Protecting Sensitive Business Information while Sharing Serial-Level Data

Protecting Sensitive Business Information while Sharing Serial–Level Data

Davide Zanetti
ETH Zurich, Switzerland
SAP (Switzerland) Inc.
zanettid@inf.ethz.ch

Srdjan Čapkun
Department of Computer Science
ETH Zurich, Switzerland
capkuns@inf.ethz.ch

Abstract

Within supply chains, the adoption of RFID technology promises to have a beneficial impact: by sharing serial–level data, supply chain partners can optimize and automate their operations. However, several partners have expressed concerns on the possible misuses of shared serial–level data; this may act as strong deterrent on data sharing, resulting in a lost of all potential benefits. This paper analyzes the problem of leakage of sensitive business information in scenarios in which business partners share serial–level data, provides a framework for describing correlations between sensitive business information and serial–level data, and presents an architecture called Sensitive Information Leakage Monitor (SILM)*, that, based on defined correlations, detects and prevents leakages of sensitive business information while it enables sharing of serial–level data.*

1. Introduction

In supply chains, sharing information among partners can improve coordination between different supply chain stages. This can lead to several improvements, like increasing productivity [7], a more efficient flow of goods [16], reducing inventory level, and cost savings [25]. The adoption of RFID technology together with solutions as the EPCglobal architecture [8] can lead to additional improvements: considering the unprecedented levels of detailed information that can be gathered by tagging at serial–level (i.e., tagging single goods with unique serial numbers), it would be possible to provide precise information about good locations, characteristics, and inventory level. This can lead to larger automation and can speed up processes (reducing costs). Additionally, new services like anti–counterfeiting [22], real–time tracking, product recall, and dynamic pricing [18] would be enabled.

However, partners are reluctant to easily share information related to observations of serial–level–tagged goods (i.e., serial–level data); they have concerns about the possible misuses of this information [3], and in particular, they fear that serial–level data can be used to infer sensitive business information [14]. That is, they fear that sharing serial–level data can cause a leakage of sensitive business information. Moreover, each partner may know its own sensitive business information, but it might not be able to identify the leakage of this sensitive business information caused by sharing its own serial–level data. This could increase fears of possible (and unidentified) leakages, and consequently, lead to situations in which a partner, to avoid any sensitive business information leakage, does not share any serial–level datum (disregarding the positive impact of information sharing). Contrary, ignoring or underestimating possible leakages through serial–level data could lead to situations in which a partner shares serial–level data to benefit from the positive impact of information sharing, but is also hurt by leakages of sensitive business information.

To overcome those situations, partners first need to identify their own sensitive business information, define the relationships (correlations) between this sensitive business information and serial–level data, and deploy a security architecture that, based on defined correlations, guarantees access only to serial–level data that cannot be used to infer sensitive business information. For this purpose, traditional access control models might not be sufficient: they prevent direct unauthorized accesses to data, but do not prevent the leakage of sensitive business information through authorized accesses. That is, they guarantee access only to selected serial–level data (i.e., those considered individually harmless), but they do not prevent from inferring sensitive business information from those selected serial–level data.

In this paper, we propose a solution for protecting sensitive business information while enabling serial–level data sharing. This solution is composed of a framework for describing correlations between sensitive business information and serial–level data, and of a

security architecture called *Sensitive Information Leakage Monitor (SILM)* that, based on defined correlations, detects and prevents the leakage of sensitive business information for given requested data. SILM extends traditional access controls: beside detecting and preventing direct security violations through a traditional mandatory access control, SILM evaluates if sharing the requested data will cause leakages of sensitive business information, and when necessary, it filters the requested data to prevent these leakages. SILM evaluates at attribute–level (fine–grained), and considers historical information. To guarantee maximal data availability, SILM filters out only those requested serial–level data that will actually cause a leakage. Moreover, SILM is conceived to overcome possible collusion between different data requesters.

In our presented work, we make the following contributions. First, through an EPCglobal–enhanced supply chain scenario, we introduce the problem of sensitive business information leakage when business partners share serial–level data, showing examples of correlations between such sensitive information and serial–level data, and suggesting requirements on how to protect sensitive business information (Section 2). Second, we define the framework for describing correlations between sensitive business information and serial–level data (Section 3). Third, we introduce the security architecture called Sensitive Information Leakage Monitor (Section 4). We conclude the paper in Section 6.

2. Sensitive business information leakage

Generally, sensitive business information is any information that, when disclosed, can be potentially misused, and consequently cause losses. In the business world, such sensitive information is described as the knowledge that can provide competitive advantages, and therefore must be protected [23]. Sensitive business information can be any information that [1, 3, 14]: releases strategic information (e.g., information that can be used during negotiation phases, like volumes, prices, etc.), reconstructs strategic connections (i.e., it reveals strategic relationships of a company, and helps to identify its important partners and channels), uncovers unfair behaviors or inefficiencies (from the point of view of the cheater), and helps to identify distribution channels (i.e., identification of the most important supply routes, or the exact locations of high–value consignments).

