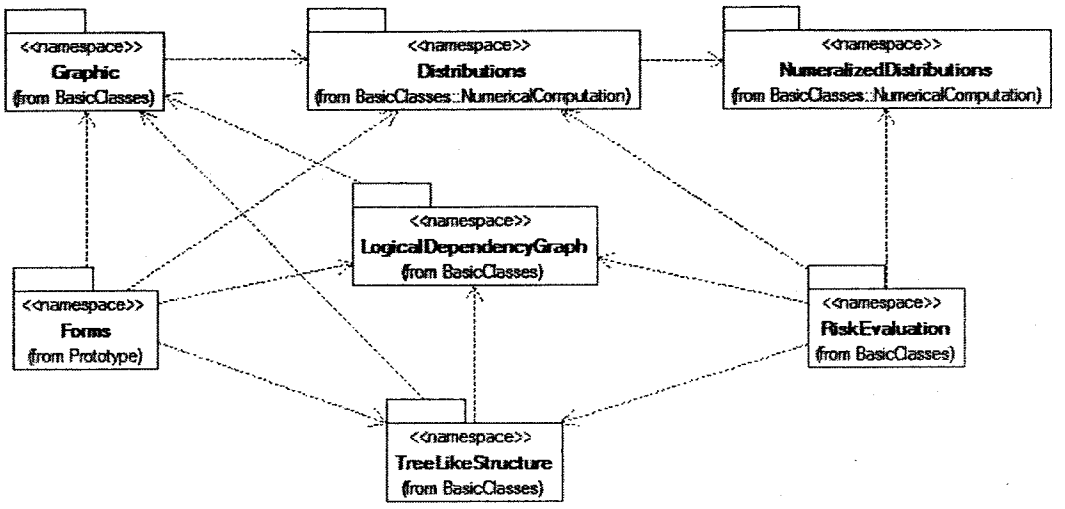

Figure 5.7: Packages and package dependencies

LogicalInferenceEngine, *ConcreteInferenceEngine* and *EvaluationEngine*. Class *LogicalInferenceEngine* implements Algorithm 2. Class *LogicalInferenceEngine* implements Algorithm 3. Class *ConcreteInferenceEngine* is designed to support the inferences at line 9 of Algorithm 1 and at line 10 of Algorithm 2 based on concrete relations among design parameters or other information. In current implementation, Class *ConcreteInferenceEngine* works only under some specific situations. We will improve its implementation in our future work.

Package *Forms* contains all classes related to the graphic user interface of the software prototype. Package Graphic (as shown in Figure 5.8) provides basic graphic classes for classes in Package *LogicalDependencyGraph* and package *TreeLikeStructure*.

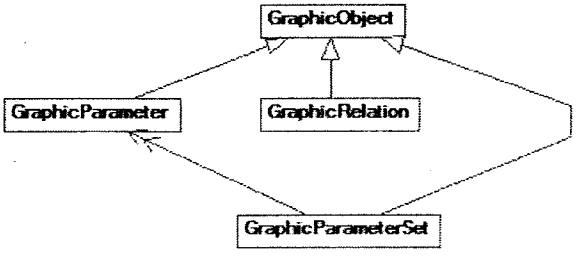

Figure 5.8: Classes in package Graphic

Package *LogicalDependencyGraph* (as shown in Figure 5.9) and *TreeLikeStructure* (as

shown in Figure 5.10) contain classes for generating, storing and drawing logical dependency graphs and recursive inference processes, respectively.

