IARPA Project - Policy Compliance for PIR Queries

Lalana Kagal
MIT CSAIL
Decentralized Information Group

Overview

- Problem Statement
- PIR Conceptual Operations
- Assumptions
- Current results
- Next steps

Problem Statement

- Proving that queries made by the client conform to mandated policies preventing leakage of unauthorized information where both the query and query results are unknown to the database server
  - What kind of language should be used to express these policies ?
  - What tools and techniques will help encourage rule-following and identify non-compliance ?

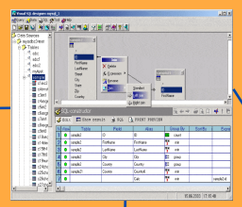
Private Information Retrieval (PIR) Concept of Operations

Analyst

A

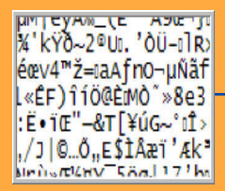
Submit plaintext query

PIR Client Front End



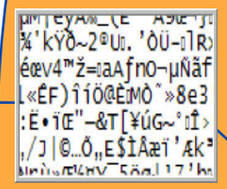
B

compile/encrypt query



Submit encrypted query to PIR DBMS agent

PIR DBMS Front End



C

compile/ encrypt DB content

Information Repository

D

Compare encrypted query against encrypted DB content

E(x)*E(y) = E(x+y)

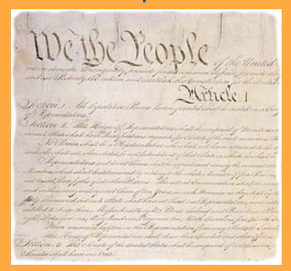
E

Generate encrypted search results

Return encrypted search results to PIR client agent

G

Monitor legal/ regulatory compliance of search criteria (plaintext queries)



F

Decrypt query results

Return plaintext results

Mohammed Atta
Khalid S. Mohammed
Marwan al-Shehhbi

Note:
DB may be replicated and distributed

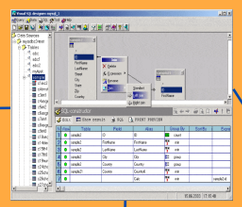
Private Information Retrieval (PIR) Concept of Operations

Analyst

A

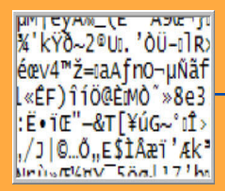
Submit plaintext query

PIR Client Front End



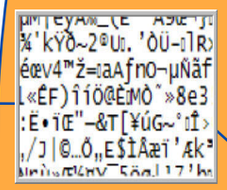
B

compile/encrypt query



Submit encrypted query to PIR DBMS agent

PIR DBMS Front End



C

compile/ encrypt DB content

Information Repository

D

Compare encrypted query against encrypted DB content

E(x)*E(y) = E(x+y)

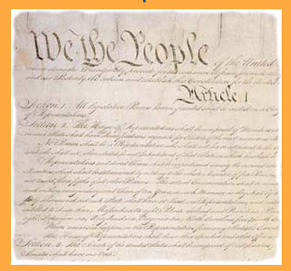
E

Generate encrypted search results

Return encrypted search results to PIR client agent

G

Monitor legal/ regulatory compliance of search criteria (plaintext queries)



F

Decrypt query results

Return plaintext results

Mohammed Atta
Khalid S. Mohammed
Marwan al-Shehhbi

Note:
DB may be replicated and distributed

Assumptions

- Threat Model
  - honest but curious
  - we are not dealing with malicious users or with two or more users colluding
- The client sends the policy reasoner a set of plaintext queries
- The policy reasoner will NOT have access to the database or the query results
- The policy reasoner will have access to the meta-data about the database
  - no. of rows, no. and name of cols, and col categorization (e.g. gender can have two values)
  - we will be given only col categorization to begin with because distribution would release too much information

Assumptions

- The policy reasoner will be told about the requester is so we can check policies of the sort "user x cannot access data a but user y can"
- Query history
  - The reasoner will store all queries provided within a query set and use it as “query history”
  - Depending upon the order in which the queries are made the result (compliance/non compliance) will be different

