Computers in Industry xxx (2010) xxx–xxx

Contents lists available at ScienceDirect

Computers in Industry

journal homepage: www.elsevier.com/locate/compind

Modeling and evaluating information leakage caused by inferences in supply chains

Da Yong Zhang, Yong Zeng *, Lingyu Wang, Hongtao Li, Yuanfeng Geng

Concordia Institute for Information Systems Engineering, Faculty of Engineering and Computer Science, Concordia University, Montreal, Quebec H3G 1M8, Canada

ARTICLE INFO

Article history:
Received 17 September 2009
Received in revised form 29 May 2010
Accepted 1 October 2010
Available online xxx

Keywords:
Information sharing
Access control
Inference
Risk evaluation
Mitigation

ABSTRACT

While information sharing can benefit supply chains significantly, it may also have an adverse 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 we shall show in this paper, those assumptions are not always true due to potential information leakage caused by inferences. Specifically, we propose a conceptual model of such information leakage. 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, we then devise a quantitative approach to evaluating 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 mitigate the risk of information leakage. Finally, we discuss a case study to illustrate how the proposed approaches work in practice.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

It is widely accepted that information sharing may significantly benefit supply chains [1–4], especially in reducing the bullwhip effect [5,6], decreasing supply chain costs, and increasing the efficiency of collaborative product developments [7,8]. For this reason, it is common for a company in a supply chain to share a large amount of information with its partners.

However, information sharing in supply chains is also a double-edged sword: it may have an adverse effect, namely, information leakage [9–13]. In general, information leakage means 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]

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 show shortly. 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 questions above provided by most existing technical solutions are not fully satisfactory (the detailed review of existing solutions is given in Section 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 may blur the boundary between confidential and non-confidential information. Second, to our best knowledge, the possibility of potential information leakage caused by inferences is simply ignored in most existing technical solutions (a few exceptions will be discussed in Section 2).

* Corresponding author at: Concordia Institute for Information Systems Engineering, Faculty of Engineering and Computer Science, Concordia University, 1455 de Maisonneuve Blvd. West, EV07.633, Montreal, Quebec H3G 1M8, Canada. Tel.: +1 514 848 2424x5801; fax: +1 514 848 3171.
E-mail addresses: da_zha@ciise.concordia.ca (D.Y. Zhang), zeng@ciise.concordia.ca (Y. Zeng), wang@ciise.concordia.ca (L. Wang), li@ciise.concordia.ca (H. Li), geng@ciise.concordia.ca (Y. Geng).

[1] Although <fn0005>information leakage may happen both inside a company and between different companies, we shall focus on the latter case.

0166-3615/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.compind.2010.10.002

In this paper, we present a systematic study of information leakage caused by inferences in supply chains. Our main contribution is two-fold. First, we propose a conceptual model for such information leakage. 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 a supply chain to better understand the nature of potential information leakage caused by inferences regardless of the underlying details of specific applications. Second, on the basis of the conceptual model, we devise a quantitative approach to evaluating 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 a supply chain to take actions against potential information leakage caused by potential inferences, such as measuring and mitigating the risk of such information leakage. To illustrate the application of the proposed model and quantitative approach, we discuss a concrete case study.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 proposes an abstract conceptual model of information leakage caused by inferences in supply chains. Section 4 devises a quantitative approach to evaluating the risks of information leakage caused by inferences in supply chains, and also discusses the principles behind three mitigation approaches. Section 5 presents a case study by applying our approach to the supply chain of a product in process industry. Finally, Section 6 concludes the paper and gives future work.

2. Related work

2.1. Supply Chain Risk Management

Supply Chain Risk Management (SCRM) is a related research area in the literature of Supply Chain Management (SCM). Giunipero and Eltantawy [14] considered Supply Chain Management (SCM) as ‘‘a formal process that involves identifying potential losses, understanding the likelihood of potential losses, and assigning significance to these losses’’ in supply chains. From another perspective, Juttner [15] defined Supply Chain Risk Management (SCRM) as ‘‘the identification and management of risk for the supply chain, through a coordinated approach amongst supply chain members, to reduce supply chain vulnerability as a whole’’.

In the literature, Supply Chain Risk Management (SCRM) usually focuses on risk sources that may cause supply chain disruptions. For example, Lockamy and McCormack [16] classified a supplier's risk sources into three categories: operational, network and external. Operational risk is the risk arising from the people, systems and internal processes through which a company operates. Network risk is the risk arising from the structure of the supplier's network. External risk is the risk arising from events driven by external forces.

Juttner [15] suggested that supply chain risk sources fell into five categories: environment, supply, demand, process and control. Environmental risk sources are uncertainties external to the supply chain, such as political, natural and social uncertainties. Supply risk sources and demand risk sources are internal to the supply chain. The former contains uncertainties arising from supplier activities and general supplier relations; the latter is associated with outbound logistics flows and product demand. Process and control are mechanisms in the supply chain that may amplify or absorb supply chain risks.

As a risk source that will not disrupt supply chains, information leakage is usually ignored in Supply Chain Risk Management (SCRM).

2.2. Information sharing in supply chains

Information leakage is regarded as a challenge to information sharing in supply chains [9]. Therefore, in the literature, related work focuses on the effect of information leakage on information sharing in supply chains.

Li [10] examined 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 [17] 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.

Anand and Goyal [13] studied the effect of demand information leakage on the material and information flows of a supply chain with horizontal competition. In their 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 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.

2.3. Information leakage prevention

Legal, organizational, social and technical methods are often taken to prevent information leakage in supply chains. In this section, we will review existing technical methods by four categories: access control, suppression, generalization and Secure Multi-party Computation(SMC).

2.3.1. Access control

Access control ensures that only authorized users can access specific information. Several access control models have been developed to meet the security requirements of information sharing and collaboration in supply chains.

Leong et al. [18] proposed a mixed access control model for a workspace-oriented Distributed Product Data Management (DPDM) system. In their model, the proposed DPDM 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. [19,20] and Kim et al. [21]. It is achieved through the integration of multi-resolution geometry and Role Based Access Control (RBAC) model [22,23]. 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 [24] 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.

Trust is also considered in some access control models. For example, Chen et al. [25] 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 [25].

Access control-based methods cannot prevent information leakage caused by inferences in supply chains because unauthorized information is not necessary for inferences.

2.3.2. Suppression

Suppression or sanitization 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 [26].

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 contained 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 [27–30]. To build the final assembly model of the product, companies will convert their CAD data in incompatible formats into neutral CAD data format, 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 the native CAD data.

Sanitization may be an effective approach to mitigating 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 paper, we will address these issues.