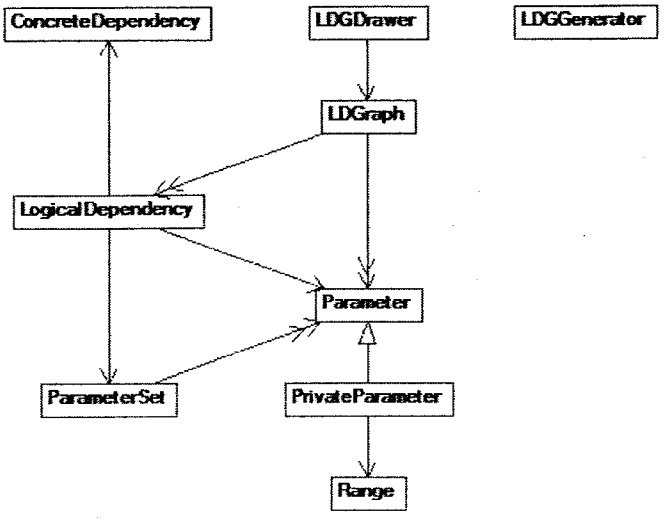

Figure 5.9: Classes in package LogicalDependencyGraph

As shown in Figure 5.11, package *Distributions* defines three classes, *NormalDist*, *DiscreteDist* and *ContinuousUniformDist*, to support the storage and display of normal distribution, discrete distribution and continuous uniform distribution, respectively. All classes in package *Distributions* implement interface *Numeralizable*, which contains a method *Numeralize* that generates a corresponding numeralized probability distribution defined in Package *NumeralizedDistributions*, which is shown in Figure 5.12. Besides

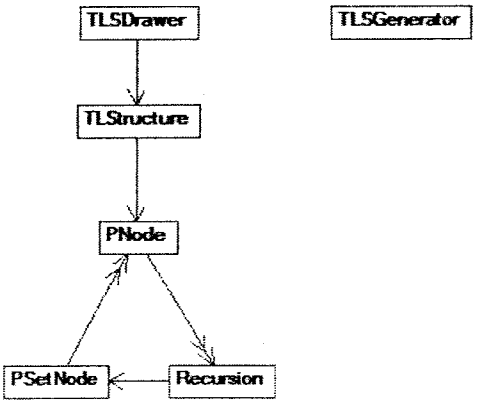

Figure 5.10: Classes in package TreeLikeStructure

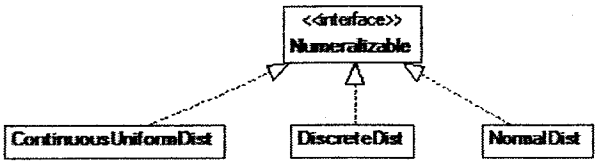

Figure 5.11: Classes in package Distributions

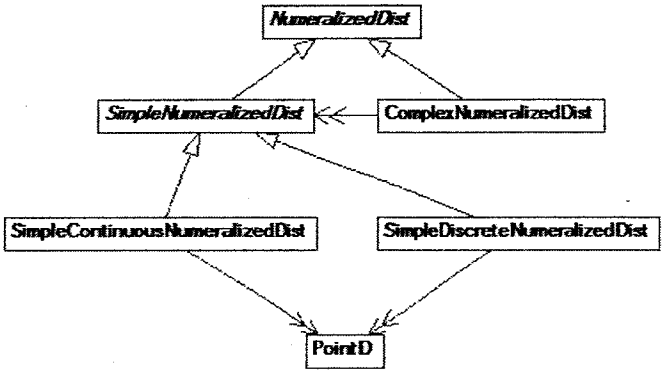

Figure 5.12: Classes in package NumeralizedDistributions

classes corresponding to simple probability distributions such as continuous uniform distribution, discrete distribution and normal distribution, package *NumeralizedDistributions* also contains a class named *ComplexNumeralizedDist* to support complex numeralized distribution.

5.2 A case study

In this section, we present a case study on the regeneration system of a natural gas dryer. A natural gas dryer is a device to remove water from compressed natural gas. A dual tower natural gas dryer has two chambers. Natural gas is dried by the desiccant in one chamber while the desiccant in another chamber is being regenerated.

The regeneration system consists of four major components: blower, heater, dryer and cooler. Figure 5.13 shows a product structure tree of the regeneration system. The regeneration system uses natural gas as regeneration gas. First, the blower is used to increase the pressure at the outlet of the blower to force regeneration gas flow toward the heater;

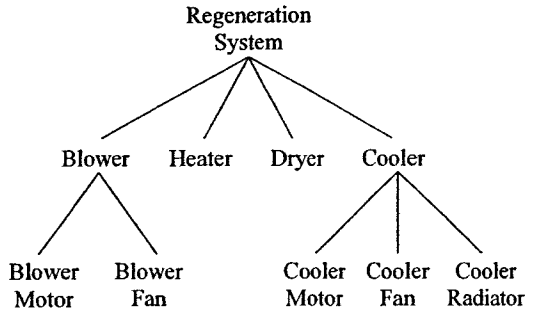

Figure 5.13: A product structure tree of the regeneration system

the heater blower heats regeneration gas to a high temperature; when hot regeneration gas passes through the dryer, it will take the moisture away from the desiccant; the cooler separates the moisture from regeneration gas by condensation.