First test: April 2010

- First phase
  - query set + identity of requester is input
  - list of non-compliant queries + justification is output
  - Efficiency will be measured as number of queries we can process per second or other metric. But not used to test us.
- Policy assurance metric must be at least .80

$$\frac{N}{N_{correct} + 1.5*N_{fp} + 2*N_{fn}}$$

where,
N is total number of queries
$N_{correct}$ is the number of queries correctly classified
N_{fp} is the number of queries incorrectly classified as violating policy
N_{fn} is the number of queries incorrectly classified as conforming to policy

Policy Assurance metric

- Second Phase metric should be .87 and Third Phase .98

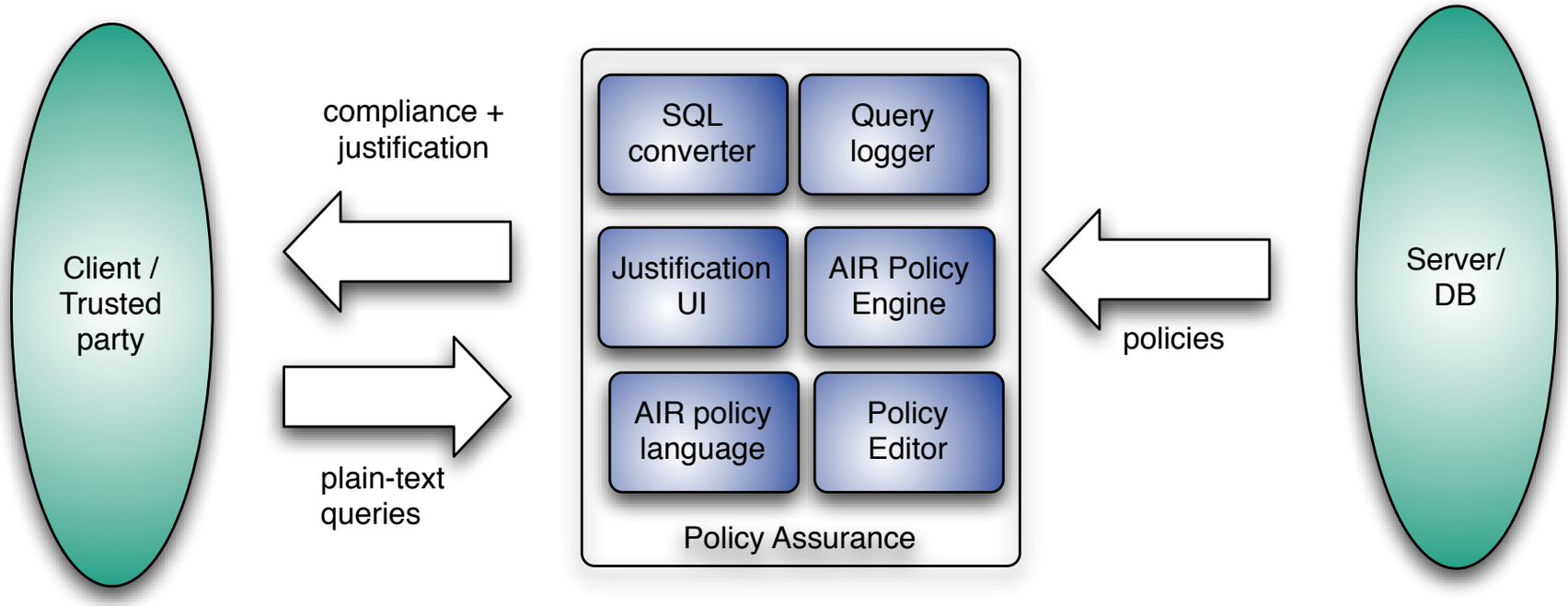
Challenges in Policy Assurance

- Need to support the query language (SPARQL)
- Not enough to just restrict certain keywords or rows or cols; policies are ambiguous
  - For example, “Access to SSN is not permitted”.
  - Does this mean that SSN values cannot be retrieved or does it also include use of SSN values to filter the results
- Policies tend to deal in abstract terms and talk about kinds of information that should not be accessible or should not be used for certain purposes
  - For example, “Access to contact information for minors is not permitted”, or “my health information cannot be used to contact me regarding experimental drugs”