In order to introduce the problem of leakage of sensitive business information in scenarios in which business partners share serial–level data, and in particular to show possible correlations between sensitive business information and serial–level data, we consider a three–partner EPCglobal–enhanced supply chain (Figure 1). Following EPCglobal[1] standard and specification, the service enabling serial–level data sharing is the *EPC Information Services* (EPCIS) [9]. EPCIS specification defines standard interfaces to capture and query EPC–related data, i.e., serial–level data also called *EPCIS events*. An EPCIS event describes a specific occurrence in the supply chain; it encapsulates information on which good has been involved in which business process, at what time, and where. An *EPCIS repository* implements the capture interface to receive EPCIS events, stores them in a database, and implements the query interface to make those EPCIS events available to other applications (for both intra– and extra–organization data sharing).

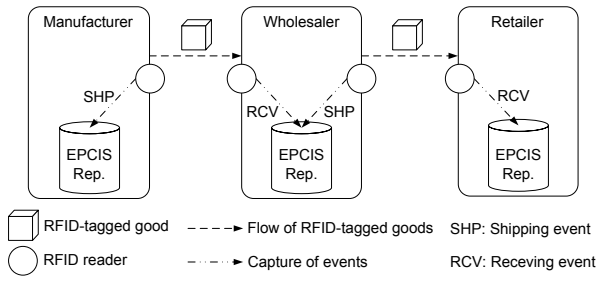

Figure 1. Three–partner supply chain: manufacturer–wholesaler–retailer

Within our scenario, goods flow from the manufacturer to the retailer; the wholesaler acts as broker in the middle. All partners implement the EPCglobal architecture. To increase processes automation and goods recognition, the manufacturer equips all the produced goods with RFID tags, and assigns to each good a unique ID (EPC number). Through RFID–readers, each partner detects RFID–tagged goods, and captures in local EPCIS repositories an event associated to each detection. An event is a tuple that encapsulates three attributes: ID, $STEP$, and $TIME$, that correspond to the different business processes ($STEP$) and relative occurrence time ($TIME$) in which a good (ID) has been involved. In order to maximize the benefits of the implementation of RFID technology (e.g., providing transparency and traceability of goods), each partner allows others[2] to retrieve through the query interface EPCIS events relative to two specific business processes: shipping and receiving.

The wholesaler buys large quantities of goods from one manufacturer, and sells smaller quantities to sev-

[1] http://www.epcglobalinc.org, EPC stands for Electronic Product Code.

[2] Using appropriate authentication and access control mechanisms.

eral retailers. Within its business, it might consider sensitive business information the time that a certain good remains under its control (stocking time), and the delivery volume over a certain period of time. Reasons for considering both stocking time and delivery volume as sensitive information can be the comparison with other wholesalers by the retailers (price vs. goods freshness or price vs. total delivery volume), or by the manufacturer (market penetration and distribution times) during negotiation phase (i.e., in order to use that information to decrease wholesaler's negotiation power). Both stocking time and delivery volume cannot be *explicitly retrieved* from the wholesaler's EPCIS repository, but they can be *inferred* by *combining* EPCIS events retrieved from the wholesaler's EPCIS repository: Examples 1 and 2 show the correlations between the above–mentioned sensitive business information and the shared EPCIS events.

Example 1 (**Stocking time**)**.** The stocking time of a certain good is the time between the instant when that good has been received, and the instant when it has been shipped. Therefore, for a certain good, it is sufficient to retrieve the receiving (RCV) and shipping (SHP) events relative to it. For example, having access to wholesaler's events as listed in Table 1, stocking time for a good with $ID = 100$ can be obtained by combining the events $\#1$ and $\#3$.

Example 2 (**Delivery volume**)**.** The delivery volume over a certain period of time represents the amount of goods that have been shipped, and presumably sold, during that period. This information can be inferred by retrieving and counting all the shipping events occurred during the considered period. For example, having access to wholesaler's events as listed in Table 1, for any good, the delivery volume over 1 hour between 10:30AM and 11:30AM of the 2008-02-01 is equal to 2 (events $\#3$ and $\#4$).

Starting from the presented scenario, it is possible to identify both relevant characteristics of sensitive business information and requirements for protecting such sensitive information:

Protecting information. What is under protection are not implicitly the events stored in a database, but the sensitive business information that could be inferred by combining those events. In Example 1, sensitive business information will leak when sharing both receiving and shipping events relative to a certain good. This means that, for that good, a shipping event can be shared, or a receiving event can be shared, but not both. Therefore, the decision whether to share a certain event is taken considering the sensitive business information that could potentially leak by sharing that event. Thus, we identify two protection strategies: perfect–privacy strategy [5], in which an event is shared only if it does not disclose any information about sensitive business information, and max–sharing strategy, in which an event is shared as far as it does not *actually cause* a leakage of sensitive business information. The former guarantees a strong leakage prevention, but a reduced data availability: in Example 1, not a single receiving nor shipping event would be shared. The latter offers a better data availability (in Example 1, a shipping event could be shared, or a receiving event could be shared, but not both), but it requires additional efforts to guarantee leakage prevention (i.e., history of shared events and anti–collusion solutions). Additionally, other strategies require metrics to quantify both data availability and information leakage; the decision whether to share a certain event is based on an optimal trade–off or threshold limits. An example of this is to use entropy as a metric to evaluate the amount of information contained in different modified datasets, and choose to publish the one that, guaranteing the same privacy/security requirements, provides the biggest amount of information [4].

