Journal of Biomedical Informatics 50 (2014) 77–94

Contents lists available at ScienceDirect

Journal of Biomedical Informatics

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

δ-Dependency for privacy-preserving XML data publishing

Anders H. Landberg *, Kinh Nguyen, Eric Pardede, J. Wenny Rahayu

Department of Computer Science and Computer Engineering, La Trobe University, Victoria 3086, Australia

ARTICLE INFO

Article history:
Received 25 August 2013
Accepted 28 January 2014
Available online 8 February 2014

Keywords:
Medical privacy
XML
Hierarchy
Dissection
Privacy-preserving healthcare data

ABSTRACT

An ever increasing amount of medical data such as electronic health records, is being collected, stored, shared and managed in large online health information systems and electronic medical record systems (EMR) (Williams et al., 2001; Virtanen, 2009; Huang and Liou, 2007) [1–3]. From such rich collections, data is often published in the form of census and statistical data sets for the purpose of knowledge sharing and enabling medical research. This brings with it an increasing need for protecting individual people privacy, and it becomes an issue of great importance especially when information about patients is exposed to the public.

While the concept of data privacy has been comprehensively studied for relational data, models and algorithms addressing the distinct differences and complex structure of XML data are yet to be explored. Currently, the common compromise method is to convert private XML data into relational data for publication. This ad hoc approach results in significant loss of useful semantic information previously carried in the private XML data. Health data often has very complex structure, which is best expressed in XML. In fact, XML is the standard format for exchanging (e.g. HL7 version 3[1]) and publishing health information. Lack of means to deal directly with data in XML format is inevitably a serious drawback.

In this paper we propose a novel privacy protection model for XML, and an algorithm for implementing this model. We provide general rules, both for transforming a private XML schema into a published XML schema, and for mapping private XML data to the new privacy-protected published XML data. In addition, we propose a new privacy property, δ-dependency, which can be applied to both relational and XML data, and that takes into consideration the hierarchical nature of sensitive data (as opposed to "quasi-identifiers"). Lastly, we provide an implementation of our model, algorithm and privacy property, and perform an experimental analysis, to demonstrate the proposed privacy scheme in practical application.

1. Introduction

Broadly speaking, there are two distinct approaches to privacy protection. One is the *access-control* approach, which seeks to protect privacy by controlling access to the data; and the other is the *privacy preservation* approach, which manipulates the data, by various privacy preservation techniques, before publishing the data for public use. The latter is the subject matter of this paper. Privacy preservation is obviously an important consideration when making medical data available beyond controlled boundaries, such as data exchange between EMR's (Williams et al., 2001; Virtanen, 2009; Huang and Liou, 2007) [1–3], as it may contain comprehensive sensitive information about patients. What these forms of data publishing have in common is that they all provide data collections that associate individuals with information relevant to these individuals. Some of the published information may be sensitive and it should not be possible to link it to a specific individual in the data. In response to this requirement, a large research community has evolved and many approaches have been proposed to protect individuals' privacy in the data.

Numerous approaches for privacy preservation in the relational domain have been proposed, most notably for publishing microdata [4–8] and transactional data [9–12]. But few approaches have been proposed for the XML domain and investigated what issues arise by doing so. Papers that address XML are mostly based on access-control techniques. Regarding access control for XML, it is useful to note that a vast amount of work has been carried out as it poses unique challenges when applying access control specific concepts to the semistructured, hierarchical and complex structure of the XML format [13–19].

* Corresponding author.
E-mail addresses: anders.h.landberg@gmail.com (A.H. Landberg), kinh.nguyen@latrobe.edu.au (K. Nguyen), e.pardede@latrobe.edu.au (E. Pardede), w.rahayu@latrobe.edu.au (J.W. Rahayu).
URLs: http://www.latrobe.edu.au/scitecheng/about/staff/profile?uname=K2Nguyen (K. Nguyen), http://homepage.cs.latrobe.edu.au/ekpardede/ (E. Pardede), http://homepage.cs.latrobe.edu.au/jwrahayu/ (J.W. Rahayu).

[1] http://www.hl7.org.

http://dx.doi.org/10.1016/j.jbi.2014.01.013
1532-0464/

As for privacy preservation through data modification, typically it is assumed that XML can be flattened to relational data and then any conventional privacy scheme can be applied. By "flattening" we mean mapping data from a hierarchical to a relational format. But medical data, in most cases, contain very rich, and therefore complex, semantics. A reading of blood pressure, for example, may depend on where in the body it is measured, which point in time in the cardiac cycle it is taken (systolic and diastolic arterial blood pressures), which instrument is used, and so on. Such rich semantics can be conveniently expressed in XML. Hence, it is no surprise that XML has been selected as the standard format for medical data exchange, as pointed out in the abstract. In light of these facts, the "flattening XML to relational" assumption, from the practical viewpoint, is untenable: the flattening process is labour-intensive and the loss of semantics is significant.

While medical data publishing is typically intended for useful purposes such as research, statistical analysis and data mining, it is inevitable that the data will be shared with the wrong people. Vaidya et. al. define data privacy as *freedom from unauthorised intrusion* [20]. This definition is quite general in nature and leaves it open as to what unauthorised intrusion actually means. In privacy preservation approaches, it is interpreted as a privacy breach where an individual and their associated information can successfully be identified within the data, assuming that unique identifiers have been removed from the data already.

Based on this key objective to protect individuals' personal information in published data, we seek to analyse and investigate key issues in protecting XML data for privacy. In a nutshell, we seek to extend the well-known approach of Anatomy [6], which was proposed for relational data, for direct application to XML data. For relational data, concepts such as QI (quasi-identifier) and SI (sensitive data) are easy to define in theory, and QI and SI data are easy to extract in practice (at least prior to the application of techniques to achieve *k*-anonymity, *l*-diversity, etc.). For XML data, as can be expected, those equivalent tasks present various interesting challenges. This paper will address those arising issues from both the theoretical and practical viewpoints.

In addition, we propose the novel concept of δ-*dependency*. This concept is to deal with the problem of *hierarchical sensitive data*, which arises in situations where sensitive data values are taken from a hierarchical tree structure, with the concepts signified by data values become more specific as we move down the tree. For an exposition of the problem of hierarchical sensitive data in the context of relational data, see Personalized Privacy Preservation [21]. The concept of δ-*dependency* can be applied to both relational and XML data. In this paper, we will present an algorithm to illustrate how δ-*dependency* can be incorporated into the transformation of XML data for privacy preservation. We will also present the results of several experiments to illustrate the effectiveness of δ-*dependency* for privacy preservation purpose.

Our paper is organised as follows. Section 2 makes four separate contributions and proposes new methods as part of our privacy protection approach: Section 2.1 proposes a novel method for the construction of an XML schema for published data (XML schema transformation). Section 2.2 proposes a new method for the production of published data (XML data mapping). Section 2.3 proposes a new privacy property, δ-dependency that can be applied to both relational and XML data, and that takes into consideration the hierarchical nature of data, which is also common to XML. Section 2.4 proposes a new algorithm that implements our privacy model, dissection technique and δ-dependency privacy property. In Section 3, we provide an experimental analysis to benchmark the proposed privacy scheme against a well-proven privacy method (Anatomy) and in practical application. Sections 4 and 5 provide a discussion of the results and draw conclusions.

2. Methods

Anatomy's process of transforming private data to published data can be conceived of as consisting of two major tasks: *dissection* and *grouping*. The purpose of dissection is to *separate QI data from SI data*, and the purpose of grouping is to *achieve the desired privacy property,* for example, *l*-diversity.

In the relational domain, the first task is trivial, especially for microdata tables. Given a private relational schema, all we need to do is to specify which attributes are QI attributes and which are SI attribute, and immediately we know what the published relational schema should look like. As an example, consider the one shown in Fig. 1, which is taken from the Anatomy paper [6]. The first table is the private data, and the last two tables, QIT and SIT, are the result of applying dissection and diversification on the first table. In this example, attributes *Age*, *Sex* and *Zip code* are QI attributes, and *Disease* is an SI attribute. The separation of QI from SI data is simple: once we specify *Age*, *Sex* and *Zip code* as QI attributes and *Disease* as SI attribute, we immediately have the structures for the QIT and SIT tables, and the separation *per se* is effectively done. With hardly doing any work, we can directly proceed to the task of grouping (diversification). Once we have chosen the groups, we can record those groups in an additional column in the first table (if we wish). Then we can simply copy the cell data to form the *QIT* and *SIT* tables. (Note that because dissection is trivial for relational data, it is hardly discussed, which is an appropriate thing to do.)

In contrast, let us now consider the private XML data shown in Fig. 2. Given that we have chosen some of the data items to be QI, and some to be SI, it is no longer obvious how we may proceed from there to perform dissection and then move onto the task of grouping to achieve the desired privacy property. It is also clear that the task of grouping is now also more complicated: we no longer just have SI attributes to contend with.

In the attempt to extend Anatomy to XML data, a number of issues inevitably arise, which will be addressed in this paper, including:

- How to transfer the key concepts of QI and SI to the XML domain?
- How to derive the published XML schema from a given private XML schema?
- How to extract data from the private XML data to produce published XML data?

Tuple ID	Age	Sex	Zipcode	Disease
1 (Bob)	21	M	10001	Dyspepsia
2	27	M	13000	Pneumonia
3	35	M	60000	Flu
4	60	M	12000	Gastritis
5	61	M	10001	Bronchitis
6	70	F	60000	Pneumonia
7	70	F	60000	Gastritis

(a) The microdata

Tuple ID	Age	Sex	Zipcode	Group-ID
1	21	M	10001	1
2	27	M	13000	1
3	35	M	60000	1
4	60	M	12000	1
5	61	M	10001	2
6	70	F	60000	2
7	70	F	60000	2

(b) The quasi-identifier table (QIT)

Group-ID	Disease	Count
1	Dyspepsia	1
1	Flu	1
1	Gastritis	1
1	Pneumonia	1
2	Bronchitis	1
2	Gastritis	1
2	Pneumonia	1

(c) The sensitive table (SIT)

Fig. 1. Privacy using Anatomy in relational microdata.

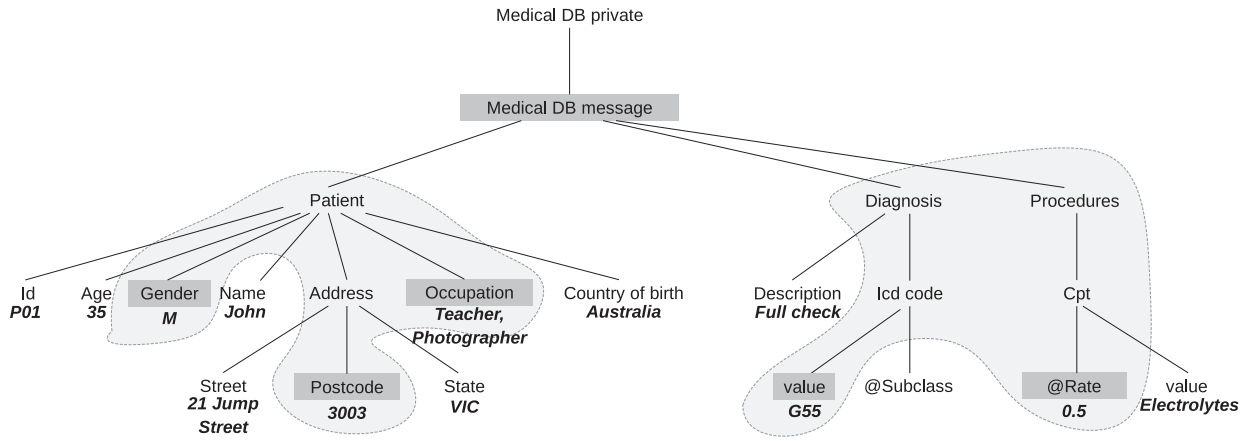

Fig. 2. Medical XML database data (private data).

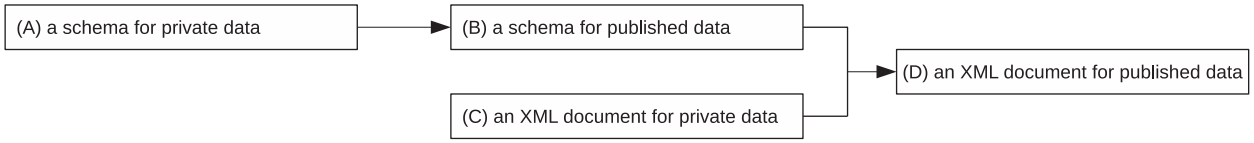

Fig. 3. Proposed methodology.