2.3.3. Generalization

Generalization, which means "replacing (or recoding) a value with a less specific but semantically consistent value" [26], 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. [31] 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 paper, we will address these issues and discuss a generalization-based mitigation approach.

2.3.4. Secure Multi-party Computation

Secure Multi-party Computation (SMC) [32–34] protects confidential information by allowing users to perform joint computation on multiple datasets while not revealing information in these datasets.

Atallah et al. [35] 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 [35].

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

2.4. Other related work

Inference control in statistical databases and privacy preserving data publishing are two related research areas in computer science.

Inference control in statistical databases has been studied for many years, as surveyed in [36], where the key issue is preventing inferences of individual values from their aggregates, which is different from the inference addressed in our paper. Similarly, privacy preserving data publishing (as surveyed in [37]) has drawn significant attentions more recently where relational tables are kept secret while the data are released either in generalized forms or as data mining results, which is again different from our problem.

3. Conceptual model

3.1. The information leakage model

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

(1) A supply chain is a complex network, in which each partner plays one or more roles. While they cooperate for some common interests, partners have different business objectives and some of them are even (potential) competitors.
(2) 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.
(3) At the same time, every partner may also collect, analyze, acquire and utilize information from its (potential) competitors.
(4) 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.
(5) 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

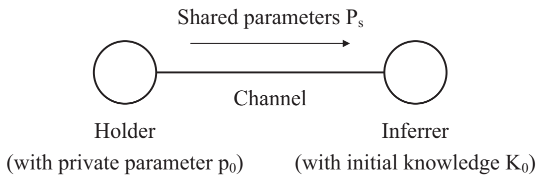


Fig. 1. The conceptual model.

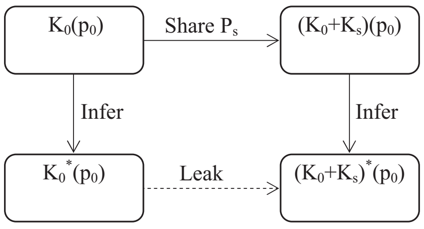


Fig. 2. The information leakage model.

CAD data in collaborative product development; a retailer may share its order information and demand forecast with the manufacturer.

(6) 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, graphs and even natural languages.

We abstract a conceptual model from concrete cases of information leakage caused by inferences in supply chains. The abstract 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 an abstract 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.

Fig. 1 shows the abstract 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 of $\langle name, actual, working \rangle$, 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 paper: 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 abstract conceptual model, the scenario 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.

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 in Fig. 2.

(1) Initially, the inferrer has knowledge K_0;
(2) The inferrer' 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' knowledge becomes $(K_0 + K_s)^*$ after it infers on $K_0 + K_s$;

If the inferrer can obtain more knowledge of the private parameter p_0 from $(K_0 + K_s)^*$ than from K_0^*, there is information of the private parameter p_0 leaked 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 of $\langle name, actual, working \rangle$. 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* and the *working* of a parameter are usually known to the holder; the *actual* and the *working* of the private parameter p_0 are unknown to the inferrer. In fact, the inferrer tries to acquire a *working* 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. Therefore, the inferrer can construct a probability distribution of the parameter by its knowledge. In practice, sometimes it may be infeasible for the inferrer to collect all possible values of a parameter and to calculate their probabilities to construct the probability distribution of the parameter. Typically, the inferrer will describe its 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.

3.2.2. Knowledge of relations between parameters

In this paper, 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 inferrer needs a logical dependency graph to conduct an inference (discussed in Section 3.3); the holder needs a logical dependency graph to model the inference procedure of the inferrer in order to evaluate the risk of information leakage caused by inferences (discussed in Section 4).

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

We take an example from the case study in Section 5 to show how to construct a logical dependency graph from a quantitative relation.

$$\textbf{\textit{DryerOutletTemp}} = \frac{\begin{array}{l}\textbf{\textit{DryerInletTemp}} - (r_0 \times \textit{TotalWaterRemo}/ \\ (60 \times \rho_{ng} \times \textit{RegenerationHeatingTime} \\ \quad \times (c_g + c_v \times \textit{WaterContentOfRegenerationGas}/ \\ \quad \rho_{ng} \times 1000000))))\end{array}}{\textbf{\textit{RegenerationFlowRate}}}, \tag{1}$$

Eq. (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; *TotalWaterRemo*, *RegenerationHeatingTime* and *WaterContentOfRegenerationGas* are operating conditions on the manual of the product.

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 *TotalWaterRemo*, *RegenerationHeatingTime*, *WaterContentOfRegenerationGas*, r_0, ρ_{ng}, c_g and c_v in Eq. (1). For this example, $P = \{DryerOutletTemp,\ DryerInletTemp,\ RegenerationFlowRate\}$.

Then we construct the logical dependency relation LD between P and 2^P. Eq. (1) indicates that if we know the actual values of *DryerInletTemp* and *RegenerationFlowRate*, we can calculate the actual value of *DryerOutletTemp*. In other words, if we want to infer parameter *DryerOutletTemp*, we may need to infer parameter *DryerInletTemp* and parameter *RegenerationFlowRate* 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*}. Therefore, for this example, $LD = \{\langle DryerOutletTemp, \{DryerInletTemp, RegenerationFlowRate\}\rangle, \langle DryerInletTemp, \{DryerOutletTemp, RegenerationFlowRate\}\rangle, \langle RegenerationFlowRate, \{DryerOutletTemp, DryerInletTemp\}\rangle\}$.

$$\textbf{\textit{DryerOutletTemp}} = \textbf{\textit{CoolerInletTemp}} + 10 \tag{2}$$

$$\textbf{\textit{DryerInletTemp}} = \textbf{\textit{HeaterOutletTemp}} \times (1 - \textbf{\textit{HeatLossRate}}) \tag{3}$$

A logical dependency graph may describe logical dependencies extracted from multiple quantitative dependencies. Eqs. (2) and (3) are two other quantitative dependencies from the case study in Section 5. We can construct the logical dependency graph with a procedure similar to the previous one. Fig. 3 shows the logical dependency graph extracted from Eqs. (1)–(3). In Fig. 3 and the rest of this paper, 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.