Table 1. Example of wholesaler's EPCIS events

#	ID	STEP	TIME
1	100	RCV	2008-02-01 09AM
2	101	RCV	2008-02-01 09AM
3	100	SHP	2008-02-01 11AM
4	101	SHP	2008-02-01 11AM

A data–dependent, fine–grained evaluation. A security architecture that aims to detect and prevent possible leakages needs an *evaluation process* that indicates if a requested event can cause a leakage. Considering that correlations between sensitive business information and serial–level data are composed of specific events (e.g., shipping events) and relations between event attributes (e.g., two events having the same ID), the evaluation process needs to classify events and identify attribute relations. Therefore, it has to consider data (i.e., it is data–dependent, opposite of data–independent, which considers only queries [2]), and fine–grained information.

Answer to query: cleaning the requested data. A security architecture that aims to detect and

prevent possible leakages needs a *cleaning process* that sanitizes requested events according to evaluation process results. Considering single events, possible cleaning strategies would be to either modify/block single attributes, or consider an event as a monolithic element and block the whole event. Considering requested events, a single query may request more events at once. The accepting/rejecting approach [2] (i.e., considering the requested events as a monolithic element) may not be the optimal solution in terms of data availability: events that will not cause a leakage of (or neither disclose any information about) sensitive business information may be blocked. A selective–filtering approach, in which the shared data are the maximum subset of the requested data (or the most relevant subset) that does not disclose any sensitive business information guarantees higher data availability, but it can be more computationally expensive.

Requester's identity. Sensitivity of information depends on the identity of the owner of the information (i.e., to whom the information refers to), and on the identity of the requester (i.e., who might receive the information). For example, from the wholesaler's point of view, stocking time is sensitive business information if associated with the retailers; the same information might not be sensitive if associated with other partners, e.g., a distribution center. Differently, from the point of view of the retailers, stocking time is not considered sensitive at all. Therefore, each owner needs to both define its own sensitive business information and have the possibility to individually associate it with different data requesters. In the case that no individual associations can be done (e.g., when publicly releasing datasets, or when all requesters are assumed to collude with each others), each owner needs to define its own sensitive business information, and simply associate it collectively.

Requester's history. Events used to infer sensitive business information can be collected singularly (i.e., through different queries), and at different instants in time. So, unless perfect–privacy strategy is adopted, it is not sufficient to consider only the on–going request; the evaluation of leakage of sensitive business information should also then consider historical information, i.e., events previously shared.

Collusion. Events in an EPCIS repository are the direct consequence of specific occurrences in the supply chain. In a EPCIS repository, there is no insertion of events associated to events gathered from other partners. However, it is reasonable to consider that partners, during business analysis, could merge the events that have been collected. Therefore, unless perfect–privacy strategy is adopted, when sharing events, the evaluation of sensitive business information leakage should not only consider the current requester and its history, but also the possible collusion of this requester with other requesters (and therefore, other requesters' history).

3. Assumptions and formal definitions

We consider the evaluation of sensitive business information leakage in a relational database with a single table. A database with multiple tables can be transformed into a universal relation [13].

A local databases D is composed of one relation rel. A_i denotes an attribute in the relation rel. RS is the relation schema of rel, and lists all the attributes A_i. a_i denotes an attribute value from the domain of A_i. $t[A_i]$ denotes the value of attribute A_i of a single tuple t in the relation rel.

An element ele can be a tuple t_i, a set of tuples $T_i = \{t_1, ..., t_n\}$, a set of sets–of–tuples $\{T_1, ..., T_n\}$, and so on.

A restriction r_i represents the permitted values a_i of an attribute A_i within the domain of A_i. r_i can be a constant, a list of constants, a range of values, or refer to another attribute (where attributes domains are compatible). res can be a set of restrictions r_i equal to $R_i = \{r_1, ..., r_n\}$, a set of sets–of–restrictions $\{R_1, ..., R_n\}$, and so on.

Definition 1 (General constraint). *A constraint c_i is a pair $(ele(c_i), res(c_i))$, in which for an attribute $A_k \in ele(c_i)$, the corresponding $r_k \in res(c_i)$ denotes the constraints on the value of a_k.*

A general constraint c_i can refer to different elements: tuple, set of tuples, set of sets–of–tuples, and so on. A n–order constraint c_i^n refers to one particular element with order n as follows: for a tuple $n = 1$, for a set of tuples $n = 2$, for a set of sets–of–tuples $n = 3$, and so on.

Definition 2 (First–order constraint). *A first–order constraint c_i^1 is a pair $(ele(c_i^1), res(c_i^1))$ where $ele(c_i^1) \subset RS$ is a subset of attributes A_k, and $res(c_i^1)$ is a set of restrictions r_k that specifies the allowed value for each attribute $A_k \in ele(c_i^1)$ ($|ele(c_i^1)| = |res(c_i^1)|$)*[3].