Challenges in Policy Assurance

- Though individual queries might not violate privacy policies, a certain combination of queries might lead to a violation.
  - For example, an analyst might generate a query that yields a target's alias and then query a different database that establishes the true identity associated with that pseudonym. If that real identity reveals that the target is a US Person, then certain subsequent queries would violate various laws and executive orders, even though a simple analysis of the query itself would not reveal a violation.
- need history-based queries

Assurance Architecture



Policy Assurance Components

- Query logger
- AIR Policy language
  - a machine-understandable policy language for expressing privacy policies
  - Semantic Web technologies for shared model of data
- AIR Reasoning engine
  - for reasoning over queries and policies to identify violations
  - justifications
  - Handle private policies

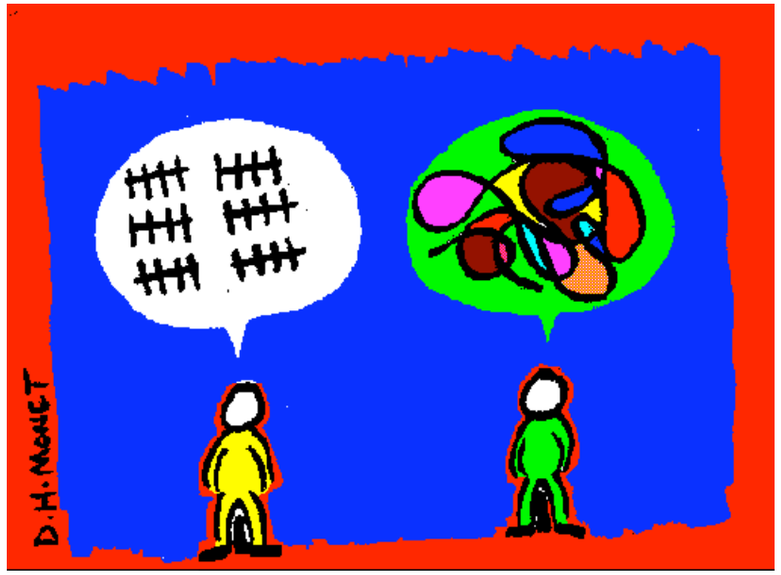


Image courtesy of http://home.ca.inter.net/~dmonet/

Policy Assurance Components

- Justification User Interface
  - Why UI ?
  - Graphical justification interface that will provide a structured natural language explanation for policy non-compliance
- SQL converter
  - Convert SQL into format understandable by AIR reasoner
- Policy development
  - Support definition of high level policies de-coupled from query and database structure