The design of the regeneration system is crucial to the efficiency of the natural gas dryer. Therefore, the manufacturer wants to prevent the design parameters of the regeneration system, including pressures, temperatures and flow rates, from being revealed to its (potential) competitors.

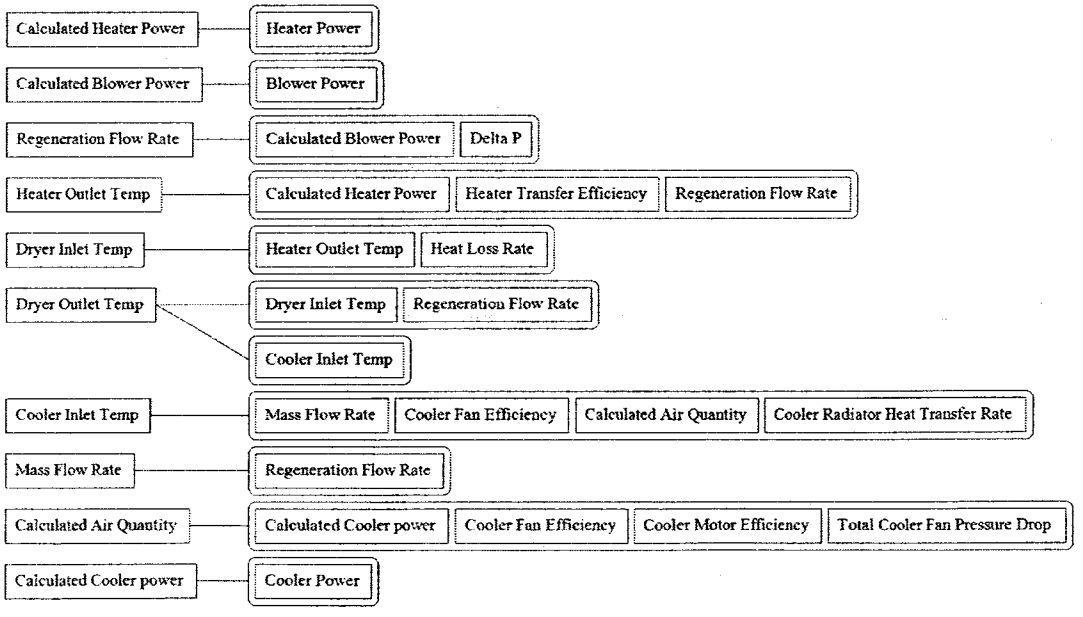

Figure 5.14: The logical dependency graph in the case study

In this case study, the manufacturer is the holder; a supplier and potential competitor

is the inferrer; design parameter *DryerOutletTemp* is the private parameter p_0; the logical dependency graph is shown in Figure 5.14; what design parameters the manufacturer share with a supplier usually depends on what components the supplier supplies. Table 5.1 gives the relation between components and shared design parameters.

Table 5.1: The relation between components and shared design parameters

Component	Shared Parameters
Blower	Blower Power, Delta P
Blower Fan	Blower Power, Delta P
Blower Motor	Blower Power
Cooler	Cooler Fan Efficiency, Cooler Radiator Heat Transfer Rate, Cooler Motor Efficiency, Total Cooler Fan Pressure Drop, Cooler Power
Cooler Fan	Cooler Fan Efficiency, Total Cooler Fan Pressure Drop
Cooler Motor	Cooler Motor Efficiency, Cooler Power
Cooler Radiator	Cooler Radiator Heat Transfer Rate
Heater	Heater Power, Heater Transfer Efficiency

We calculated the probabilities of information leakage caused by inferences with the help of the software prototype. Figure 5.15 gives the tree-like-structure that shows the recursions. Table 5.2 gives a group of probabilities of information leakage caused by inferences. The results are obtained when initial knowledge of parameters is assigned with continuous uniform distributions on the ranges of from $0.7 \times$ *actual value* to $1.3 \times$ *actual value*, parameters are shared with their actual values, and the working values of the private parameter p_0 are with the range of from $0.99 \times$ *actual value* to $1.01 \times$ *actual value*.

There are some interesting points in the above results.