Definition 3 (First–order constraint satisfaction). *A tuple t* satisfies *a first–order constraint c_i^1 if, for each*

[3] $|x|$ indicates the cardinality of x.

attribute $A_k \in ele(c_i^1)$, the value $t[A_k]$ respects the correspondent restriction $r_k \in res(c_i^1)$. The function $S(t, c_i^1)$ evaluating the constraint satisfaction for a constraint c_i^1 against a tuple t is defined as:

$$S(t, c_i^1) = \begin{cases} 1, & if \quad \begin{array}{c} \forall A_k \in ele(c_i^1), \\ t[A_k] \in res(c_i^1)[A_k] \end{array} \\ 0, & otherwise \end{cases}$$

where $S(t, c_i^1) = 1$ means that the tuple t satisfies the constraint c_i^1.

Example 3. Let $(ID, STEP, TIME)$ be the relation schema RS of D, $t_1(200, SHP, 11{:}10AM)$ and $t_2(100, SHP, 12{:}10AM)$ two tuples in D, and $c_1^1 = (\{ID, STEP, TIME\}, \{1*, SHP, *\})$ a first–order constraint[4]. The function evaluating the constraint satisfaction for c_1^1 against tuples t_1 and t_2 gives as result $S(t_1, c_1^1) = 0$ and $S(t_2, c_1^1) = 1$ respectively.

Definition 4 (Second–order constraint). *A second–order constraint c_i^2 is a pair $(ele(c_i^2), res(c_i^2))$ where $ele(c_i^2)$ is a set of tuples, and $res(c_i^2)$ is set of sets–of–restrictions ($|ele(c_i^2)| = |res(c_i^2)|$). $ele(c_i^2) = \{t_1, ..., t_n\}$, where each $t_k \in ele(c_i^2)$ is a set of attributes A_p. $res(c_i^2) = \{R_1, ..., R_n\}$, where each $R_k \in res(c_i^2)$ is a set of restrictions r_m that specifies the relations between attributes of the different tuples in $ele(c_i^2)$ ($|t_k| = |R_k|$).*

Definition 5 (Second–order constraint satisfaction). *A set of tuples T* satisfies *a second–order constraint c_i^2 if, for each attribute A_m specified in each tuple t_k in $ele(c_i^2)$, the value $T[t_k][A_m]$ respects the correspondent restriction $R_k[A_m]$ in $res(c_i^2)$. The function $S(T, c_i^2)$ evaluating the constraint satisfaction for a constraint c_i^2 against set of tuples T is defined as:*

$$S(T, c_i^2) = \begin{cases} 1, & if \quad \begin{array}{c} \forall t_k \in ele(c_i^2) \; and \; \forall A_m \in t_k, \\ T[t_k][A_m] \in res(c_i^2)[R_k][A_m] \end{array} \\ 0, & otherwise \end{cases}$$

where $S(T, c_i^2) = 1$ means that the set of tuples T satisfies the constraint c_i^2.

Example 4. Let $(ID, STEP, TIME)$ be the relation schema RS of D, $t_1(200, SHP, 11{:}10AM)$, $t_2(100, SHP, 12{:}10AM)$, and $t_3(100, RCV, 10{:}10AM)$ three tuples in D. A second–order constraint representing the restriction *"two tuples must have the same ID"* can be defined as $c_1^2 = (\{t_i, t_j\}, \{R_i, R_j\}) = (\{\{ID\}, \{ID\}\}, \{\{t_j[ID]\}, \{t_i[ID]\}\})$. The function evaluating the constraint satisfaction for c_1^2 gives as result $S(\{t_1, t_2\}, c_1^2) = 0$ against t_1 and t_2, and $S(\{t_2, t_3\}, c_1^2) = 1$ against t_2 and t_3.

More complex n–order constraints (i.e., where $n > 2$) can be obtained by adapting $ele(c_i^n)$ and $res(c_i^n)$.

Definition 6 (Sensitive business information). *A correlation si between sensitive business information and serial–level data can be represented as a set of constraints (of different orders), where a constraint of order n_1 can nest constraints with order $n < n_1$. si is defined as:*

$$si = \{c_1^q, ..., c_x^t\}$$

where each $c_i^n \in si$ can nest other constraints as follows:

$$c_i^n\{c_1^{n-1}\{c_1^{n-2}, ..., c_p^{n-2}\}, ..., c_k^{n-1}\{c_1^{n-2}, ..., c_o^{n-2}\}\}$$

where $k = |ele(c_i^n)|$, $p = |ele(c_1^{n-1})|$, and $o = |ele(c_k^{n-1})|$.