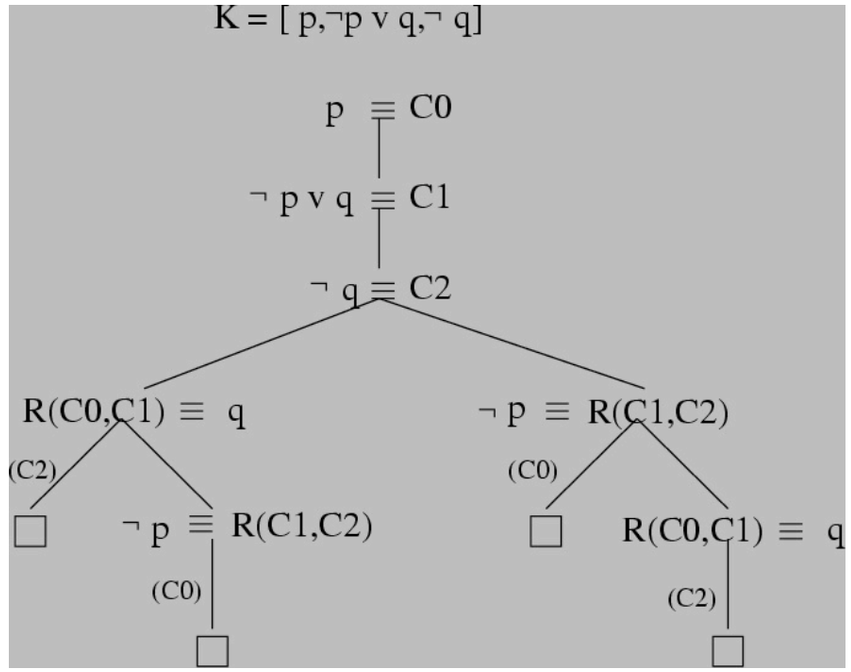


Image courtesy http://clip.dia.fi.upm.es/~logalg/slides/

Current research

- Policies based on SPARQL structure
- Use case development
  - Policy: SSN numbers may not be used in queries
  - 6 example queries
    - 4 non-compliant
    - 2 compliant
- Simplified N3 serialization for SPARQL

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?s ?id ?n WHERE {
  ?s foaf:age ?a.
  ?s foaf:openid ?id.
  OPTIONAL { ?s foaf:ssn ?n }.
  FILTER ( ?a > 18 )
}
```

SPARQL query

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix math: <http://www.w3.org/2000/10/swap/math#>.
@prefix s: <http://dig.csail.mit.edu/2009/IARPA-PIR/sparql#> .
@prefix : <http://dig.csail.mit.edu/2009/IARPA-PIR/query1#> .

:Query-5 a s:Select;
   s:cardinality :ALL;
   s:POSList [
      s:variable :S;
      s:variable :N;
      s:variable :ID;
    ];
   s:WhereClause  :WHERE.

   :WHERE a s:DefaultGraphPattern;
       s:TriplePattern  { :S <http://xmlns.com/foaf/0.1/age> :A };
       s:TriplePattern  { :S <http://xmlns.com/foaf/0.1/openid> :ID };
       s:Filter [
          a s:ComparatorExpression;
          s:TriplePattern { :A s:BooleanGT "18"^^xsd:integer }
        ];
       s:OptionalGraphPattern [
             s:TriplePattern  { :S <http://xmlns.com/foaf/0.1/ssn> :N };
        ].

#ends
```

