The Timed Decentralised Label Model

Martin Leth Pedersen, Michael Hedegaard Sørensen
{mhso10,mped10}@student.aau.dk

Daniel Lux
daniel@seluxit.com

Ulrik Nyman, René Rydhof Hansen
{ulrik,rrh}@cs.aau.dk

Aalborg University
Seluxit

NordSec 2015, October 19, 2015

The Timed Decentralised Label Model

A combination of Time and Security.

- The Decentralised Label Model
- A concept of Time: Timed Automata

The Timed Decentralised Label Model

A combination of Time and Security.

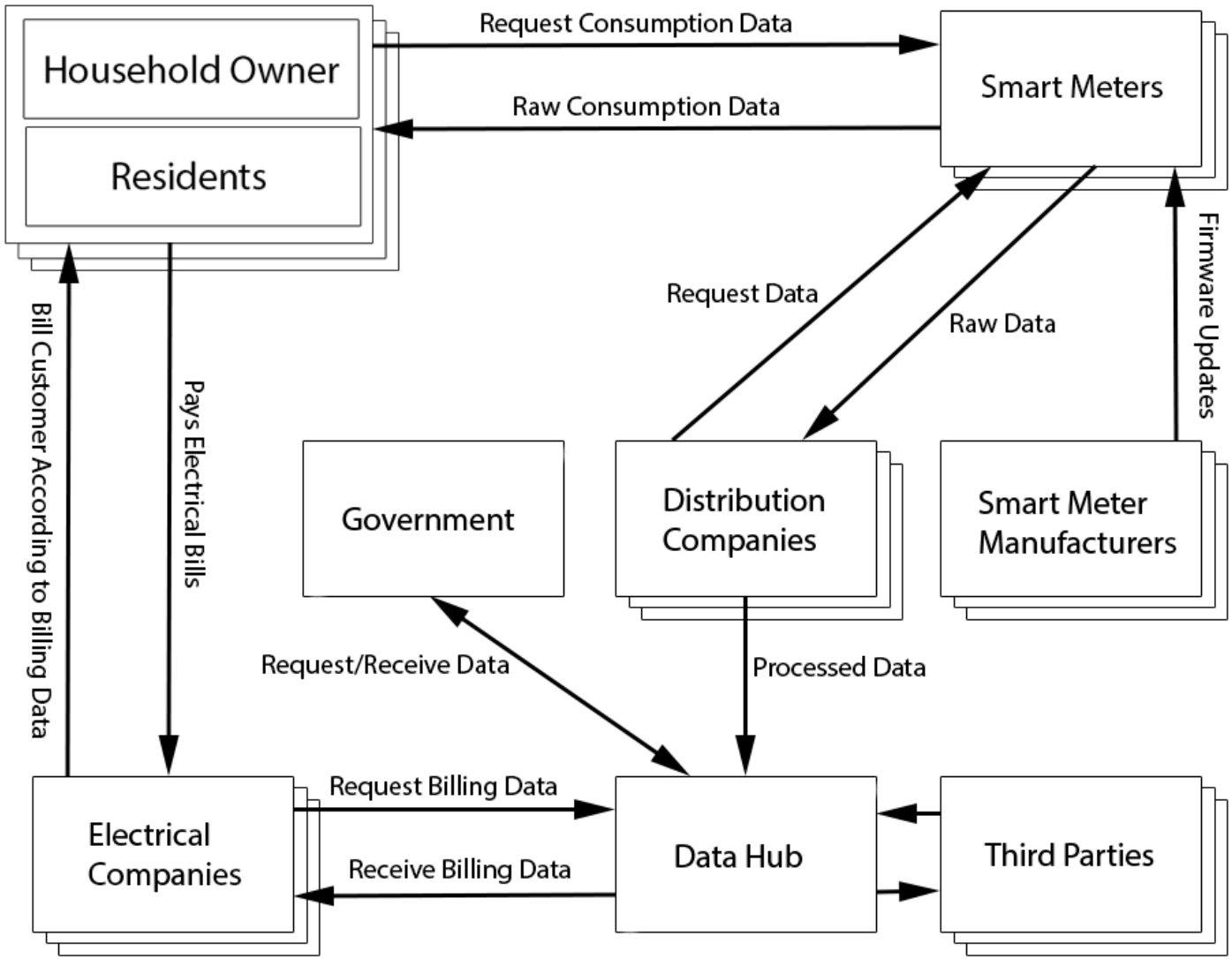
- The Decentralised Label Model
- A concept of Time: Timed Automata
- Why?

The Timed Decentralised Label Model

A combination of Time and Security.

