Transforming Provenance using Redaction

Tyrone Cadenhead, Vaibhav Khadilkar,
Murat Kantarcioglu and Bhavani Thuraisingham

The University of Texas at Dallas

ACM Symposium on Access Control Models and Technologies (SACMAT)

June 15-17, 2011

What is Provenance ?

- Provenance records the history of a document
  - E..g., is the lineage or pedigree of a resource
- Metadata about the origin and history of a piece of item
  - annotations about data items
  - account of the history affecting data items
  - Takes the form of directed graph
  - Captures the causality among documents
- Provenance determines the trustworthiness of shared information.

Why provenance is important?

- Provenance is essential for various domains
- Utility of the information shared in these domains relies on
  - quality of the information and
  - mechanisms that verify the correctness of the data
- **Potential Application:**
  - ***In healthcare***: tracks the activities of healthcare professionals, regulatory compliance
  - ***In E-science***: replicates experiments and verify the steps and the results
  - ***In business***: provides an audit trail, which can be used for accountability
  - ***In intelligence***: verifies the sources of information
  - ***In courts***: provides trace and evidence
  - ***Data quality***: estimates data reliability and trustworthiness

Why Redaction is needed?

- Health care:
  - Electronic patient record log of all activities
    - e.g., patient visits to a hospital, diagnoses and treatments for diseases, and processes performed by healthcare professionals on a patient
  - Shared among several stakeholders
    - e.g., researchers, insurance and pharmaceutical companies
- Ease of information sharing presents a risk of information misuse
- **One possible answer: Redaction**
  - **Circumvent the sensitive from released information**

What is Redaction?

- Redaction policies
  - completely or partially remove sensitive attributes of the information being shared
- Commercially available redaction tools
  - block out (or delete) the sensitive parts of documents available as text and images
- But current tools only apply over single resources
  - Not to provenance
  - Not to directed graphs

Provenance Graph

- **Features:**
  - ***Directed edges*** indicating that an event happened before another event
  - ***Causal dependencies*** between the node entities
    - Edges start at a node called ***the effect*** and points to another node called ***the cause*** of the event
  - ***Acyclic***, indicating that history is non-cyclic and immutable

- It can be build using RDF
  - RDF triple (s, p, o)
    - represented graphically as $s \xrightarrow{p} o$
    - s is causally dependent on o
    - s as the effect and o as the cause of s

Example: OPM

- The OPM model identifies three categories of entities
  - artifacts, processes and agents

- Abstract vocabulary describe relationships between the entities

  RDF Triples (examples):

