Balancing smartness and privacy for the Ambient Intelligence

Harold van Heerde

Electrical Engineering, Mathematics and Computer Science
University of Twente
... is everywhere and has influence on many aspects of daily life

... is invisible

... renders fast technical developments in new, powerful, sensing capabilities

... has a memory
Why does the AmI have this ‘memory’?
Storing of context data to facilitate ‘smartness’

- Context histories make it possible to:
  - Infer, predict and learn
    - pattern recognition
    - inferring next location
    - predicting next events
    - ...et cetera
  - Share knowledge
    - profile matching
    - finding experts
    - interests sharing
    - ...et cetera

Smartness...
...depends on quality and quantity of context data
Why does the AmI have this ‘memory’?
Storing of context data to facilitate ‘smartness’

- Context histories make it possible to:
  - Infer, predict and learn
    - pattern recognition
    - inferring next location
    - predicting next events
    - ...et cetera
  - Share knowledge
    - profile matching
    - finding experts
    - interests sharing
    - ...et cetera

Smartness...
...depends on quality and quantity of context data
This introduces a privacy problem!

- People will not always be aware of being monitored
- People do not know what happens or will happen with their data
- It will sometimes be hard to detect privacy violations
- Privacy sensitive ‘facts’ from the past will be kept forever in the system
  - Facts from the past can be used against you

Problem

We want to find a balance between smartness and privacy
Introduction

Assumptions and research directions
First approach: User oriented degradation
Second approach: application oriented degradation

This introduces a privacy problem!

- People will not always be aware of being monitored
- People do not know what happens or will happen with their data
- It will sometimes be hard to detect privacy violations
- Privacy sensitive ‘facts’ from the past will be kept forever in the system
  - Facts from the past can be used against you

Problem

We want to find a balance between smartness and privacy
Why privacy protection?

- Data should not be retained excessively
  - Imposed by the law (U.N. regulations for example)
    - Reduces impact of hacking
    - Avoid tracing
  - Trusted organizations are pushed to respect the law

Related work

- $k$-Anonymization, $l$-diversity, et cetera
- Hippocratic databases, p3p policies
- Access control (encryption, micro-views, et cetera)
- ...
Why privacy protection?

- Data should not be retained excessively
  - Imposed by the law (U.N. regulations for example)
    - Reduces impact of hacking
    - Avoid tracing
  - Trusted organizations are pushed to respect the law

Related work

- $k$-Anonymization, l-diversity, et cetera
- Hippocratic databases, p3p policies
- Access control (encryption, micro-views, et cetera)
- …
Table of contents

1. Assumptions and research directions
   - Assumptions
   - Research direction
   - Retention model

2. First approach: User oriented degradation
   - Value degradation with Life-Cycle policies
   - Model implementation
   - Functional degradation

3. Second approach: application oriented degradation
   - Degradation in isolation for a known query set
   - Examples
   - Problems
Introduction
Assumptions and research directions
First approach: User oriented degradation
Second approach: application oriented degradation

Assumptions
Research direction
Retention model

Architecture

Figure: Architecture

End users
Applications
Services
Context
(LCP)
(Trusted) DBMS

Harold van Heerde
Balancing smartness and privacy
Assumptions

- **DBMS - Honest**
  - DB and DB admin can be trusted *now* and in the near future, but might become untrusted in the future.

- **Applications - Trusted**
  - Applications have an interest to respect privacy (in order to keep their market segment).
  - Application code and data exchange communication cannot be attacked.

- **End users - Malicious**
  - Only results or services given by applications are visible for end users.
  - Attacks because of (physical) spying between users are not addressed.
Assumptions

- **DBMS - Honest**
  - DB and DB admin can be trusted *now* and in the near future, but might become untrusted in the future.

- **Applications - Trusted**
  - Applications have an interest to respect privacy (in order to keep their market segment).
  - Application code and data exchange communication cannot be attacked.