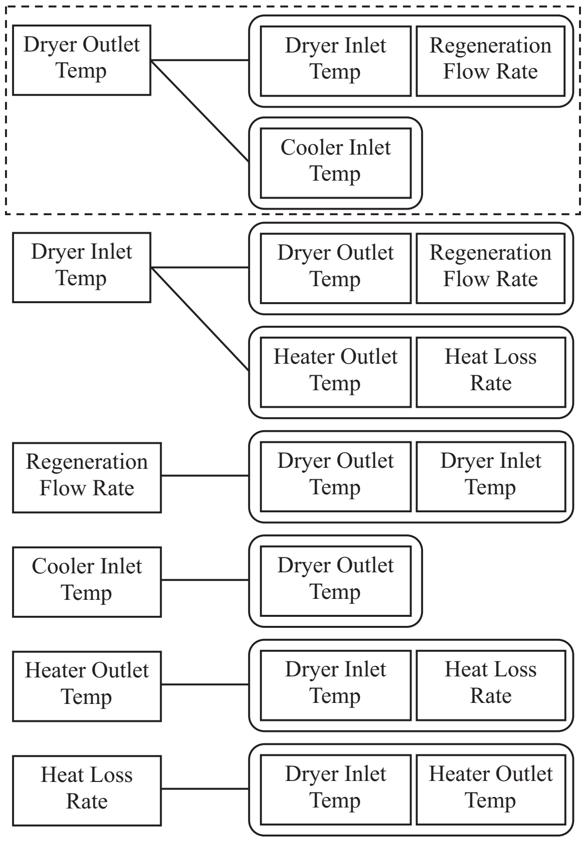


Fig. 3. An example of logical dependency graph.

3.3. Inference

In this section, we will model how the inferrer obtains knowledge through inferences with its "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 Fig. 3 surrounded by a dashed line. We denote the actual value of *DryerOutletTemp*, *DryerInletTemp*, *Regeneration FlowRate* 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 in Eq. (1). The logical dependency between parameter *DryerOutletTemp* and parameter set {*CoolerInletTemp*} is an abstract of the quantitative dependency described in Eq. (2).

(1) If the inferrer knows v_1, v_2 and v_3, it can calculate the value of *DryerOutletTemp* in either Eq. (1) or Eq. (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 the inferrer does not know v_1, v_2 and v_3, it can construct probability distribution d_1, d_2 and d_3 for parameter *DryerInletTemp*, *RegenerationFlowRate* and *CoolerInletTemp*, respectively, from its knowledge; and then it can calculate the probability distribution of *DryerOutletTemp* in either Eq. (1) or Eq. (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

Table 1
Notations and their meanings in Algorithms 1–3.

Notations	Meanings
G	A logical dependency graph
P	A set of parameters
p	A parameter
$\langle p_0, P_j\rangle$	A logical dependency between parameter p_0 and parameter set P_j
LD	A set of logical dependencies
2^P	The power set of P
$\lvert S\rvert$	The cardinality of set S
d	A probability distribution
r	A range
i, j, k, l, m	Non-negative natural numbers

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 calculating the probability distribution of a parameter that an inferrer can obtain through inferences.

Table 1 lists main notations in Algorithm 1 and their meanings. These notations are also used in other algorithms in this paper.

For parameter p_0, the inferrer can construct a probability distribution d_0^0 by its initial knowledge. Algorithm 1 tries to exclude impossible values from the range of d_0^0 so that the inferrer may have a higher possibility to acquire the actual value or a working value of parameter p_0. The following gives a line-byline discussion of this algorithm.

Algorithm 1. An algorithm for calculating the probability distribution of a parameter that an inferrer can obtain through inferences.

```
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 $0 \le i \le |P| - 1$, its initial probability distribution $d_i^0$;
Output: probability distribution $d_0$ of $p_0$;
1:  $LD' \Leftarrow \{\langle p_0, P_j\rangle | P_j \in 2^P, \langle p_0, P_j\rangle \in LD$ and $j \ge 0\}$;
2:  if $LD'$ is empty then
3:      $d_0 \Leftarrow d_0^0$;
4:  else
5:      for all $\langle p_0, P_j\rangle \in LD'$ and $0 \le j \le |LD'| - 1$ do
6:          for all $p_{jk} \in P_j$ and $0 \le k \le |P_j| - 1$;
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 $\langle p_0, P_j\rangle$, $d_{j0}$, $d_{j1}$, ..., and $d_{j(|P_j|-1)}$;
10:     end for
11:     for all $0 \le l \le |LD'|$ do
12:         $r_0^l \Leftarrow$ the range of $d_0^l$;
13:     end for
14:     $r_0 \Leftarrow r_0^0 \cap r_0^1 \cap \ldots \cap r_0^{|LD'|}$;
15:     if $r_0$ is empty then
16:         $d_0 \Leftarrow d_0^0$;
17:     else
18:         for all $0 \le m \le |LD'|$ 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
```

(1) At line 1, LD' is assigned with a set of logical dependencies in logical dependency graph G that are directly relevant to parameter p_0.
(2) Line 2 judges if LD' is empty or not.
(3) If LD' is empty, it means there is no other parameter that has logical dependency with parameter p_0. In that case, the algorithm cannot do any further inference. Therefore, the algorithm returns the initial probability distribution of parameter p_0, d_0^0, at line 3.
(4) If LD' is not empty, the algorithm executes the block from line 5 to line 23.
(5) Lines 5 and line 10 constitute a loop for all logical dependencies in LD'. For each logical dependency $\langle p_0, P_j\rangle$ in LD', a probability distribution d_0^{j+1} of parameter p_0 is inferred with the concrete dependency behind logical dependency $\langle p_0, P_j\rangle$ and probability distributions of parameters in P_j.
(6) For each parameter p_{jk} in P_j, its probability distribution d_{jk} is assigned with its initial probability distribution p_{jk} at line 7.
(7) The inference at line 9 depends on the type of the concrete dependency behind $\langle p_0, P_j\rangle$. For example, the inference may be totally different for a concrete dependency described in algebra and for a dependency described in logic.
(8) If the cardinality of LD' is $|LD'|$, the inferrer will have $|LD'| + 1$ probability distributions of parameter p_0 after the block from line 5 to line 10, and these probability distributions may be not identical. The algorithm tries to comprehend $|LD'| + 1$ probability distributions into one probability distribution of parameter p_0 from line 11 to line 23.
(9) For each probability distribution d_0^l of parameter p_0, its range is assigned to r_0^l at line 12.
(10) At line 14, r_0 is assigned with the intersection of all of these ranges.
(11) 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, the algorithm will return the initial probability distribution of parameter p_0, d_0^0, at line 16.
(12) At line 19, every probability distribution of parameter p_0 is narrowed down to the common range r_0.
(13) Every probability distribution of parameter p_0 obtained at line 19 is normalized at line 20. 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 $f_0''(x)$,

$$\forall x \in r_0, \quad f_0''(x) = \frac{f_0'(x)}{\sum_{y \in r_0} f_0'(y)}. \tag{4}$$