Example 5. Let's consider the sensitive business information as described in Section 2: stocking time of a certain good can be computed from a shipping event and a receiving event related to that good. Therefore, the correlation si_{ST} between the sensitive business information stocking time and serial–level data is composed of a tuple $t_1(*, RCV, *)$ and a tuple $t_2(*, SHP, *)$ having $t_1[ID] = t_2[ID]$. si_{ST} can be represented as $si_{ST} = c_1^2\{c_1^1, c_2^1\}$ where $c_1^1 = (t_1, R_{1,1}) = (\{STEP\}, \{RCV\})$ and $c_2^1 = (t_2, R_{2,1}) = (\{STEP\}, \{SHP\})$ are the two first–order constraints needed to specify t_1 and t_2 respectively, and $c_1^2 = (\{t_1, t_2\}, \{R_{1,2}, R_{2,2}\}) = (\{\{ID\}, \{ID\}\}, \{\{t_2[ID]\}, \{t_1[ID]\}\})$ is the second–order constraint needed to specify the relation between t_1 and t_2.

Definition 7 (Sensitive business information leakage). *An element ele causes a leakage of sensitive business information when it satisfies all the constraints c_i^n composing the correlation si representing that sensitive business information. The function $L(ele, si)$ which evaluates the leakage of sensitive business information with respect to its correlation si and an element ele is defined as:*

$$L(ele, si) = \begin{cases} 1, & if \;\; \forall c_i^n \in si, S(ele, c_i^n) = 1 \\ 0, & otherwise \end{cases}$$

where $L(ele, si) = 1$ means that the element ele causes a leakage of the sensitive business information represented by the correlation si.

[4] Asterisk is used as "any value": $(TIME = *)$ means "any value within the domain of $TIME$", $(ID = 1*)$ means "any value within the domain of ID that has the first digit equal to 1".

Example 6. From Example 5: the sensitive business information represented by the correlation si_{ST} leaks if it exists a set of tuples T such that $L(T, si_{ST}) = 1$. Since $si_{ST} = c_1^2\{c_1^1, c_2^1\}$, the considered sensitive business information leaks if $S(\{t_k, t_p\} \in T, c_1^2) = 1$, $S(t_k \in T, c_1^1) = 1$, and $S(t_p \in T, c_2^1) = 1$. Considering the following three sets of tuples:

$T_1 = \{(100, RCV, 11{:}10AM), (100, STO, 12{:}10AM)\}$
$T_2 = \{(100, RCV, 11{:}10AM), (200, SHP, 12{:}10AM)\}$
$T_3 = \{(100, RCV, 11{:}10AM), (100, SHP, 12{:}10AM)\}$

Sets T_1 and T_2 will not cause a leakage, since $S(t_2 \in T_1, c_2^1) = 0$ and $S(T_2, c_1^2) = 0$ respectively. Set T_3 will cause a leakage, since $S(\{t_1, t_2\} \in T_3, c_1^2) = 1$, $S(t_1 \in T_3, c_1^1) = 1$, and $S(t_2 \in T_3, c_2^1) = 1$.

Definition 8 (Query - Answer). *A query Q_i is equivalent to a first–order constraint c_i^1 as introduced in Definition 2: it is a pair $(ele(c_i^1), res(c_i^1))$, where $ele(c_i^1) \subset RS$ is a subset of attributes A_k, and $res(c_i^1)$ is a set of restrictions r_k that specifies the allowed values for each attribute $A_k \in ele(c_i^1)$.*

An answer T_i of a query Q_i over a database D equal to $T_i = Q_i(D) = \{t_1, ..., t_n\}$ is a set of tuples where each tuple $t_k \in T_i$ satisfies the first–order constraint c_i^1 equivalent to the query Q_i.

4. Sensitive Information Leakage Monitor

The Sensitive Information Leakage Monitor (SILM, Figure 2) detects and prevents both direct security violation and leakage of sensitive business information. The former can be obtained through a traditional Mandatory Access Control (MAC)[5], while the latter through a mechanism which we call the Leakage Access Control (LAC). Given (a) a set of users $U = \{u_1, ..., u_p\}$ in which a user $u_k \in U$ can be authenticated by any other user $u_q \in U$, (b) two users u_l and u_i, both members of U, (c) a u_l's set of correlations SI, (d) u_l's database D, (e) a query Q_i from user u_i to user u_l, and (f) the requested data $T_i = \{t_1, ..., t_n\} \subset D$ satisfying query Q_i, SILM performs as follows: triggered by the query Q_i, (i) it invokes the MAC, which evaluates and prevents direct security violations according to its defined method (e.g., through query rewriting techniques [11, 20] or directly operating on T_i to prevent such violations). Then, (ii) it invokes the LAC that, based on defined SI, evaluates (through the evaluation process) the leakages of sensitive business information, and if necessary, it additionally operates on T_i (through the cleaning process) to prevent these leakages. Finally, (iii) it answers to the user u_i with the secure version of T_i.

[5] The definition of a MAC mechanism is beyond the scope of this paper.