SPARQL query in N3

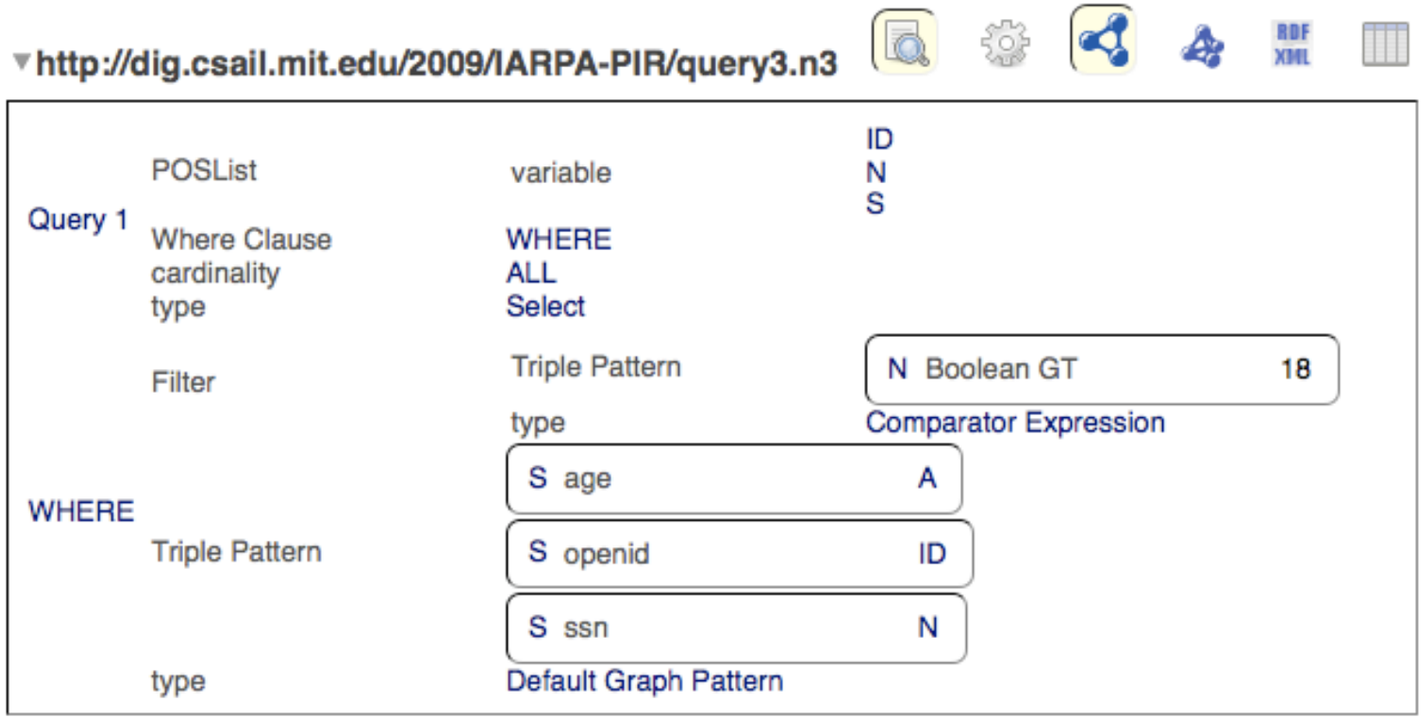
Current research

- Demo compliance/non-compliance for simple queries & policies

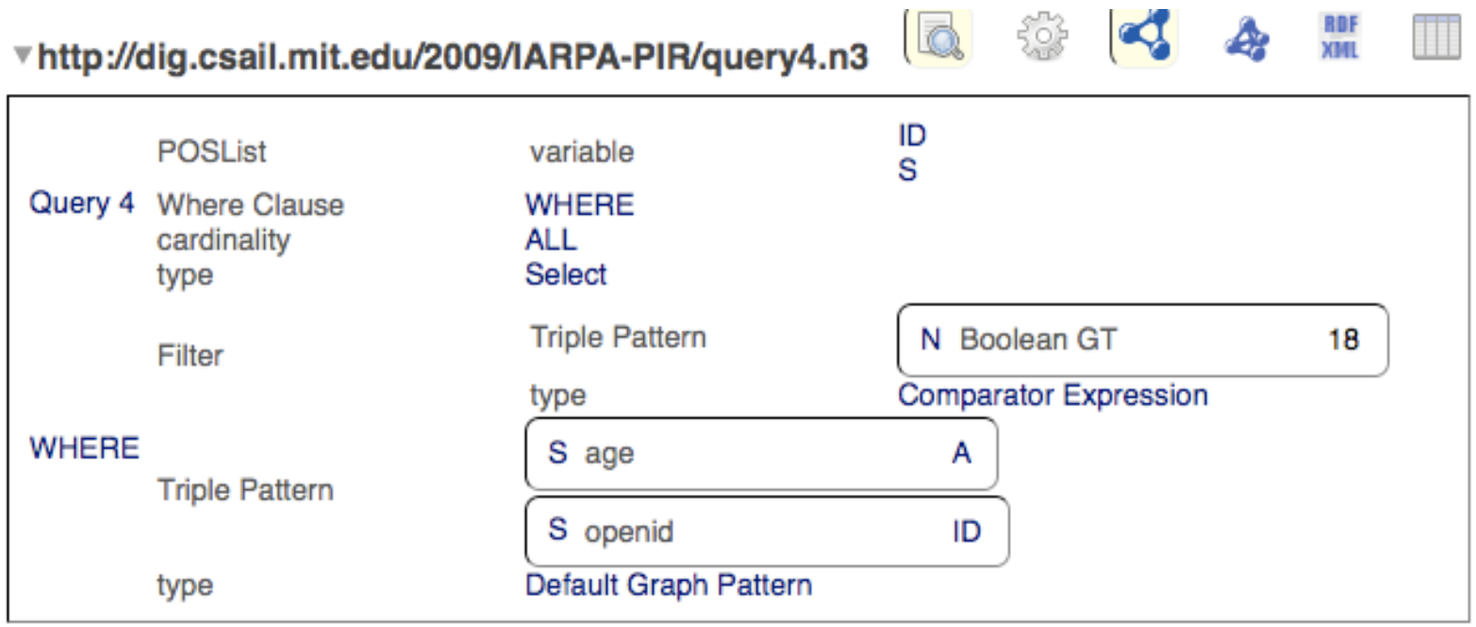
SSNPolicy	label	SSN sample policy for IARPA PIR project
	rule	SSN RULE1
	type	Policy
	description	Q is a SPARQL query
	label	SSN policy rule1
SSN RULE1	pattern	Q POSList P Where Clause W type Select
	rule	SSN RULE2 SSN RULE3 SSN RULE4 SSN RULE5
	type	Belief Rule
	assert	Q non compliant with SSNPolicy
	description	The query, Q , uses SSN values in the where clause, as a filter and retrieves SSN values as well
	label	SSN policy rule2
SSN RULE2	pattern	P variable V W Filter F Triple Pattern T T includes X ssn V F Triple Pattern S S includes V @@: _:n228
	type	Belief Rule
	assert	Q non compliant with SSNPolicy
	description	The query, Q , uses SSN values in the where clause and retrieves SSN values
	label	SSN policy rule3
SSN RULE3	pattern	P variable V W Triple Pattern T