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 $f_0''(x)$,

$$\forall v_1 \le x \le v_2, \quad f_0''(x) = \frac{f_0'(x)}{\int_{v_1}^{v_2} f_0'(y)\,dy}. \tag{5}$$

(14) At line 22, the inferrer will have $|LD'| + 1$ normalized probability distributions of parameter p_0 on the common range r_0. These probability distributions of parameter p_0 may be different because the inferrer's knowledge of parameters may be not totally accurate and as we discussed before, the inferrer may describe its knowledge of parameters approxi-

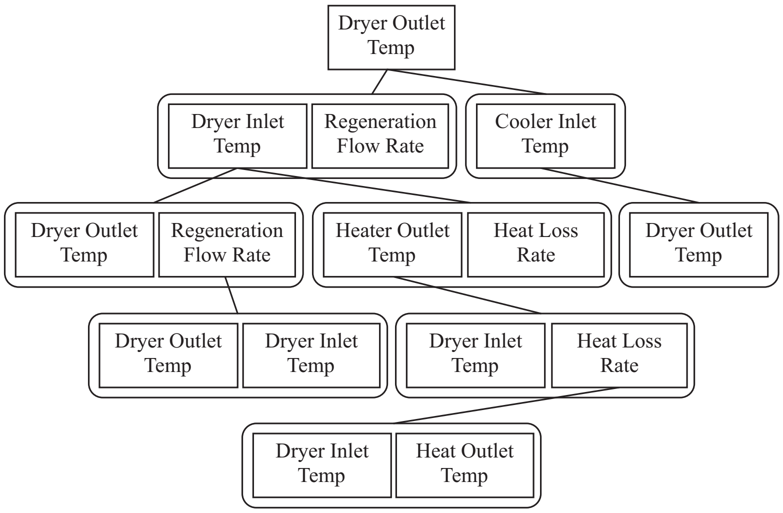


Fig. 4. An example of the tree-like structure constructed with logical dependencies in Fig. 3.

mately with simple or widely used probability distributions by following the physical characteristics of the parameter. There are many strategies for the inferrer 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.

Algorithm 1 does not take full advantage of the logical dependency graph. For example, suppose the inferrer infers the probability distribution of parameter *DryerOutletTemp* with Algorithm 1 and the logical dependency graph is as shown in Fig. 3. 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 $\langle p_i, P_j \rangle$ in the logical dependency graph, add parameter set P_j as a child node of p_i; and then continue the previous step. Fig. 4 is an example of such a tree-like structure, which is constructed with logical dependencies in Fig. 3 and gives partial recursions of inferring the probability distribution of parameter *DryerOutletTemp* with a recursive algorithm.

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 Fig. 4. The multiple occurrences of a parameter during the 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 Fig. 4; one of the two inferences will be unnecessary.

Fig. 5 gives a possible infinite loop when a recursive algorithm infers the probability distribution of parameter *DryerOutletTemp* with logical dependencies in Fig. 3. 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 of parameter *DryerOutletTemp* and we can repeat the previous procedure on the third

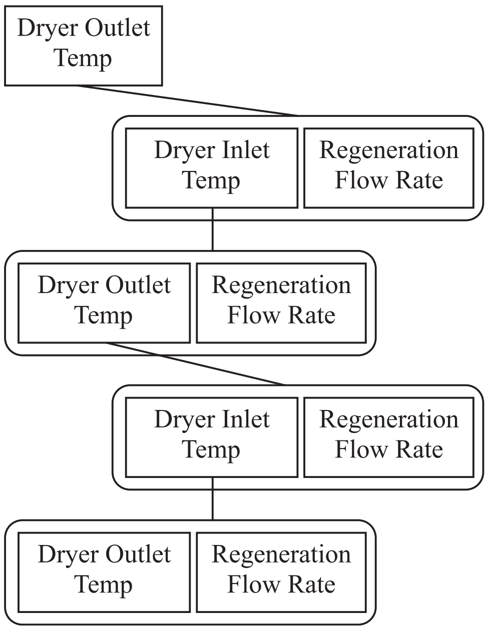


Fig. 5. An example of infinite loop.

occurrence of parameter *DryerOutletTemp* again. Apparently, it will not terminate

Algorithm 2. *rInfer*($p_i, G, d_0, d_1, \ldots, d_{|P|-1}$): a recursive algorithm for calculating the probability distribution of parameter p_i that an inferrer can obtain through inferences.

```
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 $0 \le j \le |P|-1$, 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:     $LD' \Leftarrow \{\langle p_i, P_k\rangle | P_k \in 2^P, \langle p_i, P_k\rangle \in LD, k \ge 0$ and $\langle p_i, P_k\rangle$ is marked with DiacriticalGrave;DiacriticalGrave;unused″; }
4:     if $LD'$ is not empty then
5:       for all $\langle p_i, P_l\rangle \in LD'$ and $0 \le l \le |LD'|-1$ do
6:         mark $\langle p_i, P_l\rangle$ as "used";
7:         for all $p_{lm} \in P_l$ and $0 \le m \le |P_l|-1$ do
8:           $d_{lm} \Leftarrow rInfer(p_{lm}, G, d_0, d_1, \ldots, d_{|P|-1})$;
9:         end for
1:0        $d_i^l \Leftarrow$ the probability distribution of $p_i$ inferred with $\langle p_i, P_l\rangle$, $d_{l0}$, $d_{l1}$, and $d_{l(|P_l|-1)}$;
11:      end for
12:      for all $0 \le n \le |LD'|-1$ do
13:        $r_i^n \Leftarrow$ the range of $d_i^n$;
14:      end for
15:      $r_i \Leftarrow r_i \cap r_i^0 \cap \ldots \cap r_i^{|LD'|-1}$;
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 $0 \le h \le |LD'|-1$ 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 $d_i$, $d_i^0, \ldots, d_i^{|LD'|-1}$;
24:      end if
25:    end if
26:  end if
```

Algorithm 2 is a recursive algorithm for inferring the probability distribution of a parameter, which implements recursive inference 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. □

The following explains other aspects of Algorithm 2. To avoid repeating the descriptions of Algorithm 1, the explanation will focus on the differences of Algorithm 2 from Algorithm 1.

(1) 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".
(2) 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, ..., $d_{|P|-1}$), where *rInfer* is its name and p_i, G, d_0, d_1, ... and $d_{|P|-1}$ are its inputs.
(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.
(4) Line 2 and line 4 are used to control the recursion. As discussed in the proof of Proposition 1, if the conditions in line 2 and line 4 are labeled as A and B respectively, the stopping criterion for the algorithm will be $\neg A \vee (A \wedge \neg B)$.