  <opm:Process> <opm:WasControlledBy> <opm:Agent>

  <opm:Process> <opm:Used> <opm:Artifact>

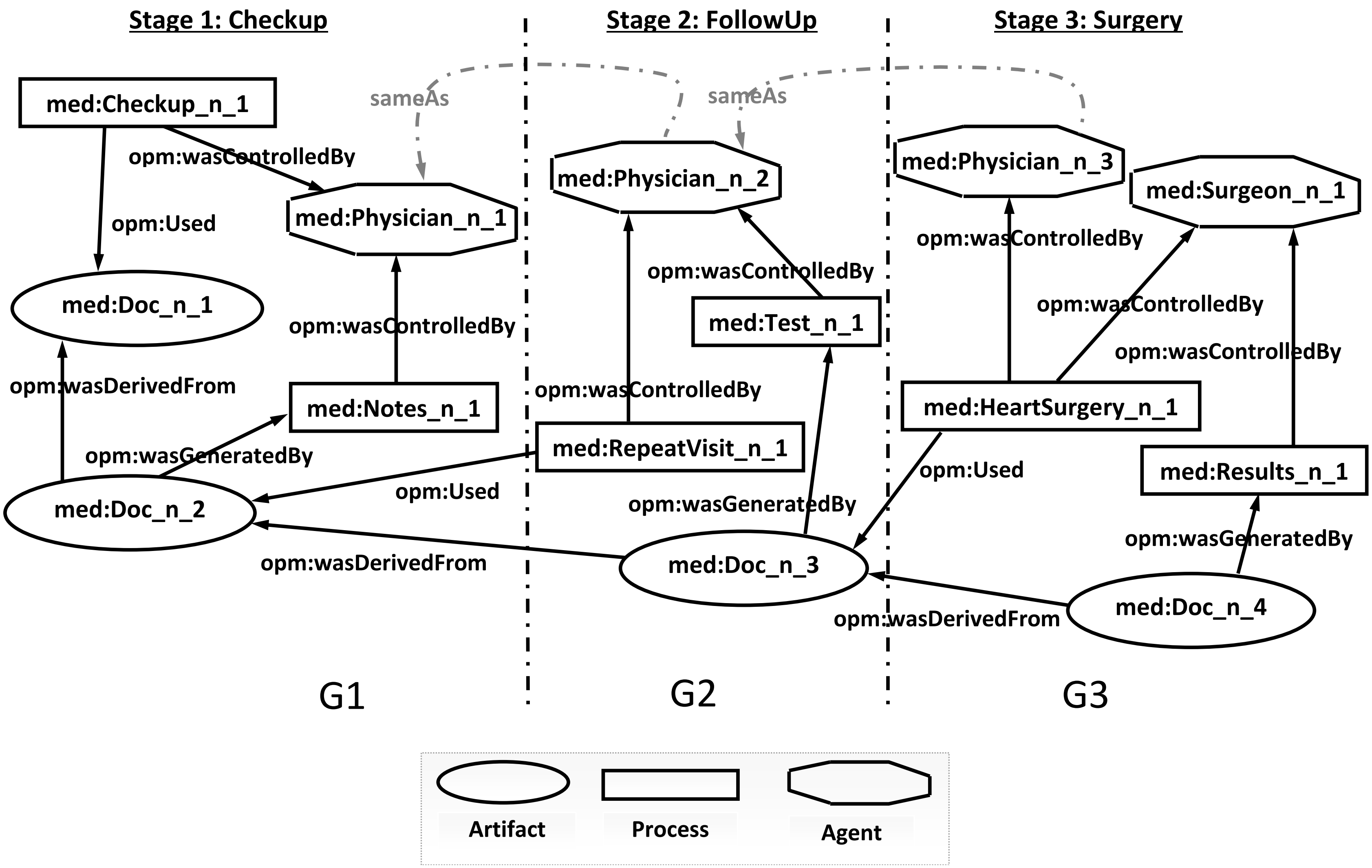
  <opm:Artifact> <opm:WasDerivedFrom> <opm:Artifact>

  <opm:Artifact> <opm:WasGeneratedBy> <opm:Process>

- Let **V** ={WasControlledBy, Used, WasDerivedFrom, WasGeneratedBy, WasTriggeredBy}

  Path (<s_1> (P) <o_n>)
  - Define **P** over **V** using regular expressions
  - (x, [p]*, y) and (x,[p]+, y)

Provenance Graph *(in healthcare domain)*



Graph Grammar

- **Goal:** Transforms an original graph to one that meets the requirements of a set of redaction policies
- **Approach:** Use graph grammar (or a graph rewriting system)
- Two steps to apply redaction policies over general directed labeled graphs:
  - Identify a resource in the graph that we want to protect. This can be done with a graph query (i.e. a query equipped with regular expressions).
    - $Gq \leftarrow q(G)$, a query q over provenance graph G produces a subgraph, Gq , in response to the query.
  - Apply a redaction policy to this identified resource in the form of a graph transformation rule.
    - $Gr \leftarrow P(G)$, where P is written in a rule language.

Graph Rewriting System

- Redaction policies are used to protect sensitive information in resources
- Formulate policies in our graph grammar system as production rules
  - In order to identify and remove any sensitive (e.g. proprietary, legal, competitive) content in these resources.
  - Policies are a formal specification of the information that must not be shared.
- Production rules are one of the following graph operations:
  - a vertex contraction, or an edge contraction, or a path contraction or a node relabeling operation.