Part of SSN policy

Current research

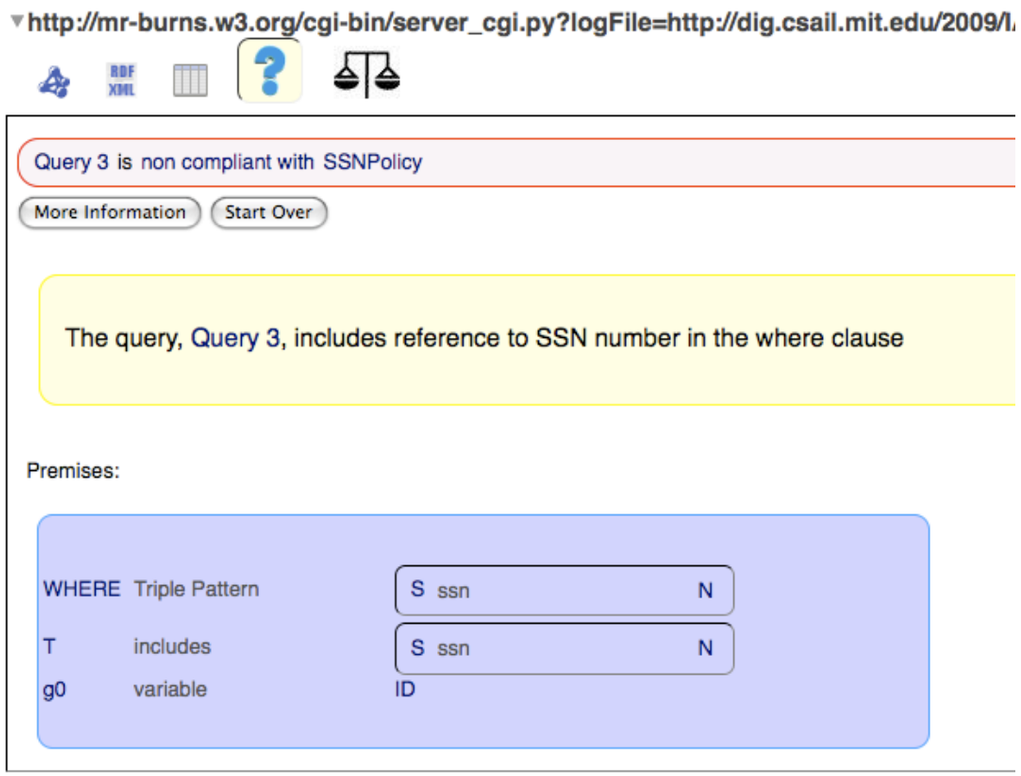


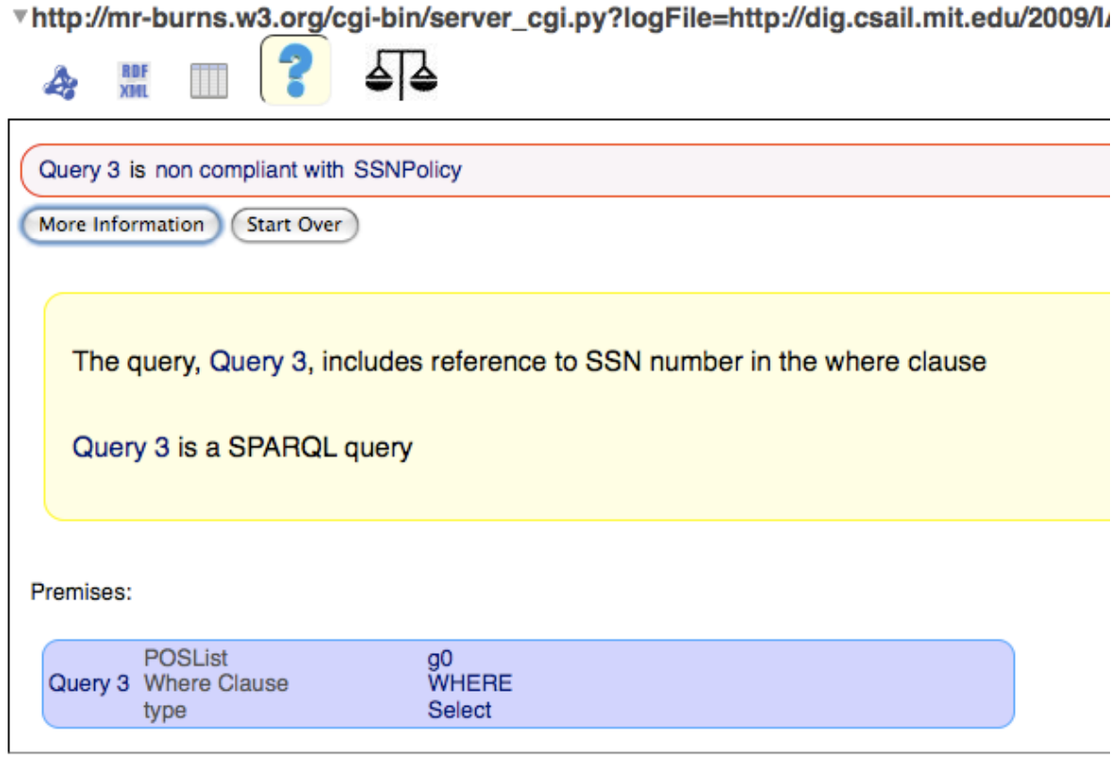
Non-compliant query



Compliant query

Current research





Current Research

- Identified several kinds of policies and queries that we can handle in the first phase
- **I. You cannot retrieve attribute X**
  - These are policies that prevent the user from getting values of certain fields such as SSN or telephone numbers
  - Example policy: You cannot retrieve SSN values
  - Non-compliant query: select * where age=55
  - Compliant query: select name, age, dob where age=55
- **II. You cannot use attribute X**
  - These policies prevent the user from using the attribute X within the query
  - Example policy: You cannot use SSN values
  - Non-compliant query: select name, age, dob where age=15, filter (SSN=123456789)
  - Compliant query: select * where age = 55