- **End users - Malicious**
  - Only results or services given by applications are visible for end users.
  - Attacks because of (physical) spying between users are not addressed.
Assumptions

- **DBMS - Honest**
  - DB and DB admin can be trusted now and in the near future, but might become untrusted in the future.

- **Applications - Trusted**
  - Applications have an interest to respect privacy (in order to keep their market segment).
  - Application code and data exchange communication cannot be attacked.

- **End users - Malicious**
  - Only results or services given by applications are visible for end users.
  - Attacks because of (physical) spying between users are not addressed.
Access control or limited retention?

**Access control**
- Policies define until when and to which data applications have access.
- Data is kept and protected within the system.
- The DB (and administrator) should be trusted now and in the future.

**Limited retention**
- Use policies to define until when and which data is kept in the system.
- Physical removal of data.
- DB can be trusted now, but might be not in the future.
Access control or limited retention?

### Access control
- Policies define until when and to which data applications have access.
- Data is kept and protected within the system.
- The DB (and administrator) should be trusted now and in the future.

### Limited retention
- Use policies to define until when and which data is kept in the system.
- Physical removal of data.
- DB can be trusted *now*, but might be not in the future.
Progressive degradation (as a retention model)

- Fill the gap between ‘all or nothing’
  - Destroy all data means no smartness for applications
  - Keeping all data means possible privacy violation for users

Example

- A supermarket uses accurate data to predict if new cash desks should be opened
- It uses per customer buying information to make personalized advertisements
- And uses general statistics to optimize its selection of goods.
The Life-Cycle Policy model
Progressive value degradation of context histories

- Users specify when and how data should be degraded
- Goal is to protect context histories:
  - Degraded data is less privacy sensitive
- Current and recent accurate data is still available
  - Degraded data is still useful in terms of smartness
- There is room for negotiation about policies between application and users
The LCP model

- Data modeled as context triplets
- Triplets are elements in different states of accuracy
- Life-Cycle policies are transitions between states
- Users can specify their own LCP

(a) The Cube

(b) Example LCP

Harold van Heerde
Balancing smartness and privacy
Functional degradation

- Data processing can be translated to SQL statements
- ‘Disable’ SQL operators by transforming the data
- Progressive degradation of the accuracy of those *abilities*

**Example**

- Keep the join ability without keeping real time values
- Degrade the ability from ‘join on minutes’ to ‘join on hour’
Functional degradation

- Data processing can be translated to SQL statements
- ‘Disable’ SQL operators by transforming the data
- Progressive degradation of the accuracy of those abilities

Example

- Keep the join ability without keeping real time values
- Degrade the ability from ‘join on minutes’ to ‘join on hour’
Value degradation and functional degradation

Figure: (Natural) value degradation, and functional degradation of abilities
Degradation in isolation for a known query set

Application oriented approach

Adequacy

Given a query $Q$ on a dataset $D$, there is a degradation function $V$ such that an alternative query $Q'$ on $V(D)$ can be found which produces the same result as $Q$.

- Provide an adequate degradation function for a known set of queries
- Goal is to keep the least amount of information necessary to provide adequacy
Example: join on time

Query: $\pi_{\text{door}.\text{event},\text{window}.\text{event}}(\text{Door} \bowtie_{\text{hour}(\text{time})} \text{Window})$

<table>
<thead>
<tr>
<th>event</th>
<th>time</th>
<th>loc</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>e1</td>
<td>15.10</td>
<td>3090</td>
<td>1</td>
</tr>
<tr>
<td>e2</td>
<td>15.10</td>
<td>3090</td>
<td>2</td>
</tr>
<tr>
<td>e3</td>
<td>15.15</td>
<td>2045</td>
<td>2</td>
</tr>
<tr>
<td>e4</td>
<td>16.30</td>
<td>4180</td>
<td>1</td>
</tr>
<tr>
<td>e5</td>
<td>17.00</td>
<td>3090</td>
<td>3</td>
</tr>
<tr>
<td>e6</td>
<td>17.15</td>
<td>5360</td>
<td>3</td>
</tr>
</tbody>
</table>