When applying Algorithm 2 in inferring the probability distribution of parameter *DryerOutletTemp* with logical dependencies in Fig. 3, the recursions are the same as shown in Fig. 4.

4. Risk evaluation

4.1. Evaluation

We have modeled the inferrer's knowledge and its inferences of the private parameter p_0 from the perspective of the holder. By such inferences, the inferrer will get a probability distribution of the private parameter p_0. What the holder concerns is what risk of information leakage it will take if the inferrer gets such a probability distribution of the private parameter p_0 by inferences. In this section, we evaluate the risk within two dimensions: the probability of information leakage caused by inferences and the consequences of such information leakage.

Definition 2 (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 gets 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 $f_p(x)$,

$$P(p) = \frac{\sum_{v_i \in r_p \text{ and } v_1 \le v_i \le v_2} f_p(v_i)}{\sum_{v_j \in r_p} f_p(v_j)}; \qquad (6)$$

If d_p is continuous, its range is $[r_3, r_4]$, and its probability density function is $f_p(x)$,

$$P(p) = \frac{\int_{\max\{v_1, v_3\}}^{\min\{v_2, v_4\}} f_p(x)\,dx}{\int_{v_3}^{v_4} f_p(x)\,dx.} \qquad (7)$$

Definition 3 (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; a logical dependency graph $G = (P, 2^P, LD)$ and $p \in P$; for $\forall\, p_i \in P$ and $0 \le i \le |P| - 1$, its initial probability distribution d_{p_i}; a parameter set P_s, in which all parameters are shared with the inferrer; $\forall\, p_j \in P_s$ and $0 \le j \le |P_s| - 1$, 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 $d_p$ in Eqs. (6) and (7);
```

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 has its initial knowledge k_0, which 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* of the private parameter p.

Algorithm 3 gives an algorithm for calculating conditional probability of information leakage $P(p|\{(p_j, d_{p_j})\})$. The following is an explanation of Algorithm 3.

(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. From line 1 to line 7, the algorithm tries to comprehend two probability distributions into one for each parameter p_j in parameter set P_s.
(2) At line 4, d_{p_i} is assigned with a comprehension of the two probability distributions. The inferrer may have many strategies to do such a comprehension. A natural and simple strategy is "trusting the latest information only", which means the inferrer will ignore d_{p_i} and use d_{p_j} in the inference.
(3) At line 8, it infers the probability distribution d_p of parameter p with Algorithm 2.
(4) At line 9, it calculates conditional probability of information leakage $P(p|\{(p_j, d_{p_j})\})$ with d_p and $[v_1, v_2]$ in Eqs. (6) and (7).

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 $P(p|\{(p_j, d_{p_j})\}) \ge t_0$, 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 $P(p|P_s) \ge t_0$, 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 4 (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 $P_2 \nleftrightarrow p$ is true.

Definition 5 (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 6 (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

$$R(p|\{(p_j, d_{p_j})\}) = P(p|\{(p_j, d_{p_j})\}) \times C(p), \tag{8}$$

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.

$$C(p) = \begin{cases} 1 & \text{if } p \text{ is confidential} \\ 0 & \text{if } p \text{ is not confidential} \end{cases} \tag{9}$$

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 the mitigation of information leakage risk

According to Definition 6 and Eq. (8), the risk of information leakage $R(p|\{(p_j, d_{p_j})\})$ is equal to the probability of information leakage $P(p|\{(p_j, d_{p_j})\})$ times the consequences of information leakage $C(p)$. Therefore, the risk of information leakage caused by inferences may be mitigated by reducing the probability and/or the consequences of information leakage. In this paper, 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 values with the range of d_{p_i}, the new probability of

Table 2
Multiple settings of the "Blower".

Power	ΔP	Power	ΔP
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

information leakage caused by inferences may be lower than $P(p|P_s)$.

In some cases, parameters may be shared as probability distributions. For example, "Blower" is a component of the natural gas dryer discussed as a case study in Section 5. "*power*" and "*PresssureDifferential* (ΔP)" are two design parameters of the "Blower". The manufacturer of the natural gas dryer may order a blower that can work under multiple settings as shown in Table 2 from a supplier. For the supplier, parameter "*Power*" may have two possible values, 1.5 and 2.1; parameter "ΔP" 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.