(1) As in this case study, the probability may not be monotonous in "Number of components the inferrer supplies" or "Number of design parameters shared with the inferrer". The monotonicity depends on the concrete relations between design parameters and the private parameter p_0, and the probability distributions assigned to design parameters as the inferrer's initial knowledge and shared knowledge. The conditions of the monotonicity are an interesting topic in the future.

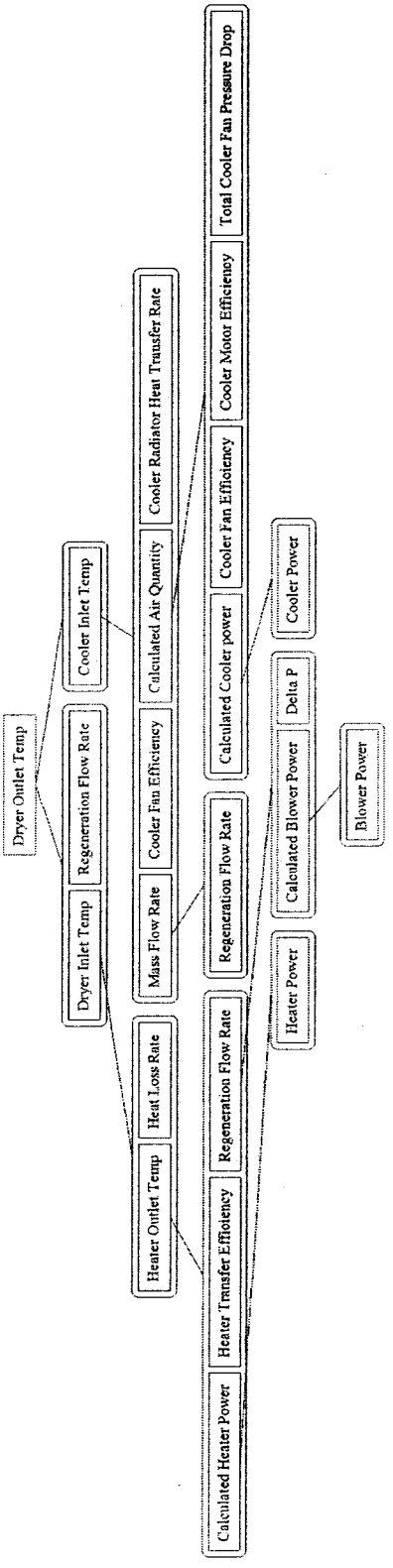

Figure 5.15: The recursive inference process in the case study

Table 5.2: Components the suppler supplies and the probabilities of information leakage caused by inferences

Components	Probability
None	4.04%
Blower	3.39%
Cooler	3.84%
Heater	3.23%
Blower & Cooler	100%
Blower & Heater	19.82%
Cooler & Heater	3.03%
Blower, Cooler & Heater	100%
Blower & Cooler Motor	3.39%
Blower Motor & Cooler Motor	5.42%
Blower Motor & Heater	6%
Blower Motor, Cooler Motor & Heater	5.99%

(2) Although the probability is not monotonous, the mitigation principles we discussed in Section 4.2 are still effective when the probabilities are high. For example, if the inferrer supplies both the blower and the heater, the probability will be 19.82%; if the inferrer supplies the blower motor and the heater, the probability will be decreased to 6%.

We have introduced some parts of the example for supplier selection in Section 4.3. In this section, we will continue the example.

Supplier s_1, s_1, s_1 and s_4 may have different initial knowledge of parameters. Corresponding to the supplier capabilities given in Table 4.4 and Table 4.5, we assign continuous uniform distributions on the ranges of from $0.7 \times$ *actual value* to $1.3 \times$ *actual value* to the initial knowledge of parameters relevant to components that a supplier has the capability to supply, and continuous uniform distributions on the ranges of from $0.4 \times$ *actual value* to $1.6 \times$ *actual value* to the other parameters in this example. Table 5.3 gives components and their relevant parameters. Table 5.4 gives the probability of information leakage of private parameter *DryerOutletTemp* caused by inferences for each combination of allocation and supplier. The results in Table 5.4 are obtained when parameters are shared with their

actual values, and the working values of the private parameter p_0 are within the range of from 0.99 × *actual value* to 1.01 × *actual value*.