- How to handle the additional features of XML such as element attributes?
- How to group the QI and SI XML data, given that they are now parts of some hierarchies?

An overview of our proposed methodology is presented in Fig. 3. As shown on the diagram, the methodology consists of two major tasks. The first task is to derive the XML schema for the intended published XML document (we will refer to this schema as the "published XML schema", or simply "published schema") from the XML schema for the private document (we will refer to this schema as the"private XML schema", or simply "private schema"). This task is represented by the arrow going from A to B. The second task is to produce the published data itself, based on the derived published schema. This task is represented by the arrow going from B and C to D.

We will use a running example to illustrate the concepts and procedures of our proposed methodology. The starting private XML document and the resulting published XML document are listed in the online Appendix.[2] Fig. 2 shows a part of the private XML document. As can be seen, the private data contains information about patients: personal details (e.g. age, gender, name, etc.) and medical data concerning various diagnoses (e.g. value for Icd code) and procedures (Cpt rate and value). Fig. 4 shows the private XML schema, Fig. 5 the published XML schema, and Fig. 6 part of the published data.

2.1. Method for constructing the XML schema for published data

Given a private XML document (or a set of XML documents), whose information may be very useful for knowledge-sharing and research purposes, the aim of our methodology is to publish the information as an XML document in some suitable format for public use. We approach this problem by formulating the extension of Anatomy for direct application to XML data. In this endeavour, the first task, which is of central importance, is to determine what the *schema* for published XML document should be.

To facilitate the defining of the published XML schema, and to avoid ambiguities, we present a formal model of the XML schema. This formal model presents an abstract view of XML schema that is relevant to the work presented in this paper. Essentially, we view a schema as a labelled rooted tree, with the additional condition that each node has a property called *occurrences*. "Occurrences" is a term borrowed from the terminology of XML schema. If an element E in an XML schema has multiple occurrences, then in an XML instance (document) of that schema, E can occur more than once in its parent element.

2.1.1. Formal representation of XML schema

Definition 1 (*Formal Representation of XML Schema*).

1. A *tree* is a *connected acyclic* graph.
2. A *rooted tree* is a tree with a node designated as the root. Thus, a rooted tree is a tuple $\langle G, root\rangle$ where $G = \langle Nodes, Edges\rangle$ is an acyclic connected graph and $root \in Nodes$.
3. A *labelled tree* is one in which each node is associated with a label.
4. An *XML schema* is a *labelled rooted tree*, with a function, called *occurrences* that maps each node to a *range* of positive integers (a range has a lower limit and an upper limit). More specifically, we assume that there are two functions available:
   - $name: node \mapsto string$ that maps a node to its name (label), and

[2] http://code.google.com/p/twitter-analytics/source/browse/trunk/XMLPrivacy/onlineappendix.txt.

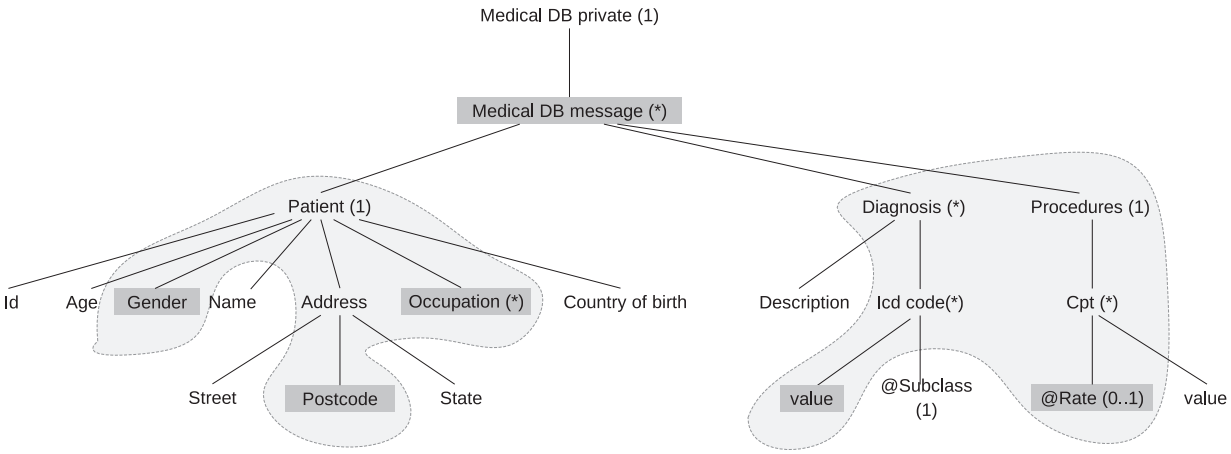

Fig. 4. Medical XML database schema (private schema).

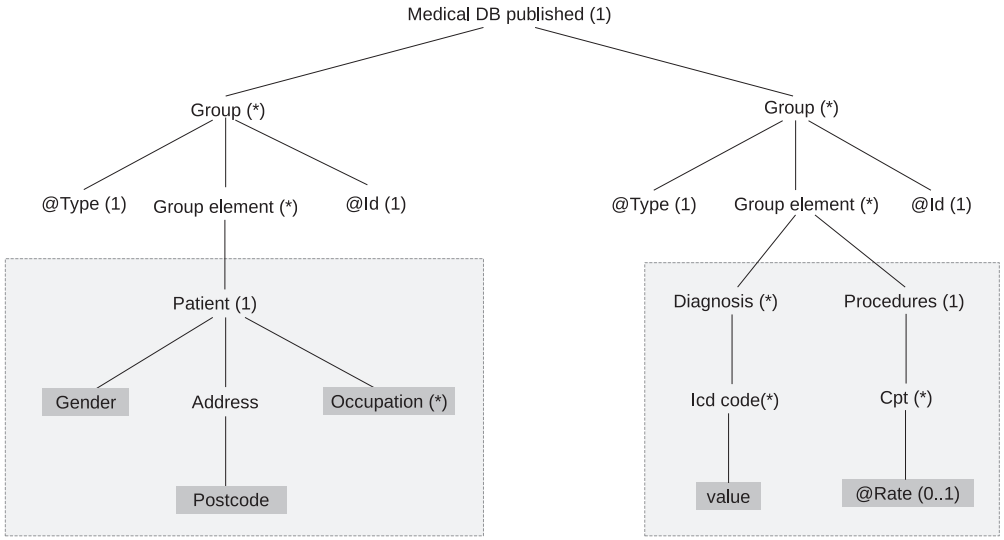

Fig. 5. Medical XML database schema (published schema).

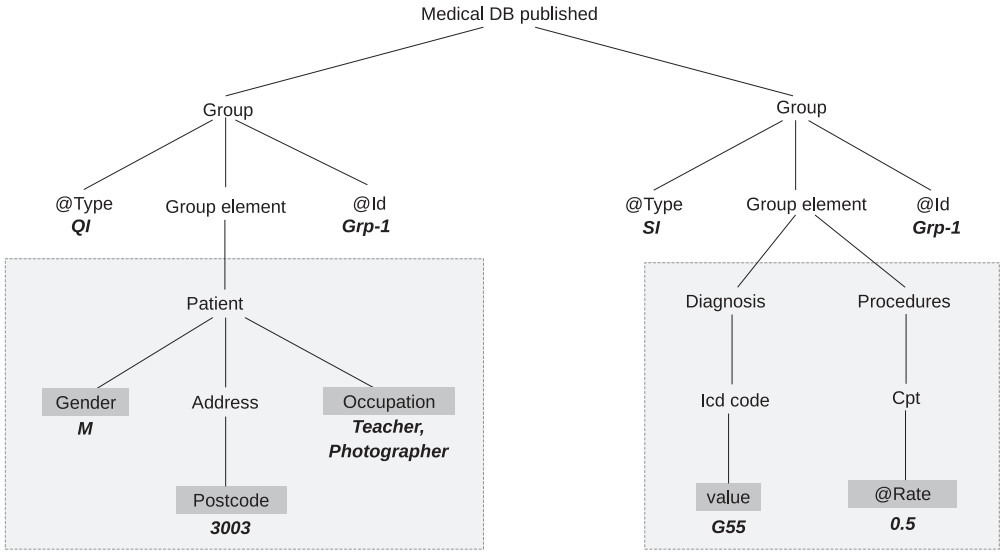

Fig. 6. Medical XML database document (published document).

- *occurrences* : *node* $\mapsto$ *range*, where *range* is a pair of non-negative integers *(lower, upper)* where *lower* $\leqslant$ *upper*. For convenience, we also assume that the *upper* can take value ∞, which represents an unrestricted upper bound. □

Thus, for our purpose, an XML schema is a labelled rooted tree, in which for each node, we can talk about its name (label) and its occurrences. But that is only "half of the story". The other "half", which also of crucial importance, concerns its practical usage. When translating from an XML schema to its formal representation, we impose the condition that *only the leaf nodes of the tree will bear information.*

It is important to make the following remarks about this restriction:

- The restriction confines the non-leaf nodes in the formal representation to the role of *structure-information-bearing*, and exclude them from the role of *data-bearing*. (This restriction allows us to later define QI-SI as sets of non-leaf nodes of the representing tree.)
- This restriction will *not* result in any loss of generality. In fact, a leaf node in the tree can represent a simple leaf node in the XML schema, or a complex leaf node, or an attribute in a node.
  Take the case of an attribute A of an XML element E. Then we represent A as a child of the node representing E. Consider, as a more specific example, an XML *leaf* element E with an attribute A. This node can store two pieces of information: (i) the value of the attribute and (ii) the "content" of the node (expressed between the pair of beginning and ending tags). Such an element can be formally modelled as a node that has two child nodes: one to represent the attribute (whose name can be the name of the attribute) and one to represent the content value of the node (whose name can simply be the string "value").
- This restriction is one on the *practical usage* of the formal representation. It is a restriction that has to be taken into account when we translate an XML schema into its formal representation. That is why the restriction does not appear in the formal definition above.

2.1.2. Definition of context element and QI and SI sets

The previous definition is a representation of XML schema in general (sufficient for the purpose of this paper). It can be used as a formal model for both private and published XML documents. The specific structure of a schema of a private document is given, and therefore, its formal representation can be taken to be already given in any particular release of XML data for public use. As for the schema for published XML data, it is our task to define it, which is the task we turn our attention to now. Note, that the concepts to be introduced in Definitions 2 and 3 are requirements for an input XML document to be privacy protected. These constraints also exists for other privacy models, including *l*-diversity and Anatomy.

For relational data, this task is easy to accomplish. Consider the example shown in Fig. 1, the schema for published data can be read directly, to a large extent, from the QIT and SIT tables there presented. Those tables contain information about both the meta-data (the schema) and the data. But what would be the equivalent for XML?

Obviously, to answer that question, we need to extend the concepts of QI and SI for XML data. Behind these concepts is the concept of an "*individual*", which is obvious in the case of relational data. Each row of the first table in Fig. 1 is the data of an individual – a patient. Each row of this table, of course, contains, the QI and SI data of an individual before the dissection. After the dissection (and grouping), each row of table QIT belongs to some (and the same) individual; similarly for each row of table SIT. Thus, for relational data, at least for the simple case of this example, data items that are in the same row belongs to the same individual.

For XML data, of course, we can no longer have the "being on the same row" mechanism at our disposal. Thus, we need to somehow capture the concept of "individual", so that we can say that certain data belongs to the same individual.

Our proposed solution to this problem is the concept of *context element* (for want of a better term). Intuitively, the context element is the node such that all the data in its subtree belongs to the same "individual". The formal definition of this concept is very simple.