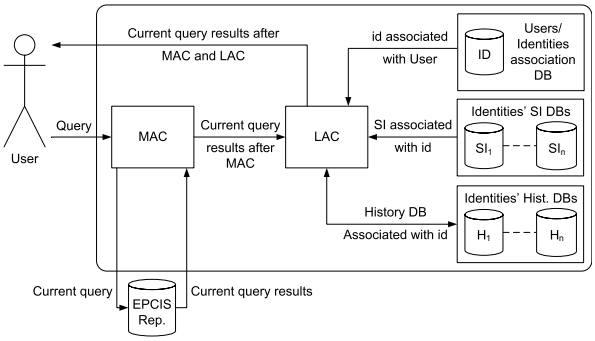

Figure 2. Components of the Sensitive Information Leakage Monitor (SILM)

According to requirements for protecting sensitive business information as described in Section 2, LAC evaluates serial–level data (tuples) at attribute level (data–dependent and fine–grained). Moreover, to increase data availability, the evaluation process follows the max–sharing strategy, in which a tuple is shared as far as it does not actually cause a leakage of sensitive business information. Thus, if sensitive business information can be inferred by combining n different tuples, LAC detects a leakage only when evaluating the n–th tuple, allowing the sharing of $n-1$ among the n tuples. As a consequence of adopting the max–sharing strategy, the evaluation process needs to consider historical information and possible collusion between data requesters (through history databases and user–identity associations respectively). To additionally increase data availability, LAC cleaning process deploys the selective–filtering approach, i.e., it considers requested data as composite elements (in opposite of consider them as monolithic elements), and removes from the requested data only those tuples that actually cause a leakage.

LAC needs additional components to perform the above–mentioned features (Figure 2): a database ID, that stores the identity id associated to each user $u_k \in U$, a group of databases SI_{id}, storing the set of correlations SI associated to each identity id (correlations as defined in Definition 6), and a group of databases H_{id}, storing relevant historical information (i.e., relevant shared events) for each identity id. Based on those components, and given a user $u_i \in U$ querying for data $T_i = \{t_1, ..., t_n\}$, LAC performs as follows (Algorithm 1): for the identity id relative to the data requester u_i, the associated history database H_{id} and the associated set of correlations SI_{id} are retrieved. To provide a selective–filtered answer, the evaluation is done separately for each tuple t in the requested data T_i; if

the union of t with the history database H_{id} will cause a leakage of any of the sensitive business information represented in SI_{id}, the tuple t is filtered out from T_i, and consequently not shared. After the evaluation of a single tuple t, the history database H_{id} is updated under certain conditions.

4.1. Evaluation strategy

The possible leakage of sensitive business information to an identity id is evaluated against the correlation si representing the concerned sensitive business information, a requested tuple t, and the history database H_{id}. This operation is represented by the function $L(H_{id} \cup t, si)$ as defined in Definition 7. The evaluation process needs to verify the constraint satisfaction $S(H_{id} \cup t, c_i^k)$ for all the constraints c_i^k composing si; if all the constraints are satisfied by $H_{id} \cup t$, then sharing t will cause the leakage of the concerned sensitive business information. According to Definition 8, a first–order constraint can be represented as a query; by extension, a n–order constraint can be represented by multiple, nested queries. Therefore, a set of tuples T satisfies a constraint c_i^k if the answer to the c_i^k–query–representation $Q_{c_i^k}$ over T is non–null. For a si composed of a set of constraints $\{c_1^m, ..., c_n^p\}$, the evaluation process verifies the answers to all c_i^k–query–representations $Q_{c_i^k}$ over $H_{id} \cup t$. If all are non–null, t will cause a leakage of the concerned sensitive business information. Contrary, if at least one constraint is not satisfied, t is declared secure, and can be shared.

4.2. History databases

To reduce both overheads and the computation time needed for the evaluation (i.e., to reduce the size of history databases), a tuple t that has been evaluated as secure and shared with a user u_i is stored into the history database H_{id} associated to u_i's identity id only if t is relevant to infer any sensitive business information associated with entity id. That is, a tuple t is stored into the history database when it can be used to obtain sensitive business information, but at the time of the evaluation, sharing t will not actually cause a leakage of sensitive business information. For example, a correlation si_1 is composed of t_1 and t_2 such that $si_1 = \{t_1, t_2\}$. If a request for t_1 occurs, and at the time of the request, t_2 was not shared, the evaluation process considers t_1 as secure: t_1 will be shared. Since t_1 is one of the components of si_1, it will be stored into the requester's history database. More formally, a tuple t is stored into a history database H_{id} if it satisfies at least one first–order constraint c_i^1 among all first–order constraints present in each correlation $si_i \in SI_{id}$ (Algorithm 1 - line 9).