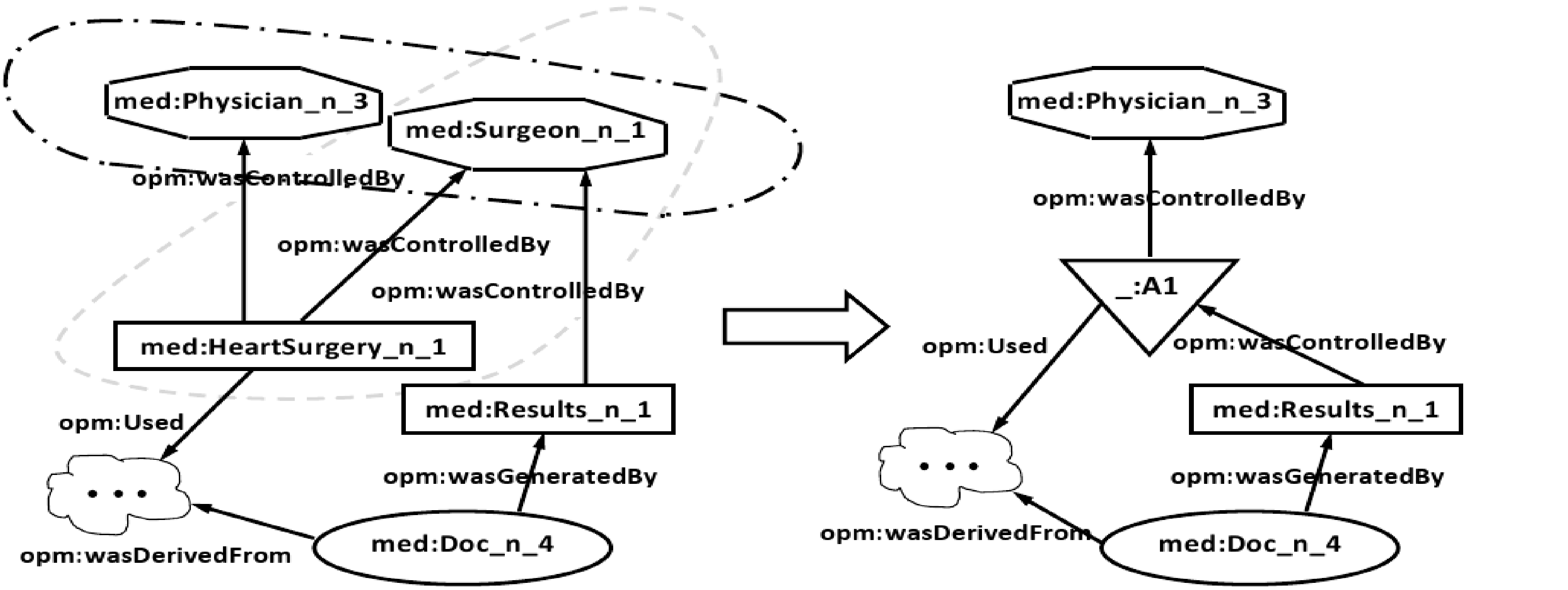
Graph Rewriting System

- *A graph rewriting system is a three tuple, $(G\ell, q, P)$*
  - *$G\ell$ is a labeled directed graph*
  - *q is a request on $G\ell$ that returns a subgraph Gq*
  - *P is a policy set*
- *For every policy $p = (r, e)$ in P, $r = (se, re)$ is a production rule*
  - *e is an embedding instruction*
- *A production rule, $r : L \rightarrow R$ where L is a subgraph of Gq and R is a graph*
  - *We also refer to L as the left hand side (LHS) of the rule and R as the right hand side (RHS) of the rule*
- *During a rule manipulation, L is replaced by R and we embed R into $Gq - L$*
- *Embedding Information, e:*
  - *This specifies how to connect R to $Gq - L$*
  - *Gives special post-processing instructions for graph nodes and edges on the RHS of a graph production rule*

Operations on Provenance

- **Basis for operations: Edge contraction**
- Edge contraction serve as the basis for deﬁning vertex contraction and path contraction:
  - Let G = (V, E) be a directed graph containing an edge e = (u, v) with v = u. Let f be a function which maps every vertex in $V \setminus \{u, v\}$ to itself, and otherwise maps it to a new vertex w.
  - The contraction of e results in a new graph $G' = (V', E')$, where $V' = (V \setminus \{u, v\}) \cup \{w\})$, $E' = (E \setminus \{e\})$, and for every $x \in V$, $x' = f(x) \in V$ is incident to an edge $e' \in E'$ if and only if the corresponding edge, $e \in E$ is incident to x in G.
- Vertex contraction: replace two arbitrary vertices *u, v* and an edge drawn between them with a new vertex *w*.
- Path contraction: each edge is processed in turn until we reach the last edge on the path.
- Edge contraction may be performed on a set of edges in any order.
- Contractions may result in a graph with loops or multiple edges. In order to maintain the deﬁnition of a provenance we delete these edges.