5. Case study

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.

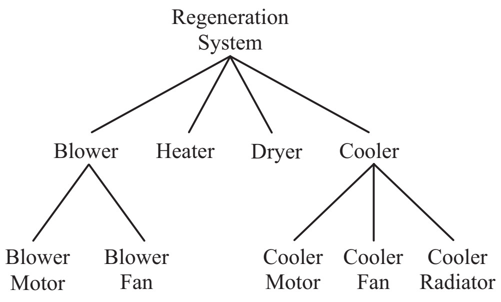


Fig. 6. A product structure tree of the regeneration system.

The regeneration system consists of four major components: blower, heater, dryer and cooler. Fig. 6 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; 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.

In our case study, the manufacturer is the holder; a supplier and potential competitor is the inferrer; design parameter "Dryer Outlet Temp" is the private parameter p_0; the logical dependency graph is shown in Fig. 7.

We developed a software prototype to calculate the probability of information leakage caused by inferences. Fig. 8 is a screenshot of the software prototype. It consists of the following modules:

(1) Setting the inferrer's initial knowledge of parameters. Now the software prototype supports three types of probability distributions, namely discrete distributions, continuous uniform distributions and normal distributions.
(2) Setting knowledge of shared parameters with the inferrer. 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.
(3) Displaying the logical dependency graph and the tree-like-structure.
(4) Calculating the probability distribution of the private parameter p_0 with numerical computation, which is an implementation of Algorithm 2.
(5) Calculating the probability of information leakage caused by inferences, which is an implementation of Algorithm 3.

In supply chains, what design parameters the manufacturer share with a supplier usually depends on what components the supplier supplies. Table 3 gives the relation between components and shared design parameters used in our case study.

We calculated the probabilities of information leakage caused by inferences with the help of the software prototype. Fig. 9 gives the tree-like-structure that shows the recursions. Table 4 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 $0.7 \times actual$ to $1.3 \times actual$, parameters are shared with their

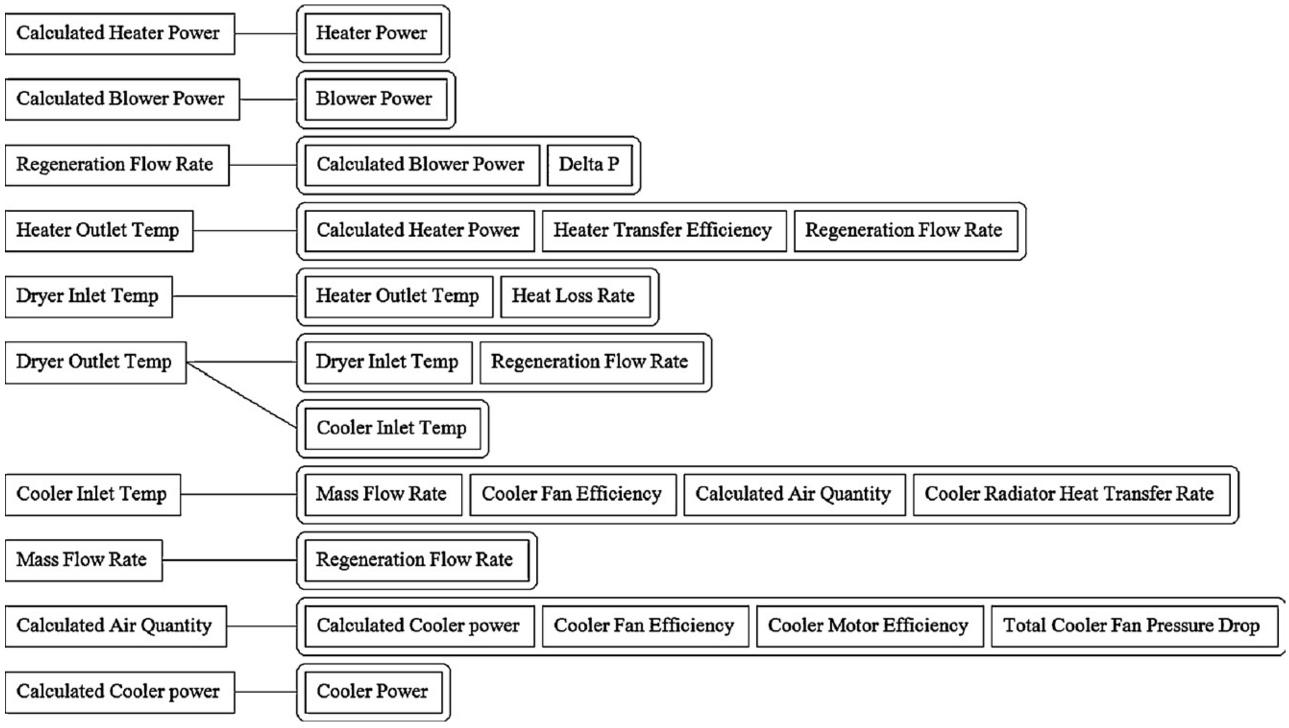
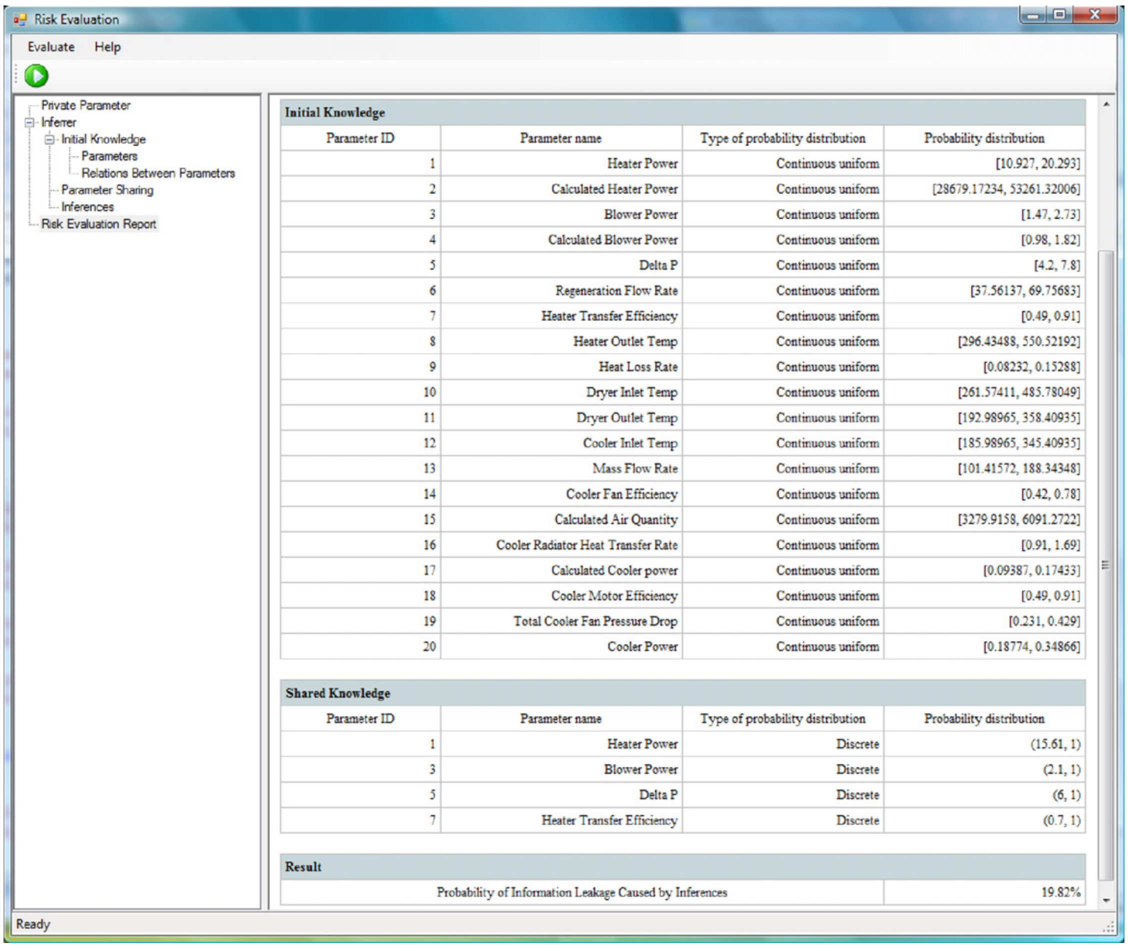


Fig. 7. The logical dependency graph in the case study.



Initial Knowledge

Parameter ID	Parameter name	Type of probability distribution	Probability distribution
1	Heater Power	Continuous uniform	[10.927, 20.293]
2	Calculated Heater Power	Continuous uniform	[28679.17234, 53261.32006]
3	Blower Power	Continuous uniform	[1.47, 2.73]
4	Calculated Blower Power	Continuous uniform	[0.98, 1.82]
5	Delta P	Continuous uniform	[4.2, 7.8]
6	Regeneration Flow Rate	Continuous uniform	[37.56137, 69.75683]
7	Heater Transfer Efficiency	Continuous uniform	[0.49, 0.91]
8	Heater Outlet Temp	Continuous uniform	[296.43488, 550.52192]
9	Heat Loss Rate	Continuous uniform	[0.08232, 0.15288]
10	Dryer Inlet Temp	Continuous uniform	[261.57411, 485.78049]
11	Dryer Outlet Temp	Continuous uniform	[192.98965, 358.40935]
12	Cooler Inlet Temp	Continuous uniform	[185.98965, 345.40935]
13	Mass Flow Rate	Continuous uniform	[101.41572, 188.34348]
14	Cooler Fan Efficiency	Continuous uniform	[0.42, 0.78]
15	Calculated Air Quantity	Continuous uniform	[3279.9158, 6091.2722]
16	Cooler Radiator Heat Transfer Rate	Continuous uniform	[0.91, 1.69]
17	Calculated Cooler power	Continuous uniform	[0.09387, 0.17433]
18	Cooler Motor Efficiency	Continuous uniform	[0.49, 0.91]
19	Total Cooler Fan Pressure Drop	Continuous uniform	[0.231, 0.429]
20	Cooler Power	Continuous uniform	[0.18774, 0.34866]

Shared Knowledge

Parameter ID	Parameter name	Type of probability distribution	Probability distribution
1	Heater Power	Discrete	(15.61, 1)
3	Blower Power	Discrete	(2.1, 1)
5	Delta P	Discrete	(6, 1)
7	Heater Transfer Efficiency	Discrete	(0.7, 1)

Result

Probability of Information Leakage Caused by Inferences	19.82%

Fig. 8. A screenshot of the prototype.

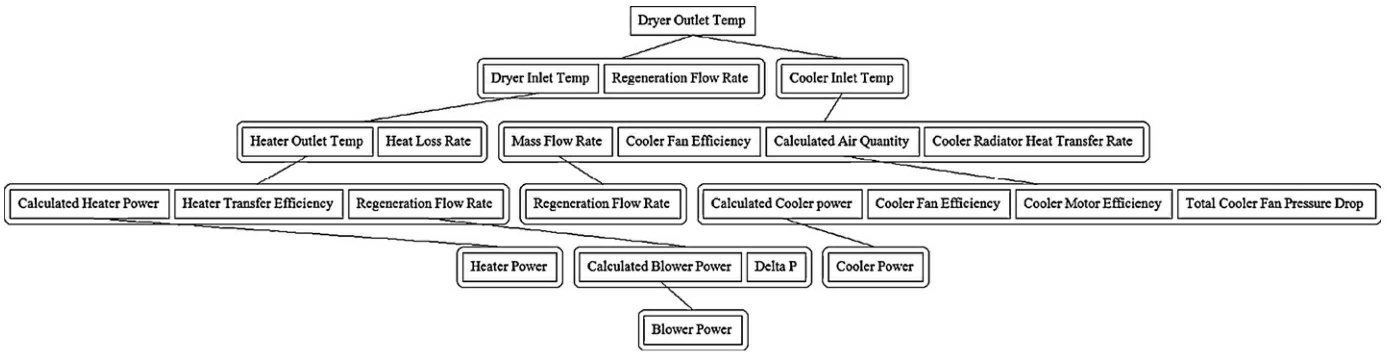


Fig. 9. The recursions in the case study.

Table 3
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

actual values, and the working values of the private parameter p_0 are with the range of from $0.99 \times \textit{actual}$ to $1.01 \times \textit{actual}$. There are some interesting points in the 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.
(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 "Blower" and "Heater", the probability will be 19.82%; if the inferrer supplies "Blower Motor" and "Heater", the probability will be decreased to 6%.

6. Conclusions and future work

In this paper, 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 our major conclusions:

Table 4
Components the inferrer 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%

(1) We modeled the inferrer's knowledge with probability distributions and logical dependency graph. One remained problem 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.