- The Decentralised Label Model
- A concept of Time: Timed Automata
- Why?
- Specification of timed behavior of security policies
  - Easier
  - Understandable

Smart meter system with communication channels outlined



The Decentralised Label Model

- Defines legal flow of data
- Principals can specify read/write rights

- Act-for hierarchy

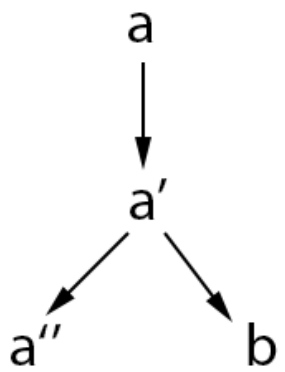


Figure : Example principal hierarchy: $a \succeq a'$, $a' \succeq a''$, and $a' \succeq b$.

The Decentralised Label Model

- Defines legal flow of data
- Principals can specify read/write rights
  - Only examples with read (confidentiality) (No integrity)
- Act-for hierarchy

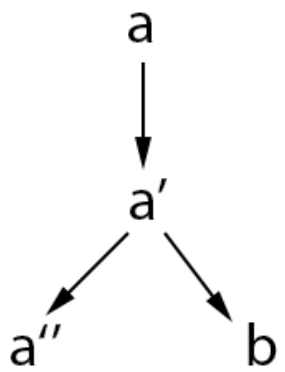


Figure : Example principal hierarchy: $a \succeq a'$, $a' \succeq a''$, and $a' \succeq b$.

A security model

$$\text{Label} = \mathcal{P}(\text{SecPol}) \qquad \text{SecPol} = \text{Owner} \times \mathcal{P}(\text{Reader})$$

A security model

$$\text{Label} = \mathcal{P}(\text{SecPol}) \qquad \text{SecPol} = \text{Owner} \times \mathcal{P}(\text{Reader})$$

- only readers that are permitted by all owners (of a particular label) are permitted to access data with that label.

A security model

$$\text{Label} = \mathcal{P}(\text{SecPol}) \qquad \text{SecPol} = \text{Owner} \times \mathcal{P}(\text{Reader})$$

- only readers that are permitted by all owners (of a particular label) are permitted to access data with that label.

$$\mathit{effectiveReaders}(L) = \bigcap_{o \in \mathit{owners}(L)} \mathit{readers}_{\succeq}(\mathit{readers}(L, o))$$

Relabeling of data

- Restriction: Assigning a higher level of security to some specific data.
- Declassification: Intended "leak" of data.

Timed Automata Example

A smart meter

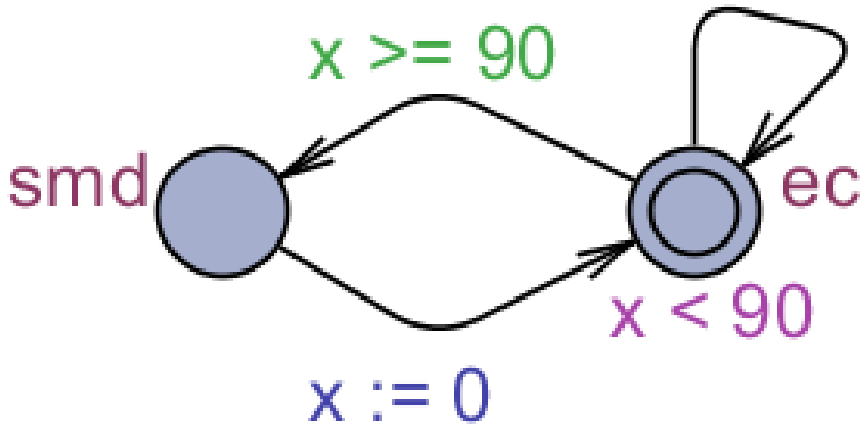


Figure : Simple smart meter example where an electrical company reads smart meter data every 90 days. The location with the double circle is a start location.

Timed Automata Example

A smart meter

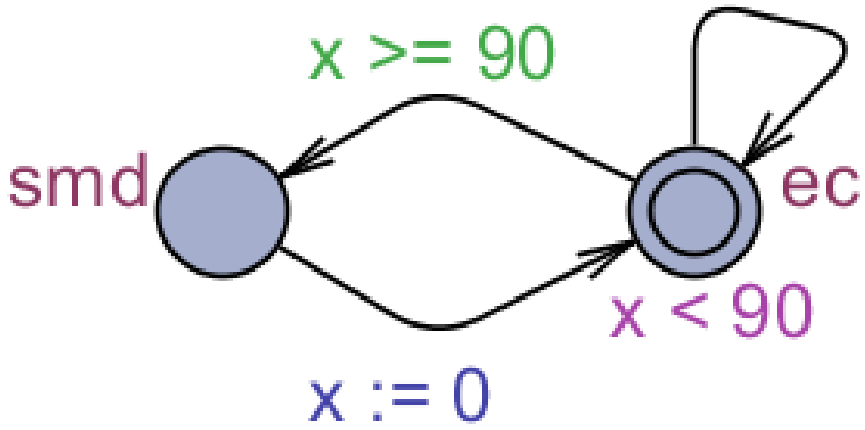


Figure : Simple smart meter example where an electrical company reads smart meter data every 90 days. The location with the double circle is a start location.