Current Research

- III. **You cannot retrieve/use X and Y**
  - These are policies that prevent the user from getting values of more than one field in the same query
  - Example policy: You cannot retrieve SSN and telephone values
  - Non-compliant query: select ssn, telephone where age=55
  - Compliant query: select name, age, dob where age=55
- IV. **You can retrieve/use X or Y.** So if you've already retrieved/used one in the past, you cannot query for the other.
  - These are policies that prevent the user from getting values of one attribute if they've already got the values of another. This policy requires the history of queries
  - Example policy: You can retrieve either SSN or telephone values
  - Non-compliant query: select ssn where age=55 and there is past query select telephone where age=55
  - Compliant query: select ssn where age=55 if there is no past query over telephone number
- V. **If you retrieve X, you must include Y.** Here Y is most likely a condition.
  - Example policy: If you retrieve photos, you must include a condition age > 18
  - Non-compliant query: select photos where ssn=1234567
  - Compliant query: select ssn, photo where age>18

Current Research

- VI. **You can only retrieve (max n of m)**
  - Example policy: You can only retrieve m out of n attributes. This also requires the log of queries.
  - Example query: You can only retrieve 2 out of (name, dob, ssn, county of birth)
  - Non-compliant query: select name, dob where age = 55 and past query was select add, ssn where age = 55
- VII. **You can only retrieve a certain percent of the database.**
  - This requires meta data about the attribute/col values. E.g. ssn divides the database into no. of rows
  - Example policy: You can only retrieve 5% of the db
  - Non-compliant query: select * where gender=f
  - Compliant query: select * where ssn=1234567 (where ssn distribution is n)

Next Steps for Phase I

- N3 semantics for SPARQL
- Complete automated conversion of SPARQL into N3
- Policy editor/UI
- Include support for SQL queries either converting SQL to RDF directly or via SPARQL
- Convert sets of queries and policies prepared by the test and evaluation team into SPARQL/SQL queries and AIR policies

Next steps

- Extend Justification UI to provide more relevant explanations
- Develop methodology to generate more specific AIR rules from abstract policies and often ambiguous policies
- Policy development toolkit

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix math: <http://www.w3.org/2000/10/swap/math#>.
@prefix s: <http://dig.csail.mit.edu/2009/IARPA-PIR/sparql#> .
@prefix : <http://dig.csail.mit.edu/2009/IARPA-PIR/query1#> .

:Query-5 a s:Select;
   s:cardinality :ALL;
   s:POSList [
      s:variable :S;
      s:variable :N;
      s:variable :ID;
    ];
   s:WhereClause  :WHERE.

   :WHERE a s:DefaultGraphPattern;
      s:TriplePattern { :S <http://xmlns.com/foaf/0.1/age> :A };
      s:TriplePattern  { :S <http://xmlns.com/foaf/0.1/openid> :ID };
      s:Filter [
         a s:ComparatorExpression;
         s:TriplePattern { :A s:BooleanGT "18"^^xsd:integer }
       ];
      s:OptionalGraphPattern [
            s:TriplePattern  { :S <http://xmlns.com/foaf/0.1/ssn> :N };
       ].

#ends
```

SPARQL query in N3

```
:SSN_RULE1 a air:BeliefRule;
      air:label "SSN policy rule1";
      air:pattern {
       :Q a s:Select;
         s:POSList :P;
          s:WhereClause :W.
      };
       air:description (:Q " is a SPARQL query");
       air:rule :SSN_RULE2, :SSN_RULE3, :SSN_RULE4, :SSN_RULE5.


  :SSN_RULE3 a air:BeliefRule;
       air:label "SSN policy rule3";
       air:pattern {
            :P s:variable :V.
            :W s:TriplePattern :T.
            :T log:includes { :X <http://xmlns.com/foaf/0.1/ssn> :V }
        };
        air:description ("The query, " :Q  ", uses SSN values in the
 where clause and retrieves SSN values");
       air:assert { :Q air:non-compliant-with :SSNPolicy }.
```

Part of an AIR policy

References

- Policy Assurance for PIR Queries, http://dig.csail.mit.edu/2009/IARPA-PIR/
- TAMI project, http://dig.csail.mit.edu/TAMI
- Tabulator extension, http://dig.csail.mit.edu/2007/tab/
- AIR specifications, http://dig.csail.mit.edu/TAMI/2008/12/AIR
- Paper on AIR, http://dig.csail.mit.edu/2008/Papers/IEEEPolicy
- SPASQL, http://www.w3.org/2006/Talks/0518-SPASQL/