Algorithm 1. Leakage Access Control

Input: Authenticated user u_i
Set of tuples T_i
Output: Secure version of the set of tuples T_i

begin
1 find identity id associated with user u_i
2 collect correlations associated with id: $SI_{id} = \{si_1, ..., si_p\}$
3 retrieve history database H_{id} associated with id
// Evaluation for each tuple.
4 **foreach** $t \in T_i$ **do**
// Eval for each correlation.
5 **foreach** $si \in SI_{id}$ **do**
// Eval if $H_{id} \cup t$ causes a leakage of the sensitive info represented by si.
6 **if** $L(H_{id} \cup t, si) = 1$ **then**
// Sharing t causes a leakage. So, clean T_i.
7 remove t from T_i: $T_i = T_i \setminus t$
8 jump to the next $t \in T_i$
end
end
// Updated H_{id} with t, iff at least one 1st-order constraint in SI_{id} is satisfied.
9 $H_{id} = H_{id} \cup t$ iff $\exists(c_i^1 \in SI_{id}) | S(t, ci_i^1) = 1$
end
end

4.3. User/Identity association

To different users, different sensitive business information may be associated, but if a collusion between these users can be assumed, sensitive business information associated to each single user becomes collective, and associated to all colluders. To represent this, and overcome leakages by collusion, an identity is associated to each user. A user has one single associated identity, while the same identity can be associated to more users. Since the evaluation is executed per entity, users sharing the same identity id can be seen as a single requester; this means that they collectively share the same set of correlations SI_{id} and the same history database H_{id}. For example, a correlation si_1 is composed of t_1 and t_2 such that $si_1 = \{t_1, t_2\}$. si_1 is listed in the correlation set SI_{id1} associated to identity $id1$. $id1$ is associated to both users u_1 and u_2. If u_1 re-

quests for t_1, and at the time of the request, t_2 was not shared, the evaluation process considers t_1 as secure: t_1 is shared and stored into history database H_{id1} (since it is a component of si_1). Now, if u_2 requests for t_2, the evaluation process considers its associated identity $id1$, and since t_1 was already shared (it is in H_{id1}), t_2 is not shared with u_2. An identity set $ID_i = \{u_1, ..., u_n\}$ contains all the users sharing the same sensitive business information and historical information. Two identity sets cannot share a common user; in the case that a new collusion is supposed, the sets of colluding users are merged (and consequently, also correlations sets and history databases are merged), creating a new identity and deleting the previous ones.

5. Related work

Confidentiality ensures that certain information is accessible only to those authorized to have access. Inference channels can be used by an user to access information it has no access right, i.e., in multilevel databases, *it is a means by which one can infer data classified at a high level from data classified at a low level* [15]. Inference problem, i.e., how to detect and remove inference channels, has been intensively studied in the past years; a general overview over the inference problem can be found in the work of Farkas and Jajodia [10]. Proposed solutions can be mainly divided into two groups: detect and remove inferences at the database design phase [6, 19, 21], and detect and remove inferences at query time [12, 17, 24]. The former analyzes attributes and relations between attributes; it will not introduce any additional evaluation during query time, but it can lead to over–classification. The latter analyzes queries (data–independent) and query results (data–dependent), and can consider both past shared data and queries; it is more computationally expensive, but it can lead to a better data availability.

Our work extends the traditional notion of data confidentially and inference problem: we consider, and consequently protect, not only data stored in a database, but also the information that can be inferred by combining those data. Similarly to Brodsky et al. [2], our work considers data–dependent analysis, and proposes a security architecture which guarantees data confidentiality based on constraints. Furthermore, we provide a more flexible definition of constraints. Regarding access control for EPCIS, Grummt and Müller [11] propose a fine–grained access control based on query rewriting, but without considering the inference problem. To our knowledge, our work is the first on inference control for EPCIS.

6. Conclusion

Supply chain partners may be willing to share serial–level data to benefit from the numerous advantages that such detailed information can give, but concerns and fears on the misuses of shared data to infer sensitive business information acts as a strong deterrent on data sharing, resulting in a lost of all potential benefits of information sharing. Moreover, partners may not be able to correlate serial–level data with sensitive business information. This could lead to several undesired situations, like those in which data are not shared, or in which data are shared without paying attention on sensitive business information leakage.

Our work overcomes those undesired situations: it protects sensitive business information while it enables serial–level data sharing, which allows partners to benefit from information sharing, and at the same time, to avoid potential losses caused by leakages of sensitive business information. Our solution consists of a framework for describing correlations between sensitive business information and serial–level data, and in a security architecture, SILM, that, based on defined correlations, detects and prevents leakages of sensitive business information in scenarios in which business partners share serial–level data. Through our framework, it is possible to represent elements like tuples, set of tuples, set of sets–of–tuples, and so on, and relations between single tuple attributes within all these elements. Our architecture, SILM, executes a data–dependent, fine–grained sensitive–business–information–leakage evaluation on current requested data and historical information. To increase data availability, SILM deploys both max–sharing strategy and selective–filtering approach. The former detects a leakage only on requested data that actually cause a leakage, while the latter defines the shared data as the maximum subset of the requested data that does not cause a leakage of sensitive business information. Additionally, SILM considers the requesters' identity and deals with possible collusion.