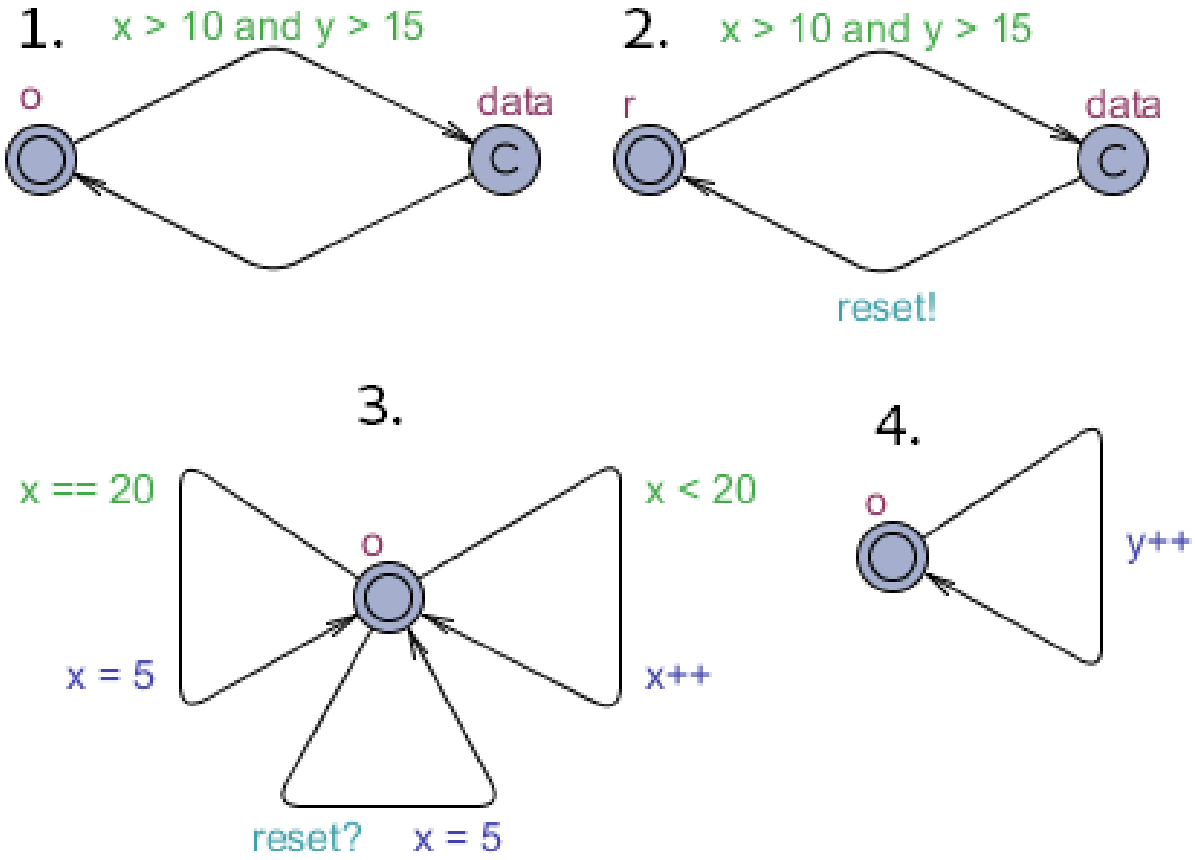
- Verification tool: UPPAAL (www.uppaal.org)

TDLM example

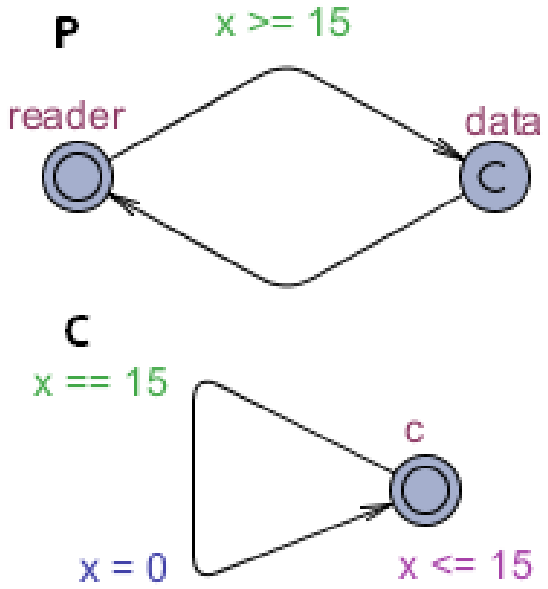
$$\{o\ (x[20; ?reset; 5] > 10\ \&\&\ y > 15) : r[!reset]\}$$