Table 5.3: Components and relevant parameters

Component	Relevant Parameters
n_2	Blower Power, Calculated Blower Power, Delta P, Regeneration Flow Rate, Mass Flow Rate
n_3	Heater Power, Calculated Heater Power, Heater Transfer Efficiency, Heater Outlet Temp, Regeneration Flow Rate, Mass Flow Rate
n_5	Cooler Inlet Temp, Cooler Fan Efficiency, Calculated Air Quantity, Cooler Radiator Heat Transfer Rate, Calculated Cooler Power, Cooler Motor Efficiency, Total Cooler Fan Pressure Drop, Cooler Power, Regeneration Flow Rate, Mass Flow Rate

Table 5.4: Allocations, suppliers and the probabilities of information leakage caused by inferences

	s_1	s_2	s_3	s_4
F_a^1	100%	1.82%	2.07%	2.77%
F_a^2	9.56%	1.82%	2.07%	3.17%
F_a^3	100%	1.82%	2.14%	2.77%
F_a^4	5.14%	1.82%	2.14%	3.17%
F_a^5	3.04%	1.18%	2.07%	2.77%
F_a^6	3.32%	1.18%	2.07%	3.17%
F_a^7	2.73%	1.18%	2.14%	2.77%
F_a^8	2.93%	1.18%	2.14%	3.17%

Table 5.6 gives the risk thresholds used in this example. Since supplier s_1 is a potential competitor, we assign a lower threshold 5% than other suppliers.

Table 5.5: Suppliers and risk thresholds

	s_1	s_2	s_3	s_4
Threshold	5%	10%	10%	10%

Compared probabilities in Table 5.4 and thresholds in Table 5.6, it is easy to find allocation F_a^5, F_a^6, F_a^7 and F_a^8 are "safe", considering the risk of information leakage caused

by inferences.

For each allocation in Table 4.6, we can calculate the costs on component n_2, n_3 and n_5. The total costs are 6, 7, 7, 8, 7, 8, 8 and 9, respectively, when the cost for each combination of component and supplier is assigned as given in Table 5.6.

Table 5.6: Components, suppliers and costs

	s_1	s_2	s_3	s_4
n_2	2	3	100	100
n_3	2	100	3	100
n_5	2	100	100	3

In this example, allocation F_a^1 is the optimal solution when only cost is considered. When both cost and the of information leakage caused by inferences are considered, the optimal solution becomes F_a^5.

Chapter 6

CONCLUSIONS AND FUTURE WORK

In this thesis, we studied the security issue related to information leakage caused by inferences in supply chains with conceptual model and quantitative methods. We proposed a conceptual model of information leakage caused by inferences in supply chains. With the conceptual model, the companies who hold confidential information can model and understand potential information leakage caused by inferences in supply chains more clearly. We put forward a quantitative approach to evaluating the risk of information leakage caused by inferences in supply chains. Our quantitative approach can help companies identify confidential information, and evaluate and mitigate the risk of potential information leakage caused by inferences in supply chains.

It is an interesting and challenging research topic to prevent information leakage caused by inferences in supply chains. The following lists some points that we may work in the future.

(1) In this thesis, we studied information leakage caused by inferences in supply chains with conceptual model and quantitative methods. Although it may be more challenging, modeling the problem with analytical and theoretical methods, such as game

theory and information theory, may bring out more fundamental results and solutions.

(2) We modeled the inferrer's knowledge with probability distributions and logical dependency graph. One problem remained is that knowledge may be inaccurate and it may be inconsistent when it is from multiple sources. It would be interesting to work out and evaluate various strategies to handle knowledge inaccuracies and inconsistencies in the context of preventing information leakage caused by inferences in supply chains.

(3) We discussed three mitigation approaches in principle and mitigation with supplier selection. These approaches will be investigated further in the future. We are especially interested in preventing information leakage caused by inferences through product design and supply chain design.

(4) We studied a case extracted from a product in process industry. In future research, we will explore products in other industries, and validate our methods in other real world applications.

PUBLICATIONS

1. **Da Yong Zhang**, Lingyu Wang, Yong Zeng, Secure Collaborative Product Development: a Literature Review, Proceedings of International Conference on Product Lifecycle Management, 2008 (PLM'08), July 9-11, 2008, Seoul, South Korea.