Definition 2 (*Context Element*). Given a tree representing a private XML schema, the *context element* is a designated XML schema element whose occurrence is $(0, \infty)$. □

Admittedly, at first sight, the definition does not seem to make much sense. To clarify the idea behind the definition, it is necessary for us to make the following points:

- As a consequence of the definition, given a private XML schema (or its equivalent tree), we require the designer of the published schema to designate an element to be the context element, and all that we require of that element is that it has multiple occurrences (that is, the occurrences of the corresponding node in the tree is $(0, \infty)$.
- We require the context element to have multiple occurrences to ensure that a conforming XML document can have many instances for this element.
- However, as far as the practical usage of the definition is concerned, that designated element should represent the individuals of interest, whose sub-elements contain the related QI and SI. In fact, as will be seen in the next definition, we require the QI and SI elements to have the context element as a common ancestor.
  Note that this point is about the usage of the definition, and therefore it is not included in the formal definition itself.

Example. In our running example (see Fig. 4), the element *Medical DB Message* is to be designated as the context element.

Having defined the context element, we can now give the formal definition of QI and SI elements. Intuitively, once the context element has been identified, all we need to do to specify QI and SI data is to point out which leaf element represents a QI item, and which represents an SI item.

Definition 3 (*QI and SI Sets*). Given a tree representing the private XML schema with a context element, the sets of QI and SI elements are two designated set of nodes which satisfy the following conditions:

(i) The two sets are disjoint.
(ii) The nodes in the two sets have the context element as a common ancestor.
(iii) The nodes in the two sets are leaf nodes. □

The first condition is obvious: a QI element cannot be an SI element and vice versa. The second condition allows the interpretation that the QI and SI elements are related to the same individual. The third condition means that QI and SI elements are data-holding elements, not structuring ones. By specifying leaf elements we automatically include the paths that lead to those elements. This substantially simplifies the definition (without loss of information).

Example. In the running example (see Fig. 4), the QI set is the set of (dark-shaded) elements *Gender, Postcode,* and *Occupation*, and the SI set is the set of elements *(Icd code) value*, and *(Cpt) @Rate*.

2.1.3. Definition of XML schema for published data

We now consider the definition of the XML Schema of Published Data. Let us have a look at Fig. 5, which can give us an idea of the overall structure of the published schema. We can divide the schema into three parts. The non-shaded part at the top (consisting of nine nodes) shows that we structure the published data into two types of groups: one type of groups for QI data (on the left) and one for SI data (on the right). The shaded part on the left shows the structure of QI data, and the shaded part on the right, the structure of the SI data.

As will be seen, to describe the QI and SI part of the published schema, we will use the concept of "fragment". This is a concept taken from the XML field. In the context of XML, a fragment is a "path" from one element to a leaf element. The term has the same meaning when applied to the tree representing the XML schema or XML document.

As it turns out, the complete mathematical definition is quite detailed and the features may be obscured by those details. To save space and better expose the ideas, we present an outline of the definition of published schema.

Note: By viewing an XML schema as a rooted tree, we inherit a number of standard useful concepts (functions), which are very useful when we wish to perform the complete formalisation of the definition outlined below. These functions include: *level (n)*: level of a node n within the tree (level of root is 0); *parent (n)*: parent node of node n; *ancestors (n)*: the set of ancestors of node n; *descendants (n)*: the set of descendants of node n; *leaf (n)*: true if n has no children, false otherwise; *path (a, d)*: the path from node a to node d as a sequence of nodes with edges connecting them; *height (tree)*: length of longest path; and a *common ancestor* of a set of nodes; etc.

Definition 4 (*The Schema for Published Data – Outline:*). Given the following[3]:

(i) A tree T_S that represents the XML schema of private data.
(ii) The designated context element.
(iii) The designated sets of QI and SI.
(iv) The name for the root element of the published XML schema.

The schema for the published data is a tree that has three parts (as can be seen in the example shown in Fig. 5):

- **The upper part.** This part consists of 9 nodes and their edges. The 9 nodes are:
  - The root node (e.g. node "Medical DB published" in Fig. 5).
  - QI-Group node (e.g. node "Group" on the left).
  - QI-Type node (e.g. node "@Type" on the left).
  - QI-Group-Element node (e.g. node "Group element" on the left).
  - QI-Id node (e.g. node "@Id" on the left).
  - SI-Group node (e.g. node "Group" on the right).
  - SI-Type node (e.g. node "@Type" on the right).
  - SI-Group-Element node (e.g. node "Group element" on the right).
  - SI-Id node (e.g. node "@Id" on the right).

This part of the tree, considered as a graph, can be defined directly. That is, we define each node, with its name and occurrences, and we can define the edges as pairs of nodes.
For example, we can define the *root* as node R such that *name (R)* = the fourth input item (e.g. "Medical DB published") and *occurrences (R)* = $(1,1)$. We can define the *qi_group* node as node *QIG* such that *name (QIG)* = "Group" and *occurrences (QIG)* = $(0,\infty)$. Then we define the edge connecting these two nodes as *(R, QIG)*. Similarly, we can define all the other nodes and edges. And finally we include those nodes and edges in the tree representing the published XML schema, i.e. with $nodes = \{R, QIG, \ldots\}$ and $edges = \{(R, QIG), \ldots\}$.

- **The QI part.** This part consists of the subtree below the QI-Group-Element node (see Fig. 5). To construct this part, first we extract from the T_S tree all the *fragments* that consist of the QI-elements and all their ancestors up to, but not including, the context element. We then attach these fragments as subtrees of the QI-Group-Element node in the T_P tree.

  Note on formalisation details: From T_S, we can extract the QI-fragments as follows. For a given QI-node, we can extract all its ancestors, call this set X. We can also extract all the ancestors of the context element, call this set Y. Then $X \setminus (Y \bigcup \{context\_element\})$ gives us all ancestors A of the QI nodes up to, but not including, the context element. We can now restrict the graph T_S to nodes in $A \bigcup \{theQINode\}$ and we get the fragments for that QI-node. In this way, we can extract all the desired fragments, as a graph, call it F. For each fragment in F, we can get the top element, the one that has no parent. For each top element *top* of a fragment, we can define an edge *(QI_Group_Element, top)*. Adding the fragments and the edges (connecting top element to the QI_Group_Element nodes) to the graph representing the upper part, we get the tree consisting of the upper part and the QI part.
- **The SI part.** We repeat what we did for the QI part to get the SI fragments. Similarly, we can attach each fragment to the tree of the upper part and QI part to obtain the final T_P tree for the published data. Pairs of corresponding QI and SI groups are linked by a common *@Id* as depicted in Fig. 5. □

At this point, we may ask this important question: In the definition of the schema for published XML data, as outlined above, where does the satisfaction of the chosen privacy property come in? For example, suppose we want to achieve l-diversity, then where would the satisfaction of l-diversity come into the picture? The answer is that, as far as the definition above is concerned, it does *not* come into the picture at all: it is simply irrelevant at this point.

To see more clearly why this is the case, it is useful to consider the analogous situation for relational data. The schema for published XML data, complex as it may appear, is equivalent to the relational schema for tables QIT and SIT in Fig. 1. This relational schema is manifested by the column names of the two tables. The grouping of the QI and SI rows is irrelevant to the structures of the two tables. It is only when we fill the QIT and SIT tables with data that the satisfaction of the chosen privacy property comes into effect, which results in the cells of columns *Group-ID* of the two table being filled with specific group IDs.

In the same way, for the XML domain, it is only when we construct the published XML document (which we will cover next) that the effect of the chosen privacy property comes into being.

2.2. Method for constructing published data

Before considering a specification for published XML data, let us have a look at the published XML document shown in Fig. 6. We can observe that it has two parts.

[3] In the algorithm to extract the published XML data, presented later, the items given here are part of the inputs for that algorithm.

The upper part, which is non-shaded, is derived directly from the structure of published XML schema shown in Fig. 5. This part does not depend on the actual data in the private XML document. The second part, which is shaded, is where we actually have the data. The main task of producing published XML document is to populate this second part with appropriate data. How are we to do that?

To express the key idea in populating the published XML document, it is convenient to define a few terms.

- By "schema-QI-fragment", we mean a fragment in the published schema which goes from the context element, but excluding the context element, to a QI-element. An example of a schema-QI-fragment is the fragment Patient–Address–Postcode in the schema in Fig. 5.
- By "instance-QI-fragment", we mean a fragment in the private or public document that is corresponding to a schema-QI-fragment in the published schema. An example is the fragment Patient–Address–Postcode: 3003 in the private document in Fig. 2, and the fragment of the same content in the published document in Fig. 6.
  Note that it is possible for us to identify an instance-fragment in the *private document* that corresponds to a schema-fragment in the *published schema* because of the shared fragment structures between the *private schema* and the*published schema.* For convenience, we will refer to this property as *"shared fragment property"*.
- Similarly for the terms "schema-SI-fragment" and "instance-SI-fragment".

Now the key idea to populate the published document is:

1. First extract the instance-fragments (from the private XML document) that are corresponding to the schema-QI-fragments and schema-SI-fragments (in the published XML schema).
2. Then attach these instance-QI-fragments and instance-SI-fragments to the various groups in the published XML document.

To facilitate the stating of the definition of the targeted published XML data, we find that it is useful to make three assumptions.

The first is about instance-fragments and the corresponding schema-fragments. Given an XML schema, an XML document (instance of the XML schema), and a schema-fragment in the XML schema, one thing we need to do is to identify all the instance-fragments in the document that correspond to the schema-fragment in the schema. To save space, and because it is a problem in the XML domain in general, we will assume that it can be done. From the formal point of view, it means that we assume that there is a function (called *fragments*),

$$fragments(schema, instance, schema\text{-}fragment)$$

that takes three arguments: a *schema*, an *instance* of the schema, and a *schema-fragment* in the schema, and returns all the *instance-fragments* in the *instance* corresponding to *schema-fragment*.

The second assumption is that, given an instance-QI-fragment or instance-SI-fragment, we can map it to a pair of (name, value), where the name depends on the corresponding schema-QI or schema-SI-fragment, and the value is the value of the leaf element of the instance-fragment. In other words, formally, we assume that there is a function

$$convert: \ instance\_QI\_fragment (or\ instance\_SI\_fragment) \mapsto (name, value)$$

For practical computation, the *name* can be taken as pathname of the schema-fragment. For example, the instance fragment *Patient-Address-Postcode* in Fig. 6 would be mapped to the name-value pair of $(Patient.Address.Postcode, 3003)$. [4]

The third assumption is about the satisfaction of the privacy property. We will assume that, with respect to a chosen privacy property, given a pair of QI and SI tables, we can tell if the pair satisfies the chosen privacy property. Formally, we assume that there is function

$$satisfy: \ (QI\_table, SI\_table) \mapsto boolean$$

where the QI and SI table has an attribute to indicate which group a row belongs to. More will be said about the role that this function plays – especially in the design of our algorithm.

A formal definition of the published XML data can now be given.

Definition 5 (*Published XML Document*). Given the followings

(i) a schema T_S ("S" stands for "secret", which means "private"),
(ii) the designated context element,
(iii) the sets of QI and SI elements (in T_S),
(iv) an instance D_S of T_S, and
(v) a chosen privacy property, which plays its role via function *satisfy*.

Let T_P be the schema of the published data. Now suppose D_P is the document/instance of the published data. Then D_P must satisfy the following conditions:

1. D_P is an instance of T_P, where T_P is the derived published schema.
2. Let *qif* be a schema QI-fragment in the published schema (which matches a QI-fragment in the private schema, and which is possible due to the *shared fragment property*), then every instance fragment of $fragments(T_S, D_P, qif)$ must appear once and only once in th published document D_P.
   This condition means that each instance-QI-fragment must appear in a QI-group according to the structure imposed by the published schema.
3. Similarly for SI-fragments. Thus, each instance-SI-fragment must appear in one of the SI-groups.
4. There is a one-to-one correspondence between the QI-groups and SI-groups in the published document D_P by the means of QI-Id and SI-Id nodes (as defined in Definition 4). (Note that this condition is not imposed by the published schema.)
5. And finally, the QI and SI groups must satisfy the chosen privacy property. More specifically, given a pair of sets of QI-groups and SI-groups in XML format,
   - First, by virtue of function *convert*, we map these two sets into a QI table and an SI table. For each instance-fragment *ifr*, $convert(ifr)$ will give it a name and a value. The name will be the column name and the value will be the value of the cell.
   - Then, by virtue of function *satisfy*, we can tell if the pair of QI and SI tables satisfies the chosen privacy purpose or not. □

Note that the definition is declarative in nature. We simply list the conditions that a published document must satisfy to be qualified as the one to be obtained from the given private schema, the context element, the QI and SI sets of element, and a private XML document.