Contraction Example



- This vertex contraction could show for example how a third party is prevented from knowing the identities of agents (i.e., surgeon) who controlled the processes (i.e., a heart surgery and a logging of results of a surgery into a patient's record).

Path Contraction

- This occurs upon a set of edges in a path that contract to form a single edge between the end points of the path.
- Edges incident to vertices along the path are either eliminated, or arbitrarily connected to one of the endpoints.

- Example: A path contraction is necessary when we want to prevent the release of the history of patient 1 prior to surgery as well as the details of the surgery procedure.

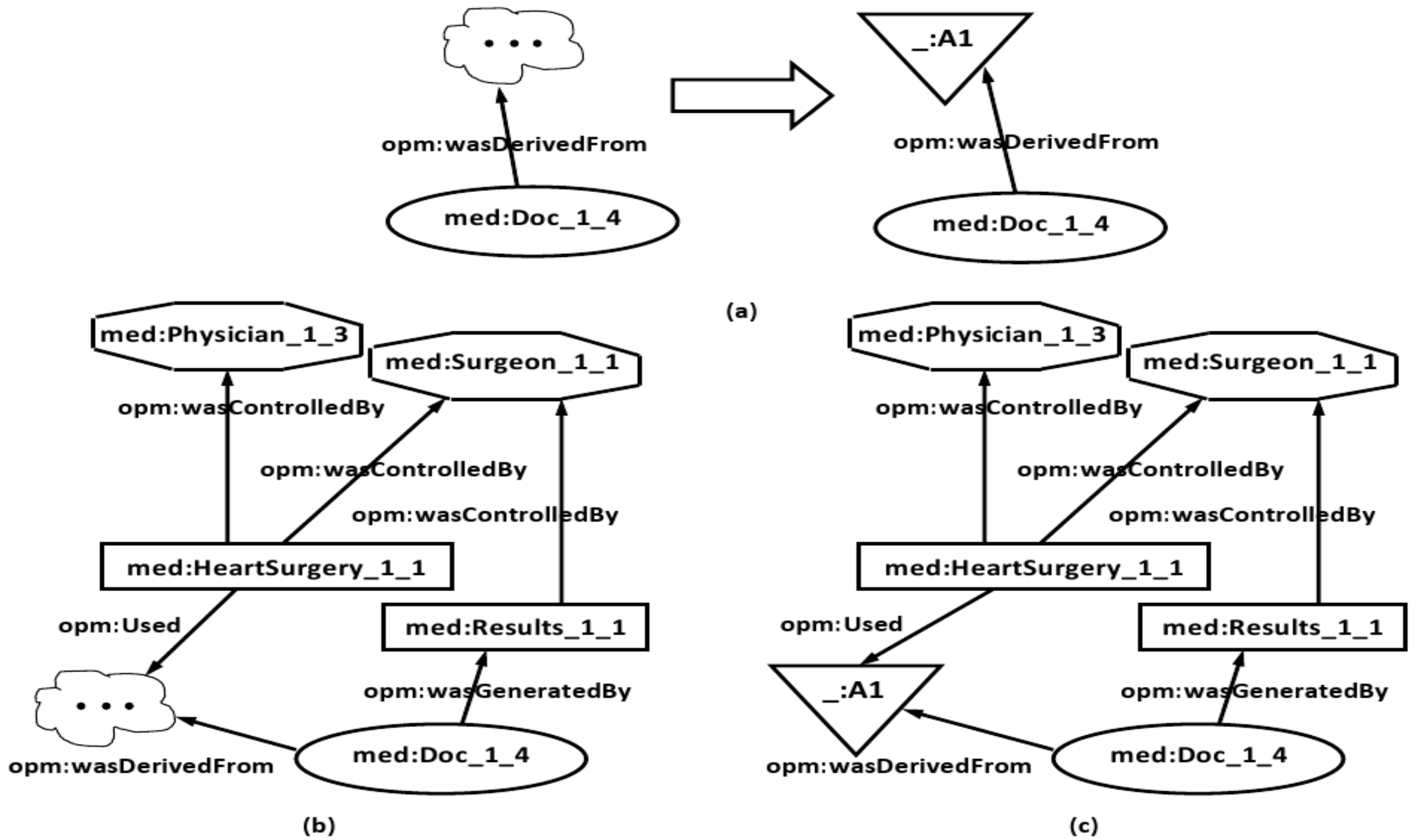
Node Relabeling

- A node relabeling operation replaces a label in a node with another label.
- This is generally a production rule whose LHS is a node in Gq and whose RHS is also a node normally with a new label.