2. Shuai Liu, **Da Yong Zhang**, Xiao Chu, H. Otrok, P. Bhattacharya, A Game Theoretic Approach to Optimize the Performance of Host-Based IDS, Proceedings of IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, 2008 (WIMOB'08), 12-14 October 2008, Avignon, France, pp.448-453.

3. Wei Liu, Min Wang, **Da Yong Zhang**, Yong Zeng, Development of Mathematical Models for Secure Collaboration in Product Lifecycle Management, Proceedings of International Conference on Engineering Design, 2009 (ICED'09), 24-27 August 2009, Stanford, CA, USA.

4. **Da Yong Zhang**, Yong Zeng, Lingyu Wang, Hongtao Li, Yuanfeng Geng, Modeling and Evaluating Information Leakage Caused by Inferences in Supply Chains, submitted to Computer in Industry.

Bibliography

[AC] Foreign Affairs and International Trade Canada. Export and Import Permit Act. http://www.dfait-maeci.gc.ca/sanctions/eipa-llei.aspx?lang=eng.

[AEDS03] M. J. Atallah, H. G. Elmongui, V. Deshpande, and L. B. Schwarz. Secure supply-chain protocols. In *Proceedings of IEEE International Conference on E-Commerce 2003*, pages 293–302, 2003.

[AG09] K. S. Anand and M. Goyal. Strategic information management under leakage in a supply chain. *Management Science*, 55(3):438–452, 2009.

[BG03] P. D. Burns and I. M. Girard. The Candaian export controls regime and the Controlled Goods Program. *Canadian International lawyer*, 5(3):117–128, 2003.

[Cal06] A. Calder. *Information security based on ISO 27001/ISO 17799: a management guide.* Van Haren Publishing, 2006.

[CBK+06] C. D. Cera, I. Braude, T. Kim, J. Han, and W. C. Regli. Hierarchical role-based viewing for multi-level information security in collaborative CAD. *Journal of Computing and Information Science in Engineering*, 6(1):2–10, 2006.

[CCC08] T.-Y. Chen, Y.-M. Chen, and H.-C. Chu. Developing a trust evaluation method between co-workers in virtual project team for enabling resource sharing and collaboration. *Computers in Industry*, 59(6):565–579, 2008.

[Che03] S. Cheadle. Export compliance: Understanding ITAR and EAR. *Microwave journal*, 48(10):80–91, 2003.

[CKHR04] C. D. Cera, T. Kim, J. Han, and W. C. Regli. Role-based viewing envelopes for information protection in collaborative modeling. *Computer-Aided Design*, 36(9):873–886, 2004.

[CN06] E. Choi and S. Niculescu. The impact of US export controls on the Canadian space industry. *Space Policy*, 22(1):29–34, 2006.

[CSF04] L. Chen, Z. Song, and L. Feng. Internet-enabled real-time collaborative assembly modeling via an e-assembly system: status and promise. *Computer-Aided Design*, 36(9):835–847, 2004.

[dBLM01] L. de Boer, E. Labro, and P. Morlacchi. A review of methods supporting supplier selection. *European Journal of Purchasing & Supply Management*, 7(2):75–89, 2001.

[Fia05] P. Fiala. Information sharing in supply chains. *Omega*, 33(5):419–423, 2005.

[FKS07] D. F. Ferraiolo, R. Kuhn, and R. S. Sandhu. RBAC standard rationale: comments on a critique of the ANSI standard on Role Based Access Control. *IEEE Security & Privacy*, 5(6):51–53, 2007.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the 19th annual ACM conference on theory of computing*, pages 218–229, 1987.

[GSK06] I. Geyskens, J.-B. E. M. Steenkamp, and N. Kumar. Make, buy, or ally: A transaction cost theory meta-analysis. *Academy of Management Journal*, 49(3):519–543, 2006.

[HFPS99] R. Housley, W. Ford, W. Polk, and D. Solo. Internet x.509 public key infrastructure certificate and crl profile, 1999. RFC2459.

[HLM03] G. Q. Huang, J. S. K. Lau, and K. L. Mak. The impacts of sharing production information on supply chain dynamics: a review of the literature. *International Journal of Production Research*, 41(7):1483–1517, 2003.

[HT06a] A. Hoecht and P. Trott. Innovation risks of strategic outsourcing. *Technovation*, 26(5-6):672–681, 2006.

[HT06b] A. Hoecht and P. Trott. Outsourcing, information leakage and the risk of losing technology-based competencies. *European Business Review*, 18(5):395–412, 2006.