[4] Sometimes, the "value" in the name-value pair can be a set of values. As an example, Fig. 2 shows a patient with two occupations, Teacher and Photographer. Because Occupation is a QI item, and the δ-dependency property (that we choose) is only concerned with SI data for grouping, we can accommodate the multiple values for Occupation (see the result shown in Fig. 6 and the online Appendix). In this case, value (in the name-value pair) can be a set.

With respect to condition 5, the way our algorithm satisfies this condition is done as follows. First we collect all the instance-QI and instance-SI fragments. Then we use the techniques presented above for function *convert* to convert each fragment into a name-value pair. Using those name-value pairs, we construct the equivalent QI and SI tables, which at this stage do not include the values for group ID. We then use the privacy technique, inherited from relational data work (e.g. for *l*-diversity) or otherwise provided (e.g. for δ-dependency), to "bucket" the rows of the QI and SI tables into groups. Then based on this grouping, we attach the instance-QI and instance-SI fragments to their respective groups in the published XML document. Thus, in the definition above, function *satisfy* is used simply as an oracle to state the constraint that the published XML document must satisfy the nominated privacy property. And in the algorithm, which is the actual construction of the published document, we must ensure that this is the case.

2.3. Method for enhancing privacy protection with δ-dependency

We propose a new privacy protection property δ-dependency, which seeks to exploit the semantics of sensitive information to improve on the privacy protection of published data. To clarify the motivation for δ-dependency, we provide the following example where, *Lung Diseases* and *Influenza* are the SI values of two individuals in the data to be privacy protected. Existing privacy methods such as Anatomy and *l*-diversity will treat them as *different* concepts and may therefore group them into the same QI/SI group. But the two SI values are both *Respiratory Diseases* (please refer to Fig. 7). This means that if we selected the SI group that contained *Lung Diseases* and *Influenza*, we can use our knowledge of the ICD-10 disease taxonomy to infer that two of the SI values are in fact the same, if we generalise them by one step in the taxonomy (hierarchy). This generalisation *reduces* the degree of diversity.

Our review of related work shows that there are still no existing dissection-based approaches that use sensitive information's generalised values to create QI groups in which SI remain distinct even after they have been generalised [22–24]. Câmpan et al. [25] propose an approach that has some similar elements to ours, but which is based on *k*-anonymity and generalisation of the published data, only considers the diversity of SI's ancestors, whose QI values are identical.

It must be noted that the technique proposed by Câmpan et al. is not based on dissection, and as such produces groups where QI and SI values are still combined. To produce privacy protected tables, it relies on generalisation of QI values, which has the effect that all QI values in each group are equivalent. However, for generalisation to be effective, it requires QI values within a group to be equal or requiring little generalisation to prevent information loss. This is a constraint because it means that the resulting QI/SI groups satisfying *p*-Sensitive *k*-anonymity either have equivalent QI values (best case), or, in the worst case, QI values were required to be generalised significantly and the resulting data loses information value. If using our method, on the other hand, this problem will not arise, as we can group *any* pairs of QI/SI, as long as the SI are sufficiently distinct with respect to their ancestors. Importantly, our proposal in this paper does not require the user to pre-configure the sensitive value hierarchy, while for the method proposed by Câmpan et al. [25], it is required, representing an additional constraint.

Definition 6 (*Sensitive value hierarchy*). A sensitive value hierarchy is a labelled rooted data tree (note, it is not a schema tree) for a specific property in an application domain, such that:

1. Each label is a value for the associated property (e.g. I64 is a value for the property *Disease* in Fig. 7).
2. Every label is distinct (no two nodes have the same value). □

It is important to note that the sensitive value hierarchy is given to us by the application domain. For example, when dealing with medical or patient records, the hierarchy in use may be represented by the ICD-10 disease codes and their respective relationships. An excerpt of this hierarchy is provided in Fig. 7. From a practical perspective, we are only considering hierarchies with single inheritance, i.e. the hierarchy is a tree where any node (except the root) has only one parent. Further, it is worthwhile noting that other prominent papers [4,5,7] on privacy also use single inheritance generalisation hierarchies.

Definition 7 (δ-*dependency of a set of sensitive values*). Given a set of sensitive values SV (in a sensitive data hierarchy), and let A be the closest common ancestor (CCA) of SV. Then we say that SV satisfies δ-dependency iff for every x in SV, we have $d(x, A) \geqslant \delta$, where $d(x, A)$ is the length (number of edges) of the path from A to x.

As additional terminology, if a set SV satisfies 2-dependency, for example, then we also say that it is "2-dependent", or it is "δ-dependent of degree 2" (or at "level 2"). □

Note on δ-dependency for unbalanced sensitive data hierarchies: Where a sensitive data hierarchy is an unbalanced tree, achieving δ-dependency for a data set that maps to the hierarchy

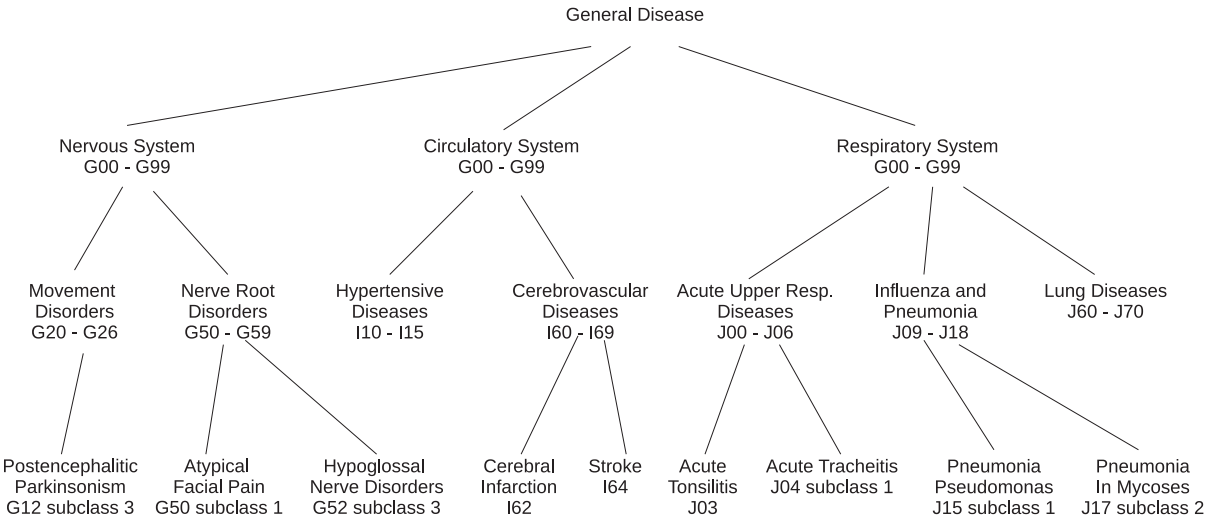

Fig. 7. ICD-10 Disease code hierarchy (excerpt).

may be more difficult, because the SI node values lie on different tree levels on just a few branches within the hierarchy. While such a scenario will inevitably constrain the maximum level of δ that can be achieved for the data set, our δ-dependency algorithm does support unbalanced hierarchies.

As an example, consider Fig. 7 and Table 3. Diseases *Postencephalitic Parkinsonism* and *Lung Diseases* are on different tree levels in the hierarchy, yet are grouped for privacy purposes. As their closest common ancestor (the root) is three and two edges away, respectively, a group of these two SI values meets the definition of 2-dependency.

Definition 8 (*δ-dependency of a QI-SI grouping at level d*). Given a QI-SI grouping T_P that consists of a number of pairs of QI and SI groups (in a privacy protected data set), we say that grouping T_P satisfies δ-dependency at level d iff every SI group in T_P is δ-dependent at least at level d.

For the special case where a node is missing or has been suppressed in the private data, then it is mapped as null value. As far as δ-dependency is concerned, the ancestor of null is also null. The distance between null and any other value (including null) is ∞. □

Case study in support of the improved privacy protection of δ-dependency. Using the algorithm that will be presented in Section 2.4, we run it on the private XML documents of a case study. Apart from showing how our dissection and grouping achieves δ-dependency, and illustrating the steps of the general algorithm for the specific case study, we also compare the δ-dependency approach with Anatomy, an approach that resembles ours most as it also dissects and diversifies the data to obtain QI and SI groups without generalising the data. We therefore believe that a comparison against Anatomy yields a fair evaluation given that XML privacy approaches are yet to adopt the notions of QI and SI groups. Other approaches such as Personalized Privacy Preservation [21] or Privacy Preserving OLAP over Distributed XML Documents [26] employ generalisation as a tool to enforce privacy protection and are therefore less suitable for a comparison study. In this section, we will discuss how the general algorithm works for the case study, and then we discuss the improvement that we can gain from δ-dependency in comparison to Anatomy. Later in Section 3, more specific quantitative results for comparison will be presented.

Given an XML document of the type illustrated in the online Appendix, and given a sample data population as illustrated in Table 2 that can be mapped into sample hierarchy illustrated in Fig. 7, the δ-dependency algorithm is to be applied with the input parameters listed in Table 1.

The sensitive information (addressed by a combination of XML element and an attribute) and their respective occurrences throughout the data are both illustrated in the hierarchy in Fig. 7 and Table 2.

To compute the QI and SI groups for the above described data, we aim for $\delta = 2$, and must therefore treat the most sensitive information in the leaf nodes of the tree the same as their generalised ancestors one tree level above. By doing this, we ensure that sensitive information such as Cerebral Infection (I62) and Stroke (I64) are not arranged into the same SI group.

To illustrate the method for grouping QI and SI such that the resulting groups satisfy δ-dependency, we outline the first number of steps manually, at each step comparing how an alternative technique (e.g. Anatomy) would have performed.

The initial step is to sort SI by their occurrences in the data, and since the most specific values are being treated in a one-level generalised manner, this must be considered when computing the occurrences. The four most frequent SI are *Acute Upper Resp.*

Table 1
Parameters.

Input parameter	Parameter value
XML document	see Listing 1 in the online Appendix
QI group size	4
δ to achieve	2
Hierarchy T_S	see Fig. 7
Context Element	medicalDBmessage
QI	Gender Postcode Occupation
SI	Icd code Rate

Diseases with a total count of 2 + 4 (*Acute Tonsilitis*) + 4 (*Acute Tracheitis*) = 10, *Movement Disorders* with a count of 8, and *Nerve Root Disorders*, *Cerebrovascular Diseases* with each a count of 6. As can be seen, occurrences are being treated cumulative bottom-up. The resulting SI group computed from this (initial) run of the algorithm is listed in Table 3. Referring back to Fig. 7, we observe that *Movement Disorders* and *Nerve Root Disorders* have a closest common ancestor (*Nervous System*) that is not the root.

In comparison, the Anatomy algorithm would have identified the four most frequent groups by their sole counts, resulting in *Movement Disorders* (4), *Hypertensive Diseases* (4), *Lung Diseases* (4), and *Postencephalitic Parkinsonism* (4) to be selected as most frequent groups, for example. *Acute Tracheitis* with an occurrence of 4 could also have been selected as it has an occurrence equal to the other (selected) groups. The difference in quality of privacy becomes obvious immediately, because Anatomy would create SI groups with both SI items from G20-G26 (*Movement Disorders*) and G12 (*Postencephalitic Parkinsonism*), which is its direct descendant, resulting in an ancestor relationship that may cause a privacy breach.

Table 3 illustrates how δ-dependency and Anatomy performed in comparison for the first (sample) SI group that was computed. Although the privacy measure $\delta = 2$ for both groups computed, the SI group generated by δ-dependency has a greater degree of privacy since there exists no ancestor–descendant (type-of) relationship between *Movement Disorders* and *Nerve Root Disorders*, as outlined in the first column of Table 3.

With continuous computation of further SI groups (and associated QI groups), it can be observed that the Anatomy approach shows random behaviour when computing semantically independent QI and SI groups. As a result of this, privacy leaks are easier to identify and individuals' identities are easier to be linked to concrete and specific sensitive information because the values in SI groups are not sufficiently diverse.

The case study illustrates the theoretical working of our approach on a sample XML document and input parameters and shown that our proposed solution computes QI/SI groups with a minimum of parent or close ancestor relationships, while maximising far ancestor relationships with the overall objective to achieve node independence in all SI groups.