- Our example nodes have generic labels but in practice each entity would be annotated with contextual information.
  - This information serves as identifiers for the respective entity.

- Before sharing information about these entities it is imperative that we remove sensitive identifiers from them.
  - For example, a physician's cell phone number and social security number are considered unique identifiers and these should be redacted whenever this physician's identity is sensitive.

Graph Transformation Step



Utility Aware Redaction

- Since rules could be applied in any order, heuristics need for ordering
- We choose three conventions for pre-ordering the production rules:
  - the original ordering (*OO*);
  - lowest to highest utility (*LHO*);
  - highest to lowest utility (*HLO*).
- We believe that provenance is more useful when it is least altered, therefore define utility as:

$$\left(1-\frac{\text{altered triples}}{\text{original triples in Gq}}\right)\times 100$$

Policy Language

- A Proposed Policy Specification

```
<policy ID="1" >
   <lhs>
        start=Doc1_4
        chain=[WasDerivedFrom]+ artifact AND
        artifact [WasGeneratedBy] process AND
        process [WasControlledBy] physician|surgeon.
        start=RepeatVisit1_1
        chain=[Used][WasControlledBy].
        start=Checkup1_1
        chain=[Used][WasControlledBy].
  </lhs>
  <rhs>_:A1</rhs>
  <condition>
   <application>null</application>
   <attribute>null</attribute>
  </condition>
  <embedding>
   <pre>null</pre>
   <post>(HeartSurgery_1_1,Used, _:A1)</post>
  </embedding>
</policy>
```

Policy Language

- lhs element describes the left hand side of a rule.
- rhs element describes the right hand side of a rule.
- starting entity
  - Each path in the lhs and rhs begins at a starting point.
- condition element has two optional sub elements,
  - the application defines the conditions that must hold for rule application to proceed,
  - the attribute element describes the annotations in *LHS*.
- Embedding element has two optional sub elements,
  - pre describes how *LH S* is connected to the provenance graph
  - post describes how *RH S* is connected to the provenance graph.

Policy Translation

- **Translate Policy Specification**
  - Executed over a preferred Data format
    - Format represent and store provenance
  - Parsed into a query over the provenance graph
    - Query support for regular expressions

- **A Semantic Approach:**
  - RDF data model
    - Support Graph Structure
  - SPARQL
    - Query the RDF graph
  - Support Graph Querying
    - SPARQL + Regular Expressions

Experiments

- Implement prototype using the following open sources technologies:
  - *Jena*
  - *XML 1.0*
  - *Java 1.6*
  - *Gleen regular expression library*
  - *The OPM toolbox*

- Experiments are executed on an IBM workstation with 8 X 2.5GHz processors and 32GB RAM