(2) We discussed three mitigation approaches in principle. 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.
(3) We took a product in process industry as our case study. In the future research, we will explore products in other industries, and validate our methods in other real world applications.

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.

Acknowledgements

The research reported in this paper is partially supported by by NSERC through a CRD project (PJ 350114-06). We are grateful to the financial support from NSERC, CRIAQ, Pratt & Whitney Canada Corp., Bombardier Inc., CAE Inc., CMC Electronics Inc., and Rolls-Royce Canada Limited.

References

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

[2] G.Q. Huang, J.S.K. Lau, 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) (2003) 1483–1517.
[3] P. Fiala, Information sharing in supply chains, Omega 33 (5) (2005) 419–423.
[4] H. Zhou, W.C. Benton Jr., Supply chain practice and information sharing, Journal of Operations Management 25 (6) (2007) 1348–1365.
[5] H.L. Lee, V. Padmanabhan, S. Whang, Information distortion in a supply chain: the bullwhip effect, Management Science 43 (4) (1997) 546–558.
[6] T. Moyaux, B. Chaib-draa, S. D'Amours, Information sharing as a coordination mechanism for reducing the bullwhip effect in a supply chain, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 37 (3) (2007) 396–409.
[7] S. Zhang, W. Shen, H. Ghenniwa, A review of internet-based product information sharing and visualization, Computers in Industry 54 (1) (2004) 1–15.
[8] W.D. Li, Z.M. Qiu, State-of-the-art technologies and methodologies for collaborative product development systems, International Journal of Production Research 44 (13) (2006) 2525–2559.
[9] H.L. Lee, S. Whang, Information sharing in a supply chain, International Journal of Manufacturing Technology and Management 1 (1) (2000) 79–93.
[10] L. Li, Information sharing in a supply chain with horizontal competition, Management Science 48 (9) (2002) 1196–1212.
[11] A. Hoecht, P. Trott, Outsourcing, information leakage and the risk of losing technology-based competencies, European Business Review 18 (5) (2006) 395–412.
[12] A. Hoecht, P. Trott, Innovation risks of strategic outsourcing, Technovation 26 (5–6) (2006) 672–681.
[13] K.S. Anand, M. Goyal, Strategic information management under leakage in a supply chain, Management Science 55 (3) (2009) 438–452.
[14] L.C. Giunipero, R.A. Eltantawy, Securing the upstream supply chain: a risk management approach, International Journal of Physical Distribution & Logistics Management 34 (9) (2004) 698–713.
[15] U. Juttner, Supply chain risk management: understanding the business requirements from a practitioner perspective, The International Journal of Logistics Management 16 (1) (2005) 120–141.
[16] A. Lockamy, K. McCormack, Analysing risks in supply networks to facilitate outsourcing decisions, International Journal of Production Research 48 (2) (2010) 593–611.
[17] H. Zhang, Vertical information exchange in a supply chain with duopoly retailers, Production and Operations Management 11 (4) (2002) 531–546.
[18] K.K. Leong, K.M. Yu, W.B. Lee, A security model for distributed product data management system, Computers in Industry 50 (2) (2003) 179–193.
[19] C.D. Cera, T. Kim, J. Han, W.C. Regli, Role-based viewing envelopes for information protection in collaborative modeling, Computer-Aided Design 36 (9) (2004) 873–886.
[20] C.D. Cera, I. Braude, T. Kim, J. Han, 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) (2006) 2–10.
[21] T. Kim, C.D. Cera, W.C. Regli, H. Choo, J. Han, Multi-level modeling and access control for data sharing in collaborative design, Advanced Engineering Informatics 20 (1) (2006) 47–57.
[22] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, C.E. Youman, Role-based access control models, IEEE Computer 29 (2) (1996) 38–47.
[23] D.F. Ferraiolo, R. Kuhn, R.S. Sandhu, RBAC standard rationale: comments on a critique of the ANSI standard on role based access control, IEEE Security & Privacy 5 (6) (2007) 51–53.
[24] Y. Wang, P.N. Ajoku, J.C. Brustoloni, 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) (2006) 149–159.
[25] T.-Y. Chen, Y.-M. Chen, 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) (2008) 565–579.
[26] L. Sweeney, Achieving k-anonymity privacy protection using generalization and suppression, International Journal on Uncertainty, Fuzziness and Knowledge-based Systems 10 (5) (2002) 571–588.
[27] N. Shyamsundar, R. Gadh, Internet-based collaborative product design with assembly features and virtual design spaces, Computer-Aided Design 33 (9) (2001) 637–651.
[28] N. Shyamsundar, R. Gadh, Collaborative virtual prototyping of product assemblies over the Internet, Computer-Aided Design 34 (10) (2002) 755–768.