The benefit of our approach as compared to Anatomy is that the semantic distance within SI groups is always optimal, where Anatomy computes random SI groups with respect to semantic distance and as such can only perform as good as δ-dependency in its optimal case.

As a final remark, it should be noted that while δ-dependency, as described in this paper, is based on, and further improves the Anatomy technique, it is equally possible to apply the δ-dependency privacy property in combination with other privacy techniques that are based on grouping. For example, *l*-diversity could

Table 2
SI occurrences.

ICD-10	Description	Count	ICD-10	Description	Count
G00-G99	Nervous System	0	G12.3	Postenc. Parkinsonism	4
I00-I99	Circulatory System	0	G50.1	Atypical Facial Pain	4
J00-J99	Respiratory System	0	G52.3	Hypoglossal Nerve Disorders	2
G20-G26	Movement Disorders	4	I62	Cerebral Infarction	4
G50-G59	Nerve Root Disorders	0	I64	Stroke	2
I10-I15	Hypertensive Diseases	4	J03	Acute Tonsilitis	4
I60-I69	Cerebrovascular Diseases	0	J04.1	Acute Tracheitis	4
J00-J06	Acute Upper Resp. Dis.	2	J15.1	Pneumonia Pseudomonas	2
J09-J18	Influenza and Pneumonia	0	J17.2	Pneumonia in Mycoses	2
J60-J70	Lung Diseases	4			

Table 3
Computed SI groups.

δ-Dependency	Common ancestor	Type-of	Anatomy	Common ancestor	Type-of
Acute Upper Resp. Diseases			Movement Disorders	×	×
Movement Disorders	×		Hypertensive Diseases		
Nerve Root Disorders	×		Lung Diseases		
Cerebrovascular Diseases			Postencephalitic Parkinsonism	×	×

be modified to create QI/SI groups in which all groups satisfy δ-dependency (in addition to *l*-diversity).

2.4. Algorithm for dissecting and computing δ-dependent published data set

We propose an algorithm that computes a δ-dependent published XML document, based on the input of a private XML document with a context node, a set of QI and SI elements, the group size and a hierarchy into which the SI values can be mapped. The goal of the algorithm is to group QI and SI nodes in such a way that they satisfy the δ-dependency privacy property (diversification) and separate QI and SI nodes so that there is no direct link between them (dissection). This is achieved by repeating steps 2–4 of the algorithm multiple times, each time attempting to increase δ by 1. If after an iteration δ-dependency cannot be achieved, the algorithm will stop.

Initially, the target δ to be achieved is set to the height of the hierarchy, as this is the maximum degree of δ that can possibly be achieved. The algorithm will then attempt to achieve 1-dependency, 2-dependency, and so on, until either the target δ, or the highest possible δ, has been achieved. The algorithm starts by identifying the SI node in the XML document, whose value is closest to the root in the hierarchy, i.e. the most general SI node, where the bottom of the hierarchy represents specific values and the root the most general value.

The first iteration will therefore try to achieve the lowest δ possible, grouping all SI nodes of the XML document into groups of *N*, based on their ancestors' values. Ancestors in this context refers to the values that are obtained by mapping an SI node from the XML document into the hierarchy, then generalising it δ times. While in the best case, *CCA* will be a multiple of *N* and have an even distribution so that all context nodes in *CCA* can be dispersed to SI groups where each SI value is distinct, this may not always be the case in practice. For example, *CCA* may not be a multiple of *N*, in which case there will be residual nodes that need to be dealt with. Anatomy does this by dispersing them to existing, random SI groups. Line 21 in our algorithm assigns the residual SI nodes to existing groups that have the least number of that SI node (if any), which can be considered an equal method in comparison to how Anatomy solves the residual node assignment step. As a result, the verification step in lines 23–33 will fail in such a scenario, unless it is skipped for those groups that were assigned a residual node.

Similarly to other related approaches that rely on grouping, the group size *N* plays an important role in the privacy protection, as it determines how many distinct SI nodes will be contained within each group, and which has a direct effect on the probability that any SI node within a group can be re-identified. Generally, this probability is 1 divided by *N*. In our experiments in Section 3, we test our algorithm against different groups sizes to analyse its effect on the efficiency and effectiveness of our method.

For this paper, we provide a simplified version of the algorithm that does not cater for multiple SI nodes per context node. This means that in our example data sets, each individual has only got one SI node. The following experiments are also limited by this scenario. We will revisit this issue of multiple SI in more detail later in the paper.

For the grouping step, it is important to know the distribution of SI values in the XML document. This required statistical information is collected in lines 7–11. We store each pair of all distinct SI node values, together with that value's occurrences in a collection of pairs. This step also stores the values and occurrences of SI nodes' ancestors (see lines 10 and 11), as we will need to group SI nodes based on what their ancestor nodes' values are.

The group creation step (lines 13–21) is implemented using a priority queue where node values with highest occurrence appear first in the queue. In this way, new groups are created by selecting the first *N* nodes and assigning them to an SI group. The corresponding QI group is also created at this step.

When grouping nodes based on ancestors' values (see line 15), the node assignment into groups still needs to fetch nodes actually occurring in the XML data. This is achieved in line 17, where a random descendant of the ancestor can be grouped. It does not matter which node is grouped, as long as its ancestor is different to all other nodes' ancestors in the same group.

Once all groups have been created, the algorithm computes δ of all groups and re-starts with $\delta + 1$, if all groups satisfy δ. Otherwise the algorithm is finished and the published XML document is returned.

In each iteration, the algorithm systematically groups different SI nodes, depending on their ancestor relationships. For example, in the first iteration, only the SI nodes' parents may be considered for grouping, while in subsequent iterations their δ-th ancestors

are considered. This eventually results in nodes being grouped, whose common ancestor cannot be inferred unless the SI nodes are generalised by δ levels.

Some groups may or may not achieve the desired δ in a given iteration of the algorithm. This will result in the overall published data set having the strongest possible privacy protection using our approach. In the optimal case, all groups, and therefore the overall resulting published data set, will be δ dependent. In the worst case, i.e. when $\delta = 0$, then the published data still satisfies the Anatomy privacy property.

Algorithm 1: Dissection with δ-dependency

```
Data:
D_S : XML document /* private data */
CE : Element /* context element */
E_QI : Set of Elements /* QI elements */
E_SI : Set of Elements /* SI elements */
T_S : Tree of Strings /* Sensitive values hierarchy */
N : Integer /* group size */
Result:
D_P : XML document /* published XML document */
PUB_QIofD_P : Set of XML Fragments /* fragments of published QI data */
PUB_SIofD_P : Set of XML Fragments /* fragments of published SI data */
1  begin
2      /*STEP 1. (Initialise)*/
3      δ ← height(T_S)
4      root_level ← 0
5      current_level ← level of SI value closest to T_S's root
6      /* STEP 2. (Extract SI statistical information from D_S) */
7      CCA ← all SI node values and occurrences in D_S
8      foreach distinct S in CCA do
9          if level(S) > current_level then
10             A ← (level(S)-current_level)-th ancestor of S in T_S
11         COL (A,count) ← (A, occurrences of A's descendants)
12     /* STEP 3. (Dissection and diversification; group creation) */
13     while COL ≠ ∅ do
14         Sort COL by count in descending order
15         PUB_SI_i ← first N SI fragments (below CE) in COL /* SI extraction */
16         foreach S in PUB_SI_i do
17             C ← random node of CCA that contains S or any of S's descendants
18             S ← lowest descendant of S (if no descendants, select S)
19             PUB_QI_i ← QI fragment (below CE) corresponding to C
20             Remove S from COL
21         Assign residual elements in COL to existing SI groups
22     /* STEP 4. (Privacy verification) */
23     foreach S in PUB_SI do
24         Compute δ_current of S
25         if δ_current < δ then
26             if current_level == root_level then
27                 Done
28             else
29                 current_level ← current_level + 1
30                 reset PUB_QI and PUB_SI
31                 repeat steps 2 to 4
32         else
33             Done
34     return PUB_QI and PUB_SI
35 end
```

The Issue of Multiple SI Nodes. Generally speaking, privacy protection techniques that are based on grouping and diversification of SI values (such as *l*-diversity and Anatomy) rely on a method for comparing SI values, to determine their (dis-)similarity and group them accordingly. Where a data set has only one atomic SI attribute, the comparison method is trivial, as it is simply a string/character comparison method. If there are multiple SI attributes, these can be concatenated and then privacy protected using existing techniques.

Table 4
Parameters of data and algorithm as used in the experiments.

Parameter name	Default	Description
n_ary	3	Cardinality. Number of child nodes per node
tree_depth	5	Hierarchical depth of SI. Number of tree levels
lvl_populated	2	Number of tree levels from the bottom up that are populated with SI nodes
freq_min	0	Minimum occurrences for SI nodes
freq_max	5	Maximum occurrences for SI nodes
group_size	5	QI/SI group size

In the literature, papers discussing privacy protection of microdata generally focus on scenarios where there is only a single, atomic SI value. Few proposed methods exist that comprehensively address the special case where there are multiple SI attributes in data that is to be privacy protected.

Machanavajjhala et al. consider this issue in their paper on *l*-diversity [5] and briefly discuss the challenges. The authors conclude that QI groups will need to be very large to support privacy protection over multiple SI, introducing significant distortion into the data due to the generalisation technique that the approach is based on. Xiao et al., in their paper on Anatomy [6], also mention multiple SI and include it as part of their future work. Wong et al. [27] claim that their method can be extended for multiple SI attributes and also for SI attributes that have multiple values, however no formal method, algorithm or experiment is provided in the paper.

A paper that specifically attempts to solve the problem is proposed by Gal et al. [28]. The authors propose an extension to *k*-anonymity [4] and *l*-diversity [5] for multiple SI attributes, by exploiting the dissection concept of Anatomy [6]. This allows the authors to reduce the distortion of the data (as no generalisation is used), but introduces a separate SI table for each SI attribute in the data.

Existing methods addressing multiple SI treat each SI attribute separate, i.e. they are not concerned with the constraint that two or more SI attributes could in fact be similar, related or even the same, which may have an impact on the privacy protection technique employed. To address such limitation, one can employ a more advanced method to compare the sets of SI values. In the XML domain, which we have been studying as part of this research paper, sub-tree matching of XML fragments appears as a possible method to achieve this. There are many proposed approaches on sub-tree matching, with Augsten et al. [29] and Cohen [30] being examples of existing papers that use tree edit distance as a similarity measure between sub-trees. In fact, the paper by Cohen specifically mentions use cases such as document similarity and similarity joining of XML documents to which its proposed methods are applicable. We believe that sub-tree matching as described in these papers, with appropriate extensions, will be suitable for our purpose to compare sets of SI nodes in XML, with the objective to create privacy protected XML data.

3. Experiments and results

In this section, we report a systematic study on our proposed XML dissection method and δ-dependency privacy property. The parameters and variables of the experiments are summarised in Tables 4–6.

In our experiments, we selected the Anatomy approach for comparison against δ-dependency and the pure dissection method. For the purpose of the experiments, we implemented both the XML dissection algorithm, the Anatomy algorithm (based on the XML dissection technique), and the δ-dependency algorithm as in Section 2.4. The objectives of our experiments are as follows:

Table 5
XML data sets: experiments configuration.

Dataset	Description	Context node	QI	SI	Group size
nasa	Astronomical Data: Datasets converted from legacy flat-file format into XML and made available to the public	dataset	altname title initial lastName name	affiliation	10, 20, 30, 40, 50
treebank	Partially-encrypted treebank: English sentences, tagged with parts of speech. The text nodes have been encrypted because they are copywritten text from the Wall Street Journal	EMPTY	DT NN IN CC JJ	NNP	10, 20, 30, 40, 50
protein	Protein Sequence Database: Integrated collection of functionally annotated protein sequences	ProteinEntry	CCHU name source length type	sequence	10, 20, 30, 40, 50
dblp	DBLP Bibliography: The DBLP server provides bibliographic information on major computer science journals and proceedings	inproceedings	title author pages booktitle url	year	5, 6, 7, 8, 9

Table 6
XML data sets: statistics.