[KA98] S. Kent and R. Atkinson. Security architecture for the internet protocol, 1998. RFC2401.

[KCR+06] T. Kim, C. D. Cera, W. C. Regli, H. Choo, and J. Han. Multi-level modeling and access control for data sharing in collaborative design. *Advanced Engineering Informatics*, 20(1):47–57, 2006.

[KN93] J. Kohl and C. Neuman. The kerberos network authentication service (v5), 1993. RFC1510.

[KWMN04] K.-Y. Kim, Y. Wang, O. S. Muogboh, and B. O. Nnaji. Design formalism for collaborative assembly design. *Computer-Aided Design*, 36(9):849–871, 2004.

[Li02] L. Li. Information sharing in a supply chain with horizontal competition. *Management Science*, 48(9):1196–1212, 2002.

[LL06] J. R. Liebman and K. J. Lombardo. A guide to export controls for the non-specialist. *Loyola of Los Angeles International and Comparative Law Review*, 28(3):497–520, 2006.

[LP02] Y. Lindell and B. Pinkas. Privacy preserving data mining. *Journal of Cryptology*, 15(3):177–206, 2002.

[LPW97] H. L. Lee, V. Padmanabhan, and S. Whang. Information distortion in a supply chain: the bullwhip effect. *Management Science*, 43(4):546–558, 1997.

[LQ06] W. D. Li and Z. M. Qiu. State-of-the-art technologies and methodologies for collaborative product development systems. *International Journal of Production Research*, 44(13):2525–2559, 2006.

[LST00] H. L. Lee, K. C. So, and C. S. Tang. The value of information sharing in a two-level supply chain. *Management Science*, 46(5):626–643, 2000.

[LW00] H. L. Lee and S. Whang. Information sharing in a supply chain. *International Journal of Manufacturing Technology and Management*, 1(1):79–93, 2000.

[LYL03] K. K. Leong, K. M. Yu, and W. B. Lee. A security model for distributed product data management system. *Computers in Industry*, 50(2):179–193, 2003.

[MCdD07] T. Moyaux, B. Chaib-draa, and S. D'Amours. Information sharing as a coordination mechanism for reducing the bullwhip effect in a supply chain. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(3):396–409, May 2007.

[MHH09] D. Mun, J. Hwang, and S. Han. Protection of intellectual property based on a skeleton model in product design collaboration. *Computer-Aided Design*, 41(9):641–648, 2009.

[MPHR98] R. M. Monczka, K. J. Petersen, R. B. Handfield, and G. L. Ragatz. Success factors in strategic supplier alliances: The buying company perspective. *Decision Sciences*, 29(3):553–577, 1998.

[MSS98] J. C. Mitchell, V. Shmatikov, and U. Stern. Finite-state analysis of ssl 3.0. In *Proceedings of the 7th conference on USENIX security symposium*, pages 201–216, 1998.

[oCa] The Government of Canada. Defence Production Act. `http://laws.justice.gc.ca/PDF/Statute/D/D-1.pdf`.

[oCb] The Government of Canada. Export and Import Permit Act. `http://laws.justice.gc.ca/PDF/Statute/E/E-19.pdf`.

[oCc] U.S. Department of Commerce. Introduction to Commerce Department Export Controls. `http://www.bis.doc.gov/licensing/exportingbasics.htm`.

[oCd] U.S. Department of Commerce. The Commerce Control List. `http://www.access.gpo.gov/bis/ear/ear_data.html#ccl`.

[Off] Office of Foreign Assets Control, U.S. Department of the Treasury. OFAC's mission. `http://www.treas.gov/offices/enforcement/ofac/`.

[oSa] U.S. Department of State. International Traffic in Arms Regulations 2009. `http://pmddtc.state.gov/regulations_laws/itar_official.html`.

[oSb] U.S. Department of State. ITAR Part 121 - The United States Munitions List. `http://pmddtc.state.gov/regulations_laws/documents/official_itar/ITAR_Part_121.pdf`.

[PHR05] K. J. Petersen, R. B. Handfield, and G. L. Ragatz. Supplier integration into new product development: coordinating product, process and supply chain design. *Journal of Operations Management*, 23(3-4):371–388, 2005.