In our future work we will explore (i) different validation methods, (ii) possible optimizations, and (iii) the definition of administrative tasks. The first point refers to the formal proof of effective leakage prevention, the validation of SILM by scalability, performance, and usability, as well as the implementation of SILM within a real–world supply chain scenario (showing implications, deployment on current IT infrastructures, and cost–benefit analysis). The second point includes the definition of metrics for both leakage and data availability, the definition of metrics–based evaluation processes, and the extension of SILM to consider existing knowledge, publicly available data, database up-

dates, and time–validity of constraints (and more in general, optimize for both reducing the time needed for the evaluation and guaranteeing a better data availability). The third point addresses problems like how, and by which means, to define and maintain access policies, and who defines and maintains these policies. This can include adopting standardize access control languages to support policies, designing a user–friendly interface in which to define sensitive business information and policies, and dealing with changes and time–validity of policies.

Acknowledgements

The authors would like to thank Oliver Kasten for suggesting ideas and helpful discussions.

References

[1] M. Aigner, T. Burbridge, A. Dada, J. Farr, A. Ilic, and A. Soppera. BRIDGE: Security analysis report. Technical report, 2007.

[2] A. Brodsky, C. Farkas, and S. Jajodia. Secure databases: Constraints, inference channels, and monitoring disclosures. *IEEE Trans. on Knowl. and Data Eng.*, 2000.

[3] T. Burbridge, V. Broekhuizen, J. Farr, D. Zanetti, E. G. Muñoz, J. J. Cantero, M. Ángel Guijarro, and J. Baños. RFID network confidentiality. BRIDGE deliverable D–4.5.1, 2007.

[4] X. Y. Chen and R. Wei. A scheme for inference problems using rough sets and entropy. In *RSFDGrC (2)*, 2005.

[5] C. Clifton, M. Kantarcioglu, and J. Vaidya. Defining privacy for data mining. In *Proc. of the Nat. Sci. Foundation Workshop on Next Generation Data Mining*, 2002.

[6] H. S. Delugach and T. H. Hinke. Wizard: A database inference analysis and detection system. *IEEE Trans. on Knowl. and Data Eng.*, 8(1), 1996.

[7] J. Dyer and K. Nobeoka. Creating and managing a high-performance knowledge-sharing network: the toyota case. *Strategic Management Journal*, 21(3), 2000.

[8] EPCglobal. Architecture framework v. 1.2. Standard, 2007.

[9] EPCglobal. EPCIS standard v. 1.0.1. Standard, 2007.

[10] C. Farkas and S. Jajodia. The inference problem: a survey. *SIGKDD Explor. Newsl.*, 4(2), 2002.

[11] E. Grummt and M. Müller. Fine-grained access control for EPC Information Services. In *IOT*, 2008.

[12] J. Hale and S. Shenoi. Catalytic inference analysis: Detecting inference threats due to knowledge discovery. In *Proc. of the IEEE Sym. on Sec. and Priv.*, 1997.

[13] T. H. Hinke, H. S. Delugach, and R. P. Wolf. Iliad: an integrated laboratory for inference analysis and detection. In *Proc. of the 9th IFIP TC11 WG11.3 Working Conference on Database Sec.*, 1996.

[14] A. Ilic, F. Michahelles, and E. Fleisch. The dual ownership model: Using organizational relationships for access control in safety supply chains. In *Proc. of the 21st Inter. Conference on Advanced Inf. Networ. and Applic. Workshops*, 2007.

[15] S. Jajodia and C. Meadows. Inference problems in multilevel secure database management systems. In *Information Security: An Integrated Collection of Essays 1st edition*. 1995.

[16] H. L. Lee, V. Padmanabhan, and S. Whang. Information distortion in a supply chain: the bullwhip effect. *Manage. Sci.*, 43(4), 1997.

[17] D. G. Marks, A. Motro, and S. Jajodia. Enhancing the controlled disclosure of sensitive information. In *Proc. of the 4th European Sym. on Res. in Computer Security*, 1996.

[18] K. Pramatari. Collaborative supply chain practices and evolving technological approaches. *Supply Chain Management - An International Journal*, 12(3), 2007.

[19] X. Qian, M. E. Stickel, P. D. Karp, T. F. Lunt, and T. D. Carvey. Detection and elimination of inference channels in multilevel relational database systems. In *Proc. of the 1993 IEEE Sym. on Sec. and Priv.*, 1993.

[20] S. Rizvi, A. Mendelzon, S. Sudarshan, and P. Roy. Extending query rewriting techniques for fine-grained access control. In *Proc. of the 2004 ACM SIGMOD Intern. Conf. on Management of Data*, pages 551–562, 2004.

[21] G. Smith. Modeling security-relevant data semantics. *IEEE Trans. on Software Engineering*, 17(11), 1991.

[22] T. Staake, F. Thiesse, and E. Fleisch. Extending the EPC network: the potential of RFID in anti-counterfeiting. In *Proc. of the 2005 ACM Sym. on Appl. Comp.*, 2005.

[23] E. Turban and J. Aronson. *Decision Support Systems and Intelligent Systems*. Prentice Hall PTR, 1997.

[24] R. Yi and K. Levitt. Data level inference detection in database systems. In *Proc. of the 11th IEEE workshop on Comp. Sec. Found.*, 1998.

[25] Z. Yu, H. Yan, and T. E. Cheng. Benefits of information sharing with supply chain partnerships. *Industrial Management & Data Systems*, 101(3), 2001.