Dataset	Size (MB)	Elements	Attributes	Max-depth	Avg-depth
nasa	23	476,646	56,317	8	5.58314
treebank	82	2,437,666	1	36	7.87279
protein	683	21,305,818	1,290,647	7	5.15147
dblp	127	3,332,130	404,276	6	2.90228

1. The first part of this section has the objective of measuring the *effectiveness* of the privacy property δ-dependency, as proposed in Sections 2.3 and 2.4. By effectiveness, we mean the δ degree that is achieved (the higher the δ degree, the better the privacy protection).
   First, we measure the δ degree achieved by our algorithm and Anatomy for comparison. With our algorithm, the published data always achieves the maximum possible δ degree. In contrast, the Anatomy algorithm may achieve some δ degree purely by chance, ranging from 0 to the maximum possible δ degree.
   Second, we measure the δ degree of our algorithm, which is the maximum possible δ degree, as a function of varying QI/SI group size, sensitive values hierarchies, even versus uneven distribution of data within the sensitive values hierarchy, and data size.
2. The second part focuses on examining the *efficiency* of the proposed XML dissection technique as proposed in Sections 2.1 and 2.2. By efficiency we mean the runtime performance and memory usage as a function of variable number of QI elements and group size. We believe performance is an important aspect, as we want to prove that extending dissection for the XML domain does not add unnecessary processing overhead. We want to show that there is no reason why XML data should be flattened if it can be privacy protected in its native form just as fast.

All experiments were conducted on an Apple MacBook Air running the 32-bit Linux Ubuntu 13.04 operating system, with an Intel Core 2 Duo CPU P7500 @ 1.60 GHz x 2, 1.9 GB main memory, and a 60 GB SSD. The programs were implemented in Java 7 and were compiled using Open JDK version 1.7.0_25. Our source code is available at Google Code.[5]

3.1. Effectiveness of δ-dependency to privacy protect data

We examine the effectiveness of δ-dependency on real and synthetic data. For the real XML, we use the *dblp* data set. For the synthetic data, we introduce a number of parameters that we use to generate a random data set with specific characteristics such as distribution of SI values, sensitive values hierarchy, data size and group size used for the algorithm. The population and distribution of sensitive values within the synthetic data set can be modelled as a tree based on the parameters as listed in Table 4. The programming code pertaining to the synthetic data-based experiments is written in the PHP language and can be freely downloaded from our repository on Google Code.[6]

3.1.1. Objective

The objective of the experiments is to show that (i) the δ-dependency technique provides superior privacy protection to the Anatomy technique (Fig. 8), and (ii) produces near-constant privacy protected data sets (Figs. 9a, b, 10a and b), which is done by measuring δ and graphing it as a function of the variables. The four variables used in the latter four experiments were selected for experimentation as they represent the most important properties of a data set that a privacy mechanism must cope with. In the following experiments we will quantitatively prove that our proposed privacy preservation model outperforms Anatomy, performs equally well under varying data inputs, and is not constrained to "ideal" scenarios.

For each experiment, with the exception of that reported in Section 3.1.3, we generate a random data set and run the *δ-dependency* algorithm against it. We also run each experiment for different

[5] http://code.google.com/p/twitter-analytics/source/browse/#svn%2Ftrunk%2FXMLPrivacy.
[6] http://code.google.com/p/delta-dependency/.

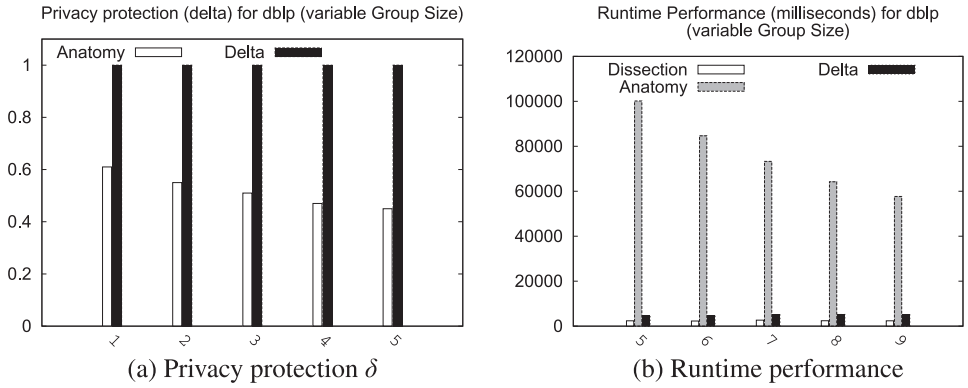

Fig. 8. Effectiveness and efficiency of privacy protection.

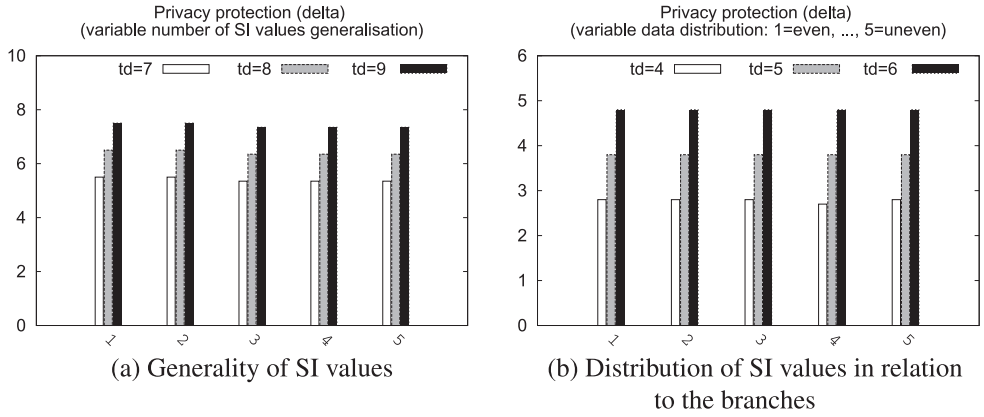

Fig. 9. Effect of the generality and branch-distribution of actual SI data.

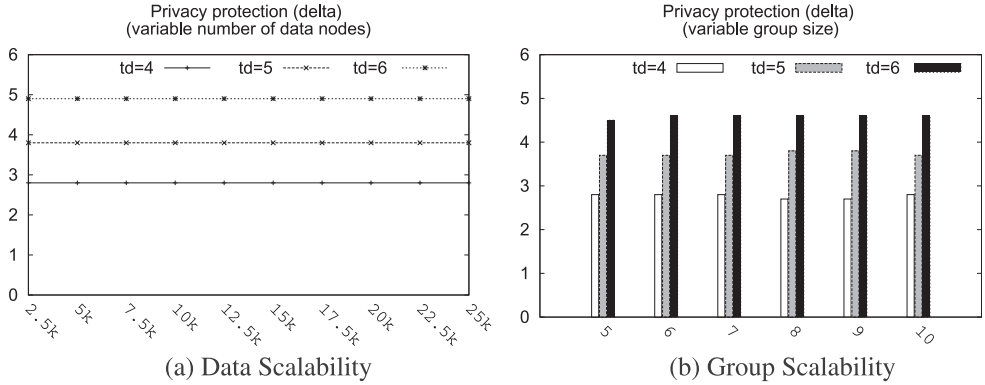

Fig. 10. Scalability of privacy method with increasing data and group sizes.

sensitive value hierarchy levels *td*, namely (4,5,6) and (7,8,9) to demonstrate how the approach behaves against this variable that specifies how many levels of generalisation the data can be protected against. For example, the tree in Fig. 7 has $td = 4$. This step is repeated 5 times and the average output value δ (depicted as *delta* in Figs. 9a, b, 10a and b) is computed, which serves as measure in all experiments and at the same time defines the quality of the privacy model.

Further, with the exception of experiment in Fig. 8, we measure δ for three different node fan-out values (3,4,5) of the hierarchy. This parameter defines the maximum semantic complexity that can exist within the data set, as it specifies how many SI values within the data set can be related to each other on each level within the hierarchy (i.e. how many SI have a common parent). The graphs in Figs. 9a, b 10a and b represent different hierarchy levels, where the top set of graphs relates to a tree depth of 6, the middle set of graphs relates to a tree depth of 5 and the bottom set relates to a tree depth of 4.

3.1.2. Experimental design

To examine the δ-dependency privacy model and algorithm with respect to varying data distribution and complexity of the sensitive value hierarchy, it is necessary to conduct a series of experiments against data sets of various patterns. We define the patterns by selecting parameters such as how specific/general SI values are, where in the sensitive value hierarchy they are mapped

(i.e. whether it is a balanced or unbalanced hierarchy), and whether the sensitive value hierarchy is narrow and deep (very specific SI, where few SI have common parents/ancestors) or wide and shallow (many SI values that have the same parents/ancestors). As it is not feasible for this research study to collect real-world representative data sets that correspond to so many combinations of previously mentioned parameters, we decided to take an alternative approach to our experiments.

We implemented a generic test data generator that first creates a sensitive value hierarchy based on input parameters tree depth and node fan-out. An associated test data set is then generated that contains sensitive data that maps into the hierarchy. This mapping is done so that the resulting synthetic data satisfies pre-defined parameters such as occurrences of SI, specific versus general SI, and distribution of SI, and corresponds to the data patterns that we want to examine.

To ensure accuracy of our experiments, we generate 5 sample data sets for each set of parameters. We then run the algorithm with different variables against each sample data set, and calculate the average degree of privacy δ for each set of runs. The first sets of experiments, reported in Fig. 8, are run against real XML data.

3.1.3. Direct comparison between Anatomy and δ-dependency

The first set of experiments is run against real data and benchmarks the δ-dependency method against the Anatomy privacy method. The XML data set used in the tests is dblp, as described in Tables 5 and 6. We applied our XML schema and data transformation model to dblp and ran our extended version of the Anatomy algorithm for XML data against the data set as configured in Table 6.

The results are reported in Fig. 8. Fig. 8a compares the effectiveness of the δ-dependency algorithm in comparison with Anatomy. For ease of comparison, we perform a transformation of the effectiveness (measured by δ degree). Because the δ-dependency algorithm always achieves the maximum δ degree possible, we normalise its efficiency to the value 1. For the Anatomy algorithm (which achieves some δ degree by chance), we compute the average δ degree of the various runs, and use it to compare with the effectiveness of δ-dependency (which is of value 1).

Fig. 8b shows that δ-dependency runs faster than Anatomy. This is due to the fact that δ-dependency is a more restrictive approach, and only requires to consider the ancestor values of the SI nodes, and not the SI nodes themselves, effectively leading to fewer comparisons and sorting operations.

The conclusion we draw from this experiment is that *δ-dependency achieves improved privacy protection and at the same time requires less runtime to achieve this.*

3.1.4. Effect of the generality of the actual sensitive data on the δ degree that can be achieved by the δ-dependency algorithm

How the actual SI data is located in the sensitive value hierarchy may affect the δ degree achievable by the δ-dependency algorithm. In the best scenario, those actual SI data are located at the leafs of the sensitive value hierarchy. The closer those locations are to the root, we expect that a lower δ degree will be achieved. In Fig. 9a scenario in which we generate the data with a bias towards the leaf, is given the value 1, and the scenario in which we generate data with bias towards the root is given the value 5. In addition, we also vary the depth *td* of the sensitive value hierarchy from 7 to 9. As far as this experiment is concerned, the randomness, even with the bias, yield similar results. Even though this is not conclusive evidence, it may indicate that the actual maximum δ degree achievable in practice is resistant to the degree of generality of the actual sensitive data.

3.1.5. Effect of the distribution of the actual sensitive data in relation to branches on the δ degree that can be achieved by the δ-dependency algorithm

How the actual sensitive data are located among the branches of the sensitive value hierarchy may affect the δ degree. The best scenario is when the actual sensitive data are distributed evenly among all branches, and the worst scenario is when they are concentrated on just a few branches. For Fig. 9b, the scenario in which we generate data with bias towards even distribution is given the value 1 and the worst scenario in which we generate data with the tendency to concentrate in a few branches, is given the value 5. Also, as in the previous experiment, we vary the depth of the sensitive value hierarchy between 4 and 6.

Similar to the previous experiment, the randomness, even with bias towards even distribution among the branches or concentrated on a few branches, yield similar results. As before, even though this is not conclusive evidence, it may indicate that the actual maximum δ degree achievable in practice is resistant to the distribution of the actual sensitive data with respect to the branches.