[ROA07] K. Rouibah and S. Ould-Ali. Dynamic data sharing and security in a collaborative product definition management system. *Robotics and Computer-Integrated Manufacturing*, 23(2):217–233, 2007.

[SCFY96] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[SG01] N. Shyamsundar and R. Gadh. Internet-based collaborative product design with assembly features and virtual design spaces. *Computer-Aided Design*, 33(9):637–651, 2001.

[SG02] N. Shyamsundar and R. Gadh. Collaborative virtual prototyping of product assemblies over the Internet. *Computer-Aided Design*, 34(10):755–768, 2002.

[SLKSL08] D. Simchi-Levi, P. Kaminsky, and E. Simchi-Levi. *Designing and managing the supply chain: concepts, strategies and case studies.* McGraw-Hill/Irwin, 3rd edition, 2008.

[Swe02] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):571–588, 2002.

[TAPH05] W. Tolone, G.-J. Ahn, T. Pai, and S.-P. Hong. Access control in collaborative systems. *ACM Computing Surveys*, 37(1):29–41, 2005.

[WABN06] Y. Wang, P. N. Ajoku, J. C. Brustoloni, and B. O. Nnaji. Intellectual property protection in collaborative design through lean information modeling and sharing. *Journal of Computing and Information Science in Engineering*, 6(2):149–159, 2006.

[WC] Public Works and Government Services Canada. Presentation: Controlled Goods Program. `http://ssi-iss.tpsgc-pwgsc.gc.ca/dmc-cgd/publications/presentations/srvl-ovrvw-eng.html`.

[Weia] E. W. Weisstein. Bipartite graph. From MathWorld - - A Wolfram Web Resource. http://mathworld.wolfram.com/bipartitegraph.html.

[Weib] E. W. Weisstein. Power set. From MathWorld - - A Wolfram Web Resource. http://mathworld.wolfram.com/powerset.html.

[Wik] Wikipedia. Supply chain. http://en.wikipedia.org/wiki/Supply_chain.

[Yao86] A. Yao. How to generate and exchange secrete. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pages 162–167, 1986.

[YL06] T. Ylonen and C. Lonvick. The Secure Shell (SSH) authentication protocol, 2006. RFC4252.

[ZC91] Y. Zeng and G. D. Cheng. On the logic of design. *Design Studies*, 12(3):137–141, 1991.

[Zen02] Y. Zeng. Axiomatic theory of design modeling. *Transactions of the SDPS: Journal of Integrated Design and Process Science*, 6(3):1–28, 2002.

[Zen04a] Y. Zeng. Environment-based design: process model. Technical report, Concordia Institute for Information Systems Engineering, Concordia University, Montreal, 2004. p. 40.

[Zen04b] Y. Zeng. Environment-based formulation of design problem. *Transactions of the SDPS: Journal of Integrated Design and Process Science*, 8(4):45–63, 2004.

[Zen08] Y. Zeng. Recursive Object Model (ROM): modeling of linguistic information in engineering design. *Computers in Industry*, 59(6):612–625, 2008.

[ZG99a] Y. Zeng and P. Gu. A science-based approach to product design theory part I: formulation and formalization of design process. *Robotics and Computer-Integrated Manufacturing*, 15(4):331–339, 1999.

[ZG99b] Y. Zeng and P. Gu. A science-based approach to product design theory part II: formulation of design requirements and products. *Robotics and Computer-Integrated Manufacturing*, 15(4):341–352, 1999.

[Zha02] H. Zhang. Vertical information exchange in a supply chain with duopoly retailers. *Production and Operations Management*, 11(4):531–546, 2002.

[ZJ07] H. Zhou and W.C. Benton Jr. Supply chain practice and information sharing. *Journal of Operations Management*, 25(6):1348–1365, 2007.

[ZPA+04] Y. Zeng, A. Pardasani, H. Antunes, Z. Li, J. Dickinson, V. Gupta, and D. Baulier. Mathematical foundation for modeling conceptual design sketches. *ASME Transactions: Journal of Computing and Information Sciences in Engineering*, 4(2):150–159, 2004.

[ZSG04] S. Zhang, W. Shen, and H. Ghenniwa. A review of internet-based product information sharing and visualization. *Computers in Industry*, 54(1):1–15, 2004.

[ZY09] Y. Zeng and S. Yao. Understanding design activities through computer simulation. *Advanced Engineering Informatics*, 23(3):294–308, 2009.