[29] L. Chen, Z. Song, L. Feng, Internet-enabled real-time collaborative assembly modeling via an e-assembly system: status and promise, Computer-Aided Design 36 (9) (2004) 835–847.
[30] K.-Y. Kim, Y. Wang, O.S. Muogboh, B.O. Nnaji, Design formalism for collaborative assembly design, Computer-Aided Design 36 (9) (2004) 849–871.
[31] D. Mun, J. Hwang, S. Han, Protection of intellectual property based on a skeleton model in product design c ollaboration, Computer-Aided Design 41 (9) (2009) 641–648.
[32] A. Yao, How to generate and exchange secrete, in: Proceedings of the 27th Annual Symposium on Foundations of Computer Science, 1986, pp. 162–167.
[33] O. Goldreich, S. Micali, A. Wigderson, How to play any mental game, in: Proceedings of the 19th Annual ACM Conference on Theory of Computing, 1987, pp. 218–229.
[34] Y. Lindell, B. Pinkas, Privacy preserving data mining, Journal of Cryptology 15 (3) (2002) 177–206.
[35] M.J. Atallah, H.G. Elmongui, V. Deshpande, L.B. Schwarz, Secure supply-chain protocols, in: 2003 IEEE International Conference on E-Commerce, 2003, 293–302.
[36] N. Adam, J. Worthmann, Security-control methods for statistical databases: a comparative study, ACM Computing Surveys 21 (4) (1989) 515–556.
[37] B.C.M. Fung, K. Wang, R. Chen, P.S. Yu, Privacy-preserving data publishing: a survey on recent developments, ACM Computing Surveys 42 (4) (2010) 14:1–14:53.

Da Yong Zhang is an analyst and programmer at iPerceptions Inc. He worked at the Design Lab of Concordia University, Montreal, Canada as a Master's student and research assistant from January 2008 to April 2010. His research topic at Concordia University was modeling, evaluating and mitigating the risk of information leakage in inter-organization collaborations. He holds a Master of Applied Science in Information Systems Security from Concordia University, Montreal, Canada, a Master of Science in Computer Engineering from Harbin Institute of Technology, Harbin, China and a Bachelor of Science in Computer Engineering from Harbin Institute of Technology, Harbin, China.

Dr. Yong Zeng is a professor in the Concordia Institute for Information Systems Engineering at Concordia University, Montreal, Canada. He is Canada Research Chair in Design Science (2004–2014). He received his B.Eng. degree in structural engineering from the Institute of Logistical Engineering and M.Sc. and Ph.D. degrees in computational mechanics from Dalian University of Technology in 1986, 1989, and 1992, respectively. He received his second Ph.D. degree in design engineering from the University of Calgary in 2001. His research is focused on the modeling and computer support of creative design activities. He and his research group have been approaching the research from philosophical, mathematical, linguistic, neurocognitive, and computational perspectives. His research results, which range from the science of design, requirements engineering, human factors engineering, computer-aided product development, product lifecycle management, to finite element modeling, have been applied to manufacturing industry, pharmaceutical industry, and municipality.

Dr. Lingyu Wang is an associate professor in the Concordia Institute for Information Systems Engineering (CIISE) at Concordia University, Montreal, Quebec, Canada. He received his Ph.D. degree in Information Technology in 2006 from George Mason University. He holds a M.E. from Shanghai Jiao Tong University and a B.E. from Shen Yang Institute of Aeronautic Engineering in China. His research interests include database security, data privacy, vulnerability analysis, intrusion detection, and security metrics. He has co-authored one book, co-edited two books, and co-authored over 60 refereed conference and journal articles.

Hongtao Li is the chief chemical process designer of Xebec Adsorption Co. Engineering Department, Montreal, Canada. He received his B.Eng. degree in Chemical Engineering Institute at East China University of Science & Technology in 1995 and his M.Eng. in Mechanical Engineering at the Concordia University in 2008, Montreal, Canada. His interest lies in chemical process conceptual design, detail design and chemical process computational simulation.

Yuan Feng Geng is studying the Master of Library and Information Science (MLIS) in the School of Information Studies, McGill University, Montreal, Canada. She is also an intern in Air Canada's Environmental Affairs Department for a project of the business information management. She received her B.Eng. degree in Mechanical Engineering from Southeast University in 1994 and her M.Eng. in 2009 from the Concordia Institute for Information Systems Engineering, Concordia University, Canada. Her research interests include product lifecycle management, library systems and the record management.