Privacy-Aware Data-Intensive Applications

Michele Guerriero
Politecnico di Milano
michele.guerriero@polimi.it
Today's Information and Communication Technologies

- Advancements in ICTs enable the development of powerful and more efficient infrastructures and services:
  - collection of big data from different sources
  - increase demand for Data-Intensive Applications (DIAs)
The Evolution of Modern Data Processing

- From Map-Reduce to Directed Acyclic Graph-based execution
- The rise of distributed stream processors for large-scale and real-time data processing
- The Lambda architecture for balancing batch and streaming computations
- The Google Dataflow Model: a unified programming model for both batch and streaming data pipelines
Are We Missing Something?

- The problem with Big Data is not just how do we process them.
- In many cases Big Data are personal and often sensitive.
- Privacy becomes more and more a primary concern in modern DIAs.
Towards Privacy-Aware DIAs

- Data subjects should be able to specify requirements on how their data are used
- DIA designers and developers should be able to easily enforce such requirements

Solution:

1. a language to let data subjects to specify privacy policies on modern DIAs
2. an automatic mechanism to enforce such policies
A Privacy Model for Modern Data-Intensive Applications
**View Generalization Policies for Data Subject-Specific Streams**

- **View generalization policies (VGP):** allow data subject to define views over data **subject-specific streams**

- A VGP attached on a data subject-specific stream defines how tuples referring to a given data subject should be published when a given context holds

- Use Domain Generalization Hierarchy (DGH) to define views

![Diagram of Domain Generalization Hierarchy]

- $D_2 = \{\ast\}$
- $D_1 = \{1, \ldots, 100\}$
- $D_0 = \mathbb{N}$
VGP Desired Effect

VGP by data subject $ds_1$:

if context($ctx$) 
then generalise($S_2$, 1)

Time window of 6 time units

$O_2$: for each time window sum the content per data subject

Specifies to which level of the associated DGH the content of $S_2$ must be generalised
Data Subject Eviction Policies

- **Data subject eviction policies (DSEP):** allow data owners to avoid their data to be considered by a given computation.

- A DSEP attached on a *data subject-generic stream* $S$ defines in which context tuples referring to a given data subject should be evicted from the input streams of the operator that produces $S$. 
### DSEP Desired Effect

**DSEP by data subject $ds_1$:**

```
if context($ctx$) then evict($S_3$)
```

#### Data Table

<table>
<thead>
<tr>
<th>data subject</th>
<th>timestamp</th>
<th>content</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ds_1$</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>$ds_2$</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>44</td>
<td>44</td>
</tr>
<tr>
<td>$ds_2$</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>73</td>
<td>73</td>
</tr>
<tr>
<td>$ds_2$</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>$ds_2$</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

#### Time Window

The time window of 6 time units is highlighted.

#### Context

Context ($ctx$) is represented by the highlighted area.

#### Execution

- **$S_1$:**
  - Data subject $ds_1$
  - Timestamp: 8, 12
  - Content: 8, 12

- **$O_2$:** For each time window counts the tuples with value greater than 50.
  - Time window: 6 time units
  - Counts: 6, 1

- **$S_3$:**
  - Data subject $ds_3$
  - Timestamp: 6
  - Content: 6
Defining the Context

● Context modeled as a set of contextual variables:

1. **dynamic variables** change during a user session (e.g. the various real-time data computed by a given DIA, the user location, etc.)
   - past values might be of interest

2. **static variables** does not change during a user session (e.g. the user identity, her purpose, etc.)
   - only their current value is of interest

● **Policy enabling context**: Metric Temporal Logic formula specifying conditions over the past value of dynamic variables as well as the current value of static variables
Automatic Policy Enforcement via Dataflow Rewriting

- Define a set of privacy enhancing dataflow operator
- **PastConditionChecker (PCC)**: checks the validity of past conditions over dynamic variables
- **ViewBuilder (VB)**: enforces the VGPs specified on a given data subject specific stream
- **DataSubjectEvictor (DSE)**: enforces the DSEPs on a given data subject generic stream
Enforcing View Creation Policies

VGP by data subject $ds_1$:

\[
\text{if } \text{Past}_{[T1,T2]}(S_1>30) \& S_2<10 \\
\text{then generalise}(S_3, 1)
\]
Enforcing Data Subject Eviction Policies

DSEP by data subject $d_{S_1}$:
\[
\text{if } \text{Past}_{[T_1,T_2]}(S_1 > 30) \& S_1 < 10 \\
\text{then } \text{evict}(S_3)
\]
Evaluation Plan

● Performance evaluation focused on:

1. understanding the introduced performance overhead

2. understanding the main model variables that affect performance and how

● Apply trace-checking to verify the correctness of the policy enforcement implementation

● Apply the proposed approach on real-world use cases (how? How to find them?)

● How to compare when there are really no similar approaches out there?
Preliminary Results

- Prototype implementation on top of the Apache Flink dataflow processor
- Preliminary performance evaluation on a cluster of 30 cores:

<table>
<thead>
<tr>
<th>Example Application 1</th>
<th>Latency</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Policy</td>
<td>1.5 ms</td>
<td>61.11 t/ms</td>
</tr>
<tr>
<td>1 VGP</td>
<td>2.8 ms</td>
<td>56.24 t/ms</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example Application 2</th>
<th>Latency</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Policy</td>
<td>1.9 ms</td>
<td>60.74 t/ms</td>
</tr>
<tr>
<td>1 DSEP</td>
<td>5.3 ms</td>
<td>57.14 t/ms</td>
</tr>
</tbody>
</table>
Future Work and Thesis Plan

- Rigorously follow the evaluation plan
- Dataflow computing and programming fits very well with model-based approaches:
  1. Apply model-driven approach to further simplify the development of privacy-aware DIAs
  2. Extend results from previous research on model-driven engineering for DIAs
Conclusion

● Novel scenarios require new solutions to protect data

● Need to provide data owners with control over their data

● Design and development of privacy-aware applications needs to be made easy

● Data protection solutions are beneficial to both:
  1. data owners (empowered with control)
  2. data controllers (increased confidence of users, decreased liability)
Thank You!