3.1.6. Data scalability

Fig. 10a relates to the third set of experiments that tests how the privacy scheme scales with data sets of increasing sizes. To test this, we generate data sets with a specific number of nodes, whereby the parameter *Maximum Occurrences* (freq_max) along the *x*-axis specifies the maximum possible occurrences of SI nodes, and the minimum occurrences is set to a default value of 0. Thus, this parameter gives an indication of the overall size of the dataset as it defines how many occurrences of any SI node there is within the data.

As expected, the size of the data does not impact on the quality of the privacy technique, where quality is the degree of privacy protection δ that can be achieved. For the three graphs in Fig. 10a, the maximum δ that can be achieved is 3, 4 and 5, respectively. The results indicate that our proposed technique performs very close to optimal privacy protection for data sets of up to 22,500 SI nodes.

3.1.7. Effect of group size

Fig. 10b shows the results of the experiments in which we group the data into QI/SI groups of variable sizes $(5, \ldots, 10)$ and then measure δ within the groups. The starting value for this variable was intentionally selected to be 5, as it (1) is consistent with experiments in related research works [31,6,5], and (2) because small group sizes are undesirable due to the fact that a decreasing group size implies an increase in the probability of privacy breach. Analysing the *Group Size* (group_size) parameter gives an indication of how resilient the δ-dependency privacy model is to group sizes. It can be observed that the degree of diversity δ remains near-constant as we increase the group size.

The group size is an important parameter as it is proportional to the probability P of re-constructing an individual's link between QI and SI values because (in the optimal case) $P(Group) = 1/GroupSize$. Further, we observe that a variable depth of the hierarchy that represents the semantic relationships in the data does not significantly influence the degree of privacy.

3.2. Efficiency of Dissection and δ-dependency

This section seeks to prove the efficiency of our proposed XML Dissection technique and the δ-dependency algorithm. As mentioned earlier in the paper, we believe performance is a key concern as we want to show that it is just as efficient to process XML data in its native format for privacy protection, as instead of needing to flatten it and re-assemble it for use by a relational

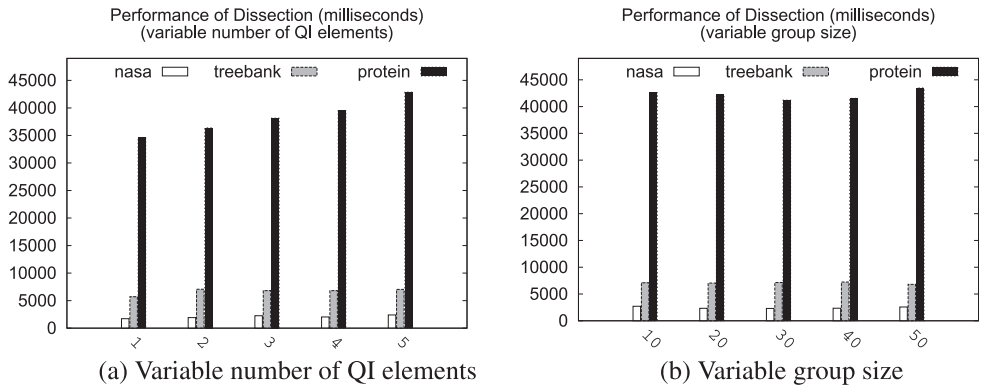

Fig. 11. Dissection runtime performance.

privacy model. The data sets that were used in the experiments are described in Tables 5 and 6, outlining what type of information they contain and how they were configured, i.e. which attributes were considered as QI and SI. We will refer to the data sets as *nasa*, *treebank*, *protein* and *dblp*, throughout this section.

For the sake of simplicity, we refer to the proposed method as the Dissection method, and benchmark it against the Anatomy privacy protection method. The two methods are denoted by Dissection, and Anatomy in the figures in this section. Since we are not using any randomised functions, each experiment is run once, and the absolute values are reported. We include the benchmark against the Anatomy method for a fair comparison.

Note that we do not report on memory usage for the dissection method as it is constant. Dissection by itself traverses an XML document and, when a QI or SI is identified, the relevant context path is extracted and written to an output file, i.e. not stored in memory other than for the file write. In fact, the only memory use Dissection has is to keep track of the path to a QI or SI, to allow for computation of the context fragment.

These data sets are freely available for download[7] to allow full re-creation of our results. For each of the data sets, we selected a set of at least 5 QI elements, an SI element and a context element (the configuration). As we are measuring runtime performance and memory usage, it is therefore not important for the Dissection algorithm which elements are QI or SI. The Anatomy algorithm that is run for comparison is run with the same configuration.

3.2.1. Dissection runtime performance

The Dissection algorithm parses an XML document and, in real-time, identifies QI and SI fragments of XML data that it copies into two separate files. This is achieved by keeping track of the document path that is currently being traversed, which provides the detail about the ancestry of a given node (i.e. the fragment information). We run the Dissection algorithm against all three XML documents with variable number of QI elements (1–5) and group sizes (10–50). QI and SI elements are leaf elements in the document schemas that contain data, while the context element is an element that is repeated throughout the XML document.

We first examine the effect of the number of QI elements in the runtime performance measure. In Fig. 11a, we vary the number of QI elements from 1 to 5, while the group size is set to 10, and examine the total runtime per dataset. The same tests are repeated for different group sizes, which we vary from 10 to 50, and results reported in Fig. 11b.

In the Dissection method, the number of QI elements has a notable effect on the runtime performance, as opposed to the size of the groups. The reason for this is because for each QI, the program must track and update the ancestry path, which costs additional time such as for element name comparisons. We observe that for the larger data sets this can make a significant difference. The parameter on the *x*-axis is then changed to group size and we observe that it has no notable influence of the runtime as the number of QI elements. This is because the ancestry path computations are the same in each experiment, as the group size is set to 10. It therefore consumes the same amount of time, no matter how large the group size is.

Note on technical details: Both the Dissection and Anatomy method is implemented as a SAX (simple API for XML) parser handler, which means it streams the XML data and merely writes it to the respective output files, namely for QI and SI elements and groups. In Dissection, no analysis of QI, SI or context node data values is required, so the only memory usage is for the ancestry path element names, and is insignificant. We have therefore decided not to report on memory usage for the Dissection method. For Anatomy, the SAX handler also processes the streaming (private) XML data in real-time, but requires more time and memory to perform the computations required to achieve diversification of SI values within each group. We report on these measurements in the subsections below.

3.2.2. Anatomy runtime performance

To show how the dissection of XML documents performs when a privacy property is used, we choose the Anatomy [6] method. Anatomy first introduced the dissection method to achieve privacy in relational data, with an additional constraint with respect to the diversity of SI values within each group. Since dissection is the basis for our XML dissection method, we believe that benchmarking against Anatomy provides a fair comparison. A direct performance comparison between Dissection and Anatomy with respect to runtime is reported in Fig. 13.

As stated earlier in the paper, our method will work with any privacy protection property that is based on dissection, but we believe it is sensible to create a performance baseline based on the approach that initially introduced the dissection method. We therefore provide another set of experiments that runs the Anatomy algorithm against the XML documents. In these experiments, the data is dissected and, in each group of n group elements, there are exactly n different SI values in the best case. Note, that these experiments are not measuring the effectiveness of privacy preservation in XML (this will be addressed in the next subsection), but it gives an indication of how much runtime is required to achieve dissection in XML and therefore gives an idea of the efficiency of the dissection method for XML.

[7] http://www.cs.washington.edu/research/xmldatasets/www/repository.html.

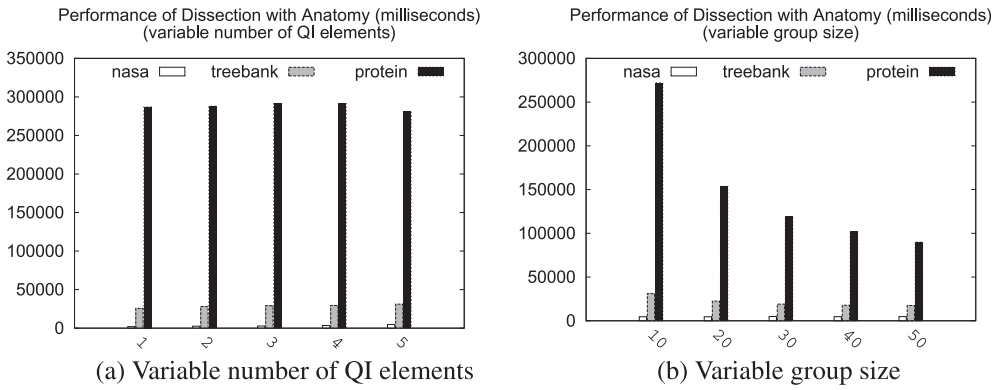

Fig. 12. Anatomy runtime performance.

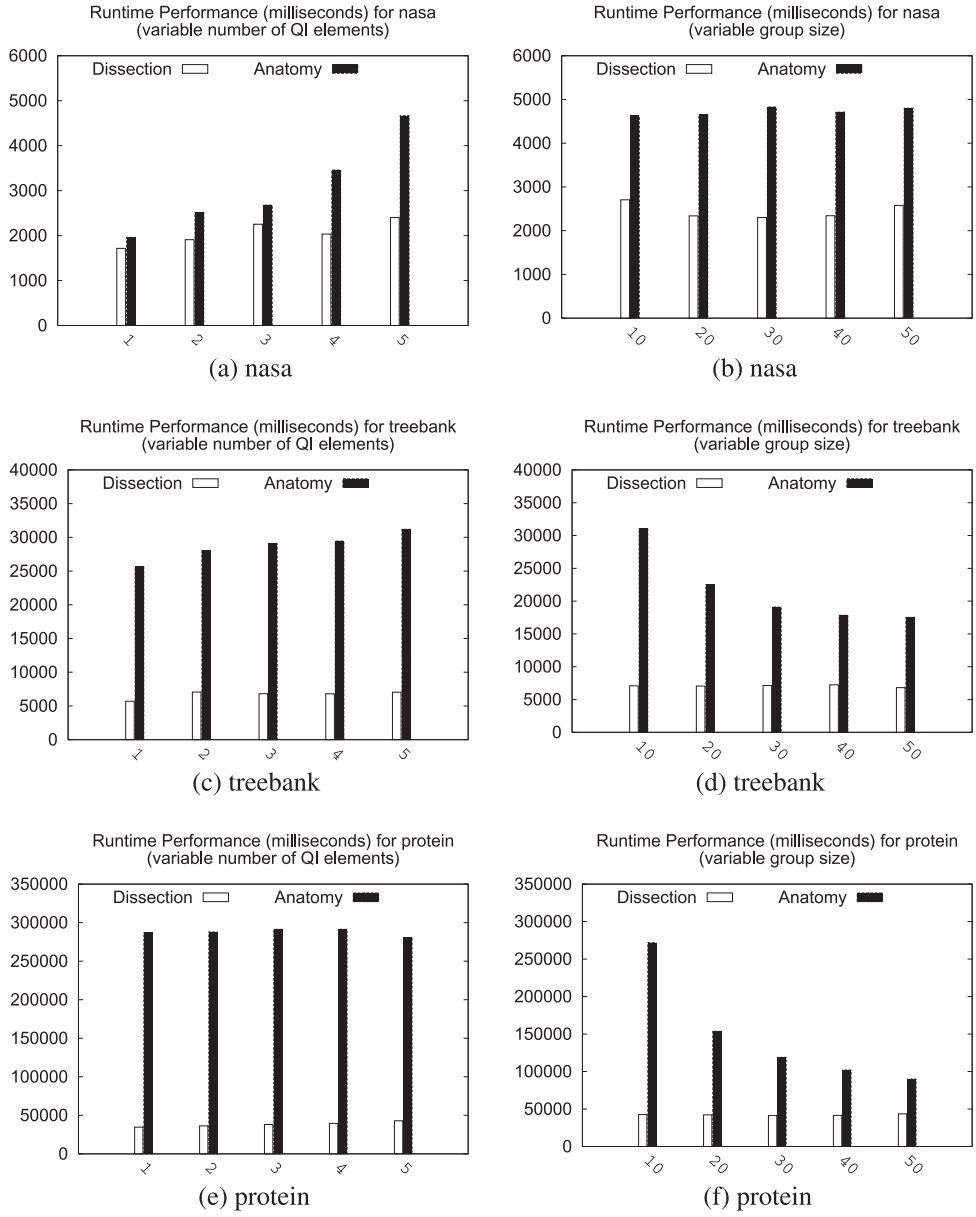

Fig. 13. Runtime comparison between Dissection and Anatomy.