Experiments

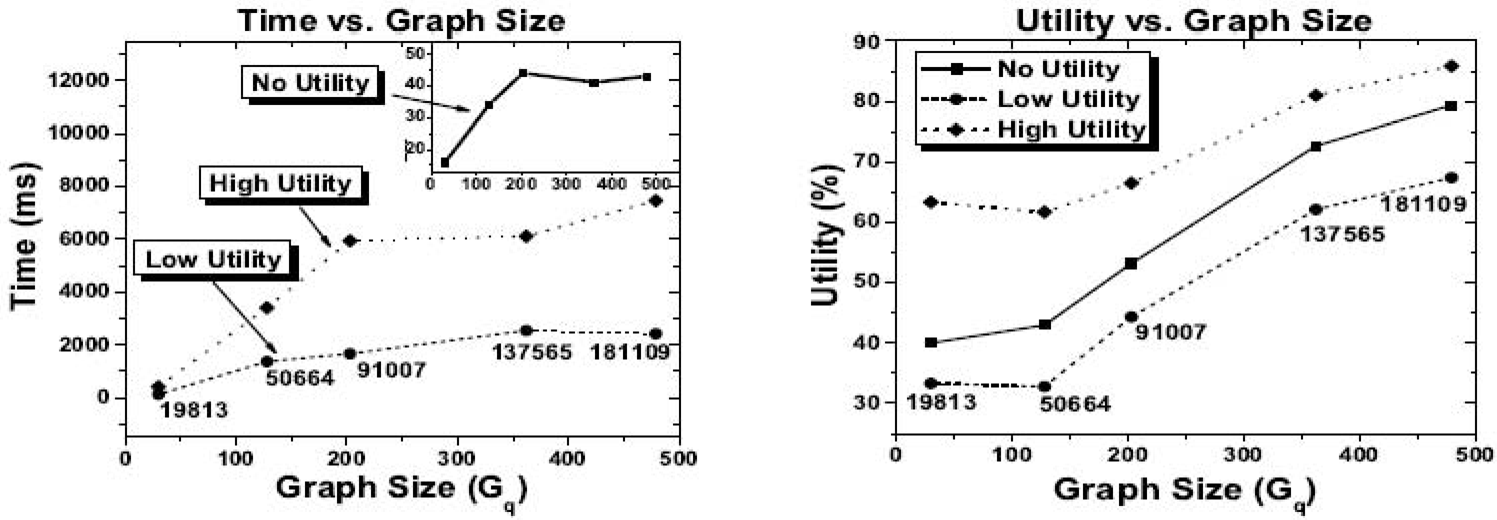


Figure 5: Comparison of Redaction Time and Utility vs. Graph Size

Experiments

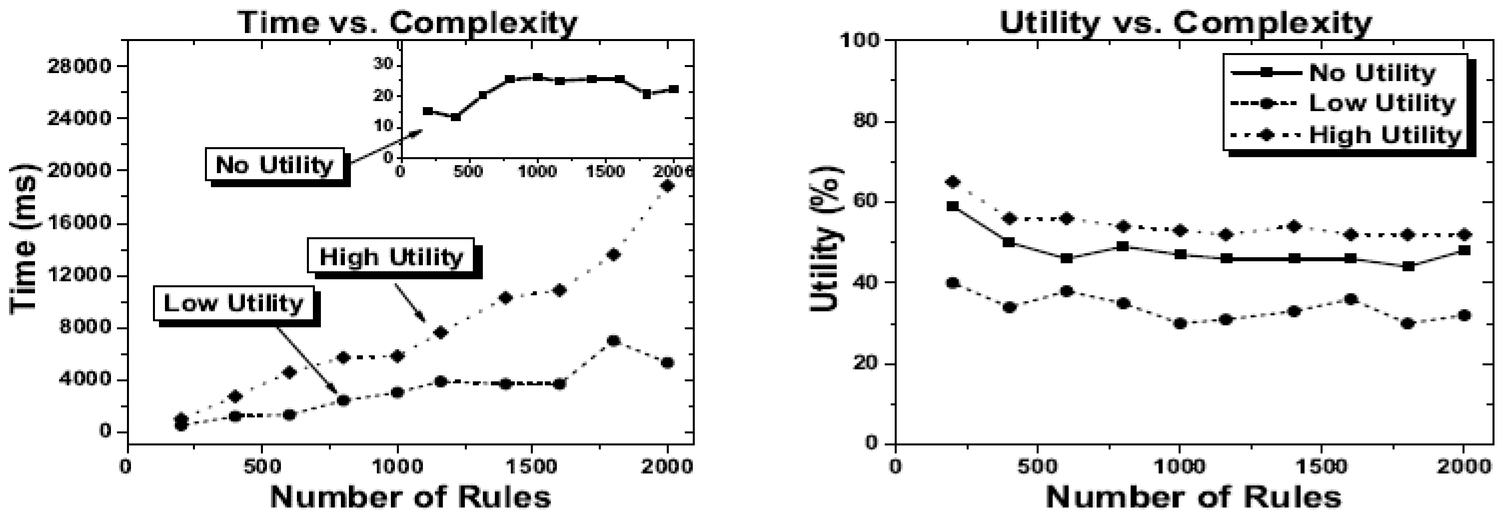


Figure 6: Experimental Comparison of Complexity

Conclusions

- We proposed the first automated redaction policy tools for provenance
- Many issues need to be addressed further
  - Privacy
  - Inference
  - Usability