(a) Door

<table>
<thead>
<tr>
<th>event</th>
<th>time</th>
<th>loc</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>w1</td>
<td>15.10</td>
<td>3090</td>
<td>2</td>
</tr>
<tr>
<td>w2</td>
<td>16.05</td>
<td>1027</td>
<td>1</td>
</tr>
<tr>
<td>w3</td>
<td>17.02</td>
<td>2045</td>
<td>3</td>
</tr>
<tr>
<td>w4</td>
<td>18.05</td>
<td>2045</td>
<td>3</td>
</tr>
<tr>
<td>w5</td>
<td>18.15</td>
<td>2045</td>
<td>3</td>
</tr>
<tr>
<td>w6</td>
<td>18.16</td>
<td>2045</td>
<td>1</td>
</tr>
</tbody>
</table>

(b) Window

Result: $\{(e_1, w_1), (e_2, w_1), (e_3, w_1), (e_4, w_2), (e_5, w_3), (e_6, w_3)\}$
Example: join on time (cont’d)

Query: $\pi_{V(\text{door}).\text{event'}, V(\text{window}).\text{event'}} (V(\text{Door}) \Join_{\text{time'}} V(\text{Window}))$

<table>
<thead>
<tr>
<th>event’</th>
<th>time’</th>
<th>loc’</th>
<th>id’</th>
<th>event’</th>
<th>time’</th>
<th>loc’</th>
<th>id’</th>
</tr>
</thead>
<tbody>
<tr>
<td>e1</td>
<td>a</td>
<td>3090</td>
<td>1</td>
<td>w1</td>
<td>a</td>
<td>3090</td>
<td>2</td>
</tr>
<tr>
<td>e2</td>
<td>a</td>
<td>3090</td>
<td>2</td>
<td>w2</td>
<td>b</td>
<td>1027</td>
<td>1</td>
</tr>
<tr>
<td>e3</td>
<td>a</td>
<td>2045</td>
<td>2</td>
<td>w3</td>
<td>c</td>
<td>2045</td>
<td>3</td>
</tr>
<tr>
<td>e4</td>
<td>b</td>
<td>4180</td>
<td>1</td>
<td>w4</td>
<td>d</td>
<td>2045</td>
<td>3</td>
</tr>
<tr>
<td>e5</td>
<td>c</td>
<td>3090</td>
<td>3</td>
<td>w5</td>
<td>d</td>
<td>2045</td>
<td>3</td>
</tr>
<tr>
<td>e6</td>
<td>c</td>
<td>5360</td>
<td>3</td>
<td>w6</td>
<td>d</td>
<td>2045</td>
<td>1</td>
</tr>
</tbody>
</table>

(c) $V(\text{Door})$
(d) $V(\text{Window})$

Result: $\{(e_1, w_1), (e_2, w_1), (e_3, w_1), (e_4, w_2), (e_5, w_3), (e_6, w_3)\}$
Example: join on location

Query: \( \pi_{door\cdot event, window\cdot event} (\text{Door } \bowtie_{\text{loc}} \text{ Window}) \)