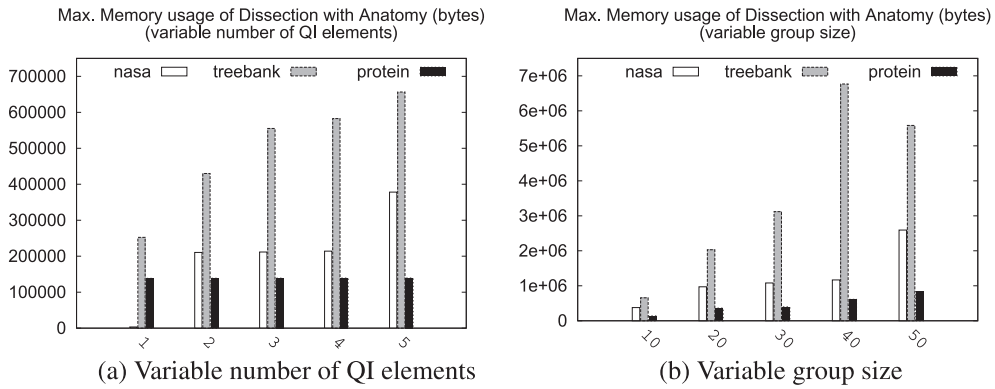

Fig. 14. Anatomy memory usage.

Fig. 12a reports the runtime performance of our XML dissection method when the Anatomy privacy protection method is used. In our experiment, the streaming XML data is analysed in real-time and grouped in a way so that each group contains distinct values. We observe that the algorithm is near-agnostic to the number of QI elements. This is due to the fact that for Anatomy, QI values play no role in how to create groups, and the constant runtime performance confirms this. We believe that the slight variation in runtime is due to the distribution of SI values within the XML data, and the circumstance that n distinct SI nodes may not appear in order throughout the document all the time. Zhou et al. analyse this issue in their paper on Continuous Privacy Preserving Publishing of Data Streams [32]. We decide to investigate the effects of streaming XML for privacy preservation as part of future work.

We examine runtime performance for Anatomy with varying group size and observe a decline in time required to create groups. The results are reported in Fig. 12b. This decline in runtime is because the group creation phase involves sorting a priority queue in descending order, adding significant overhead to the group creation. Therefore, with increasing group size, and therefore less groups per XML document, this overhead becomes less and less of a performance issue. Due to the scale, this trend is best visible for the protein database but also exists for the other two datasets.

3.2.3. Anatomy memory usage

For Anatomy we report maximum memory usage required for varying number of QI elements (see Fig. 14a) and group sizes (see Fig. 14b). We calculate memory usage based on the byte size of XML fragments that need to be kept in memory. When a new group is created, memory is freed up because XML fragments can be written to the output files and removed from the program memory. The memory usage for a relational Anatomy algorithm is small and is linear with respect to increasing QI or group size because all that needs to be stored in memory is a pointer to the table rows that may be added to a group when it can be created. In an XML scenario, the program must keep in memory the ancestry path to the QI and SI nodes. A factor that influences the memory requirements for applying Anatomy on XML documents is the structure of the XML schema and document.

We observe that for the *nasa* data set there is a notable increase in memory use (see Fig. 14a) which is caused by the depth of the document tree structure (see protein dataset specifications in Fig. 6). The max-depth of the protein dataset is nearly 5 times more than the other two data sets. As a result, more intermediary nodes must be stored in memory where QI or SI are located deep in the document tree structure. This also explains the sudden spike between QI = 4 and QI = 5 of the nasa dataset in Fig. 14a, where the first 4 QI elements are siblings and the 5th element is deep in the tree.

By contrasting Figs. 12b and 14b we observe that there is a tradeoff between speed and memory usage for achieving dissected and *anatomised* XML documents in real-time. With increasing group sizes there are more temporary XML fragments that must be kept in memory until groups can be created, while the savings in runtime are due to the reduced re-sorting of the priority queue as described earlier. We conclude with a remark on runtime performance and memory use of our method. If required, more memory can be saved by not storing the actual XML fragments in main memory but on a disk, only keeping pointers in memory. However, this will inevitably increase runtime. The fact that our implementation streams XML data, there is no possibility to "backtrack" through the XML document to re-create the XML fragments. On the other hand, it is possible to re-create an XML fragment based on the ancestry path to an QI or SI element, given that all element names, attributes and their values are temporarily stored. The saving is due to the fact that the path will only need to be stored once per QI or SI element.

4. Discussion

We implemented our proposed XML dissection method and the δ-dependency privacy model and algorithm. We performed a series of experiments, both on real XML datasets and on synthetic data, and analysed the performance of the dissection method and the degree δ of privacy for four key variables. Experiments conducted on real data were analysed in respect of varying number of QI elements and group sizes. The four sets of experiments on synthetic data have been analysed across two dimensions, being depth of the hierarchy T_S and node fan-out values of T_S. These two sets of parameters give a coverage of different data sets with varying sizes, data distribution and semantic complexity.

The objectives of the experiments were to test that:

1. Dissection based data privacy is efficient and practical for the XML domain; and
2. The δ-dependency privacy scheme provides additional protection beyond current privacy preserving properties.

To test (1) we wanted to show that runtime and memory usage are acceptable to warrant a real life application of XML dissection. This is particularly of concern in scenarios where resources such as CPU and memory are limited, such as mobile devices that are increasingly used in place of traditional computers. Although XML can have a far more complex structure than relational data, it was shown that our dissection algorithm has linear runtime

and memory increase, so it is easy to predict how the privacy model would behave under different parameters including number of QI and group size. As pointed out earlier, there is still much room for improvement of the algorithm to make it more suitable for resource constrained environments.

To test (2) it was necessary to identify parameters that could be used to describe XML data structures of various semantic complexity and size. Experimentation with these parameters enabled us to test that data sets could be re-arranged so that all new groupings (QI and SI groups) meet the requirements of established and proven privacy schemes such as Anatomy, and extend these by incorporating our novel δ-dependency privacy constraint. The experiment results show that, when the δ-dependency algorithm is applied to data sets of different semantic diversity, data distribution and data size, and re-arranged into groups of different sizes, a near-constant δ privacy property can be achieved, which gives rise to sufficient reliability of the approach. We showed that δ-dependency prevents semantic relationships within the sensitive information of the data to be used as a means to infer a connection between sensitive information and individuals' quasi identifiers.

5. Conclusions

We have identified that current privacy models for XML are mainly based on access control and filtering techniques, and that these approaches are not sufficient to ensure adequate privacy protection in XML data. Further, we showed that while a number of strong privacy preserving properties are available in relational data, these have not yet been applied to XML. Taking these research challenges, we developed a privacy model for XML based on the dissection method and a new privacy propery δ-dependency. We proved that our model exceeds existing privacy preserving properties as found in relational data and is practical in application from a performance point of view.

References

[1] Williams MH, Venters G, Venters G, Marwick DH. Developing a regional healthcare information network. IEEE Trans Inf Technol Biomed 2001;5(2):177–80. <http://dblp.uni-trier.de/db/journals/titb/titb5.html#WilliamsVVM01>.
[2] Virtanen T. The Finnish national eHealth archive and the new research possibilities. In: Nursing informatics; 2009. p. 688–91.
[3] Huang E-W, Liou D-M. Performance analysis of a medical record exchanges model. IEEE Trans Inf Technol Biomed 2007;11(2):153–60.
[4] Sweeney L. k-Anonymity: a model for protecting privacy. Int J Uncertain Fuzziness Knowledge-Based Syst 2002;10(5):557–70.
[5] Machanavajjhala A, Gehrke J, Kifer D, Venkitasubramaniam M. l-Diversity: privacy beyond k-anonymity. In: ICDE; 2006. p. 24.
[6] Xiao X, Tao Y. Anatomy: simple and effective privacy preservation. In: VLDB; 2006. p. 139–50.
[7] Li N, Li T, Venkatasubramanian S. t-Closeness: privacy beyond k-anonymity and l-diversity. In: ICDE; 2007. p. 106–15.
[8] Fan L, Xiong L. An adaptive approach to real-time aggregate monitoring with differential privacy. IEEE Trans Knowledge Data Eng 2012;99(PrePrints):1. <http://doi.ieeecomputersociety.org/10.1109/TKDE.2013.96>.
[9] Loukides G, Gkoulalas-Divanis A, Shao J. Efficient and flexible anonymization of transaction data. Knowledge Inf Syst 2013;36(1):153–210.
[10] Xu Y, Wang K, Fu AW-C, Yu PS. Anonymizing transaction databases for publication. In: KDD; 2008. p. 767–75.
[11] Loukides G, Gkoulalas-Divanis A, Malin B. Anonymization of electronic medical records for validating genome-wide association studies. Proc Natl Acad Sci 2010;107(17):7898–903. http://dx.doi.org/10.1073/pnas.0911686107.
[12] Loukides G, Gkoulalas-Divanis A, Malin B. Coat: constraint-based anonymization of transactions. Knowledge Info Syst 2011;28(2):251–82. http://dx.doi.org/10.1007/s10115-010-0354-4.
[13] Sladic G, Milosavljevic B, Konjovic Z, Vidakovic M. Access control framework for XML document collections. Comput. Sci. Inf. Syst. 2011;8(3):591–609.
[14] Müldner T, Leighton G, Miziolek JK. Parameterized role-based access control policies for XML documents. Info Secur J: A Global Perspect 2009;18(6):282–96.
[15] Kocatürk MM, Gündem TI. A fine-grained access control system combining MAC and RBACK models for XML. Inf Lith. Acad. Sci. 2008;19(4):517–34.
[16] Zhang H, Zhang N, Salem K, Zhuo D. Compact access control labeling for efficient secure XML query evaluation. Data Knowledge Eng. 2007;60(2):326–44.
[17] Yang X, Li C. Secure XML publishing without information leakage in the presence of data inference. In: VLDB; 2004. p. 96–107.
[18] Bertino E, Castano S, Ferrari E, Mesiti M. Specifying and enforcing access control policies for XML document sources. World Wide Web 2000;3(3):139–51.
[19] A.H. Landberg, J.W. Rahayu, E. Pardede, Privacy-aware access control in XML databases. In: ADC; 2010. p. 85–92.
[20] Vaidya J, Zhu Y, Clifton CW. Privacy preserving data mining. Advances in information security, vol. 19. Springer; 2006.
[21] Xiao X, Tao Y. Personalized privacy preservation. In: SIGMOD conference; 2006. p. 229–40.
[22] Meyerson A, Williams R. On the complexity of optimal k-anonymity. In: PODS; 2004. p. 223–28.
[23] Samarati P, Sweeney L. Generalizing data to provide anonymity when disclosing information (abstract). In: PODS; 1998. p. 188.
[24] Ciriani V, di Vimercati SDC, Foresti S, Samarati P. k-Anonymity. In: Secure data management in decentralized systems. US: Springer; 2007. p. 323–53.
[25] Câmpan A, Truta TM. Extended p-sensitive k-anonymity; 2006. p. 30.
[26] Bertino E, Cuzzocrea A. Privacy preserving OLAP over distributed XML documents. In: ICPP workshops; 2009. p. 603–11.
[27] Wong RC-W, Li J, Fu AW-C, Wang K. (alpha, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In: KDD; 2006. p. 754–9.
[28] Gal TS, Chen Z, Gangopadhyay A. A privacy protection model for patient data with multiple sensitive attributes. IJISP 2008;2(3):28–44.
[29] Augsten N, Barbosa D, Böhlen MH, Palpanas T. Efficient top-k approximate subtree matching in small memory. IEEE Trans Knowledge Data Eng 2011;23(8):1123–37.
[30] Cohen S. Indexing for subtree similarity-search using edit distance. In: SIGMOD conference; 2013. p. 49–60.
[31] Landberg AH, Rahayu JW, Pardede E. n-Dependency: dependency diversity in anatomised microdata tables. Logic J IGPL 2011;19(5):679–702.
[32] Zhou B, Han Y, Pei J, Jiang B, Tao Y, Jia Y. Continuous privacy preserving publishing of data streams. In: EDBT; 2009. p. 648–59.