<table>
<thead>
<tr>
<th>event</th>
<th>time</th>
<th>loc</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>e1</td>
<td>15.10</td>
<td>3090</td>
<td>1</td>
</tr>
<tr>
<td>e2</td>
<td>15.10</td>
<td>3090</td>
<td>2</td>
</tr>
<tr>
<td>e3</td>
<td>15.15</td>
<td>2045</td>
<td>2</td>
</tr>
<tr>
<td>e4</td>
<td>16.30</td>
<td>4180</td>
<td>1</td>
</tr>
<tr>
<td>e5</td>
<td>17.00</td>
<td>3090</td>
<td>3</td>
</tr>
<tr>
<td>e6</td>
<td>17.15</td>
<td>5360</td>
<td>3</td>
</tr>
<tr>
<td>(e) Door</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>event</th>
<th>time</th>
<th>loc</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>w1</td>
<td>15.10</td>
<td>3090</td>
<td>2</td>
</tr>
<tr>
<td>w2</td>
<td>16.05</td>
<td>1027</td>
<td>1</td>
</tr>
<tr>
<td>w3</td>
<td>17.02</td>
<td>2045</td>
<td>3</td>
</tr>
<tr>
<td>w4</td>
<td>18.05</td>
<td>2045</td>
<td>3</td>
</tr>
<tr>
<td>w5</td>
<td>18.15</td>
<td>2045</td>
<td>3</td>
</tr>
<tr>
<td>w6</td>
<td>18.16</td>
<td>2045</td>
<td>1</td>
</tr>
<tr>
<td>(f) Window</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Result: \( \{(e_1, w_1), (e_2, w_1), (e_3, w_3), (e_3, w_4), (e_3, w_5), (e_3, w_6)\} \)
Example: join on location (cont’d)

Query: $\pi V(\text{door}).\text{event'}, V(\text{window}).\text{event'} (V(\text{Door}) \bowtie_{\text{loc'}} V(\text{Window}))$

<table>
<thead>
<tr>
<th>event</th>
<th>time</th>
<th>loc</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>e1</td>
<td>15.10</td>
<td>a</td>
<td>1</td>
</tr>
<tr>
<td>e2</td>
<td>15.10</td>
<td>a</td>
<td>2</td>
</tr>
<tr>
<td>e3</td>
<td>15.15</td>
<td>b</td>
<td>2</td>
</tr>
<tr>
<td>e4</td>
<td>16.30</td>
<td>d</td>
<td>1</td>
</tr>
<tr>
<td>e5</td>
<td>17.00</td>
<td>a</td>
<td>3</td>
</tr>
<tr>
<td>e6</td>
<td>17.15</td>
<td>e</td>
<td>3</td>
</tr>
</tbody>
</table>

(g) Door

<table>
<thead>
<tr>
<th>event</th>
<th>time</th>
<th>loc</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>w1</td>
<td>15.10</td>
<td>a</td>
<td>2</td>
</tr>
<tr>
<td>w2</td>
<td>16.05</td>
<td>c</td>
<td>1</td>
</tr>
<tr>
<td>w3</td>
<td>17.02</td>
<td>b</td>
<td>3</td>
</tr>
<tr>
<td>w4</td>
<td>18.05</td>
<td>b</td>
<td>3</td>
</tr>
<tr>
<td>w5</td>
<td>18.15</td>
<td>b</td>
<td>3</td>
</tr>
<tr>
<td>w6</td>
<td>18.16</td>
<td>b</td>
<td>1</td>
</tr>
</tbody>
</table>

(h) Window

3090 $\rightarrow a, 2045 \rightarrow b, 1024 \rightarrow c, 4180 \rightarrow d, 5360 \rightarrow e$

Result: $\{(e_1, w_1), (e_2, w_1), (e_3, w_3), (e_3, w_4), (e_3, w_5), (e_3, w_6)\}$
Additional information is needed to keep the degraded data adequate

- Possibly use secure hardware, access control, distributed keys, *et cetera*
- At least the data itself can be stored in a non-secure database

Possible approach: one-way hash function

- But, domain is finit and sometimes small: easy to brake irreversability
- Use an additional key, and keep the key secret
  - Still requires access control, and disclosure of the key means disclosure of all data
Problems

- Additional information is needed to keep the degraded data 
  adequate
  - Possibly use secure hardware, access control, distributed keys, *et cetera*
  - At least the data itself can be stored in a non-secure database

Possible approach: one-way hash function

- But, domain is finit and sometimes small: easy to brake irreversability
- Use an additional key, and keep the key secret
  - Still requires access control, and disclosure of the key means disclosure of all data
Conclusion

Ultimate goal is to balance privacy and smartness:

- Giving control to users limits asymmetric information
- Physical removal of data prevents unauthorized data disclosure
- Progressive degradation balances users wishes and application requirements
Questions?