TDLM example

$$\{o\ (x[20; ?reset; 5] > 10\ \&\&\ y > 15) : r[!reset]\}$$



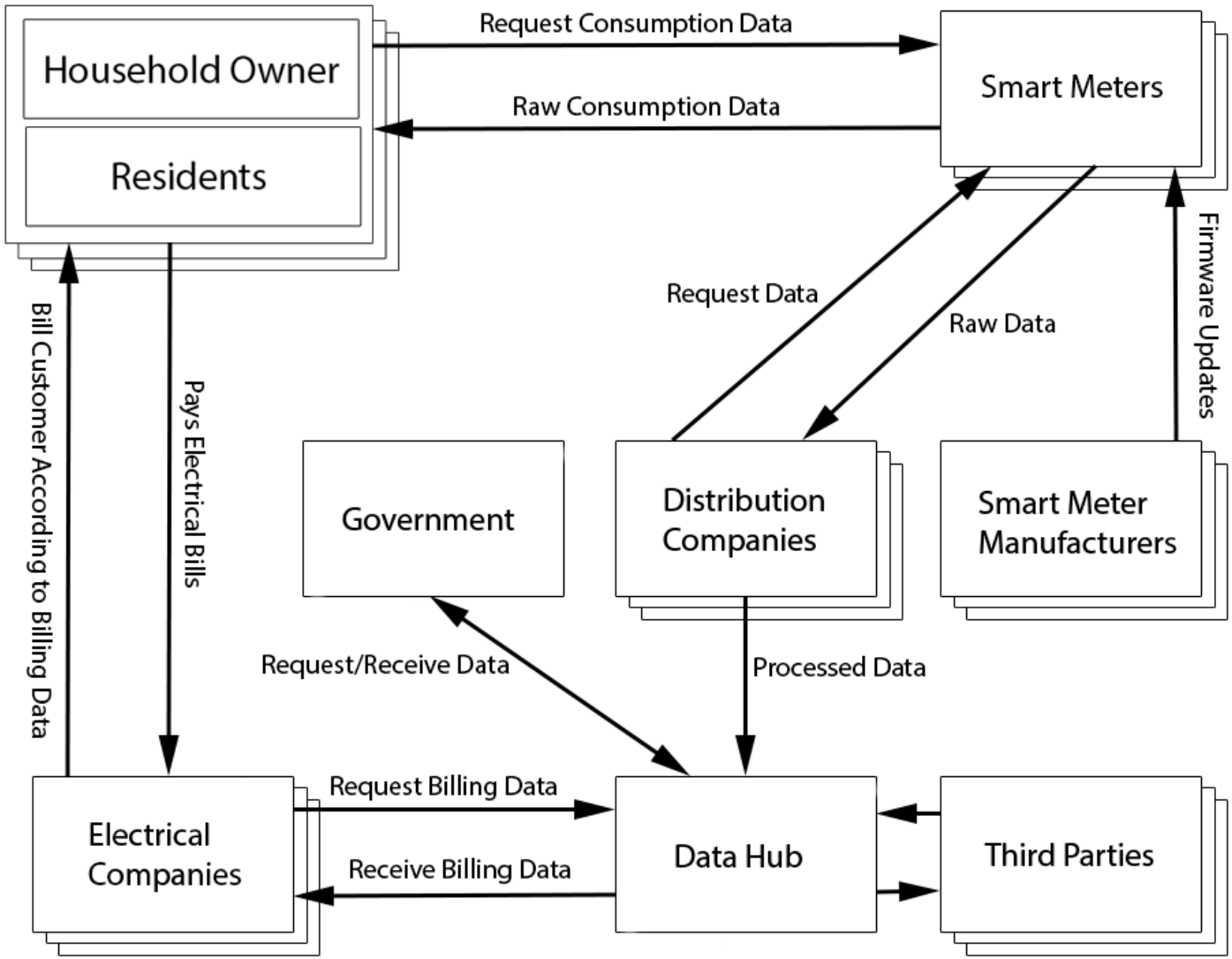
Verification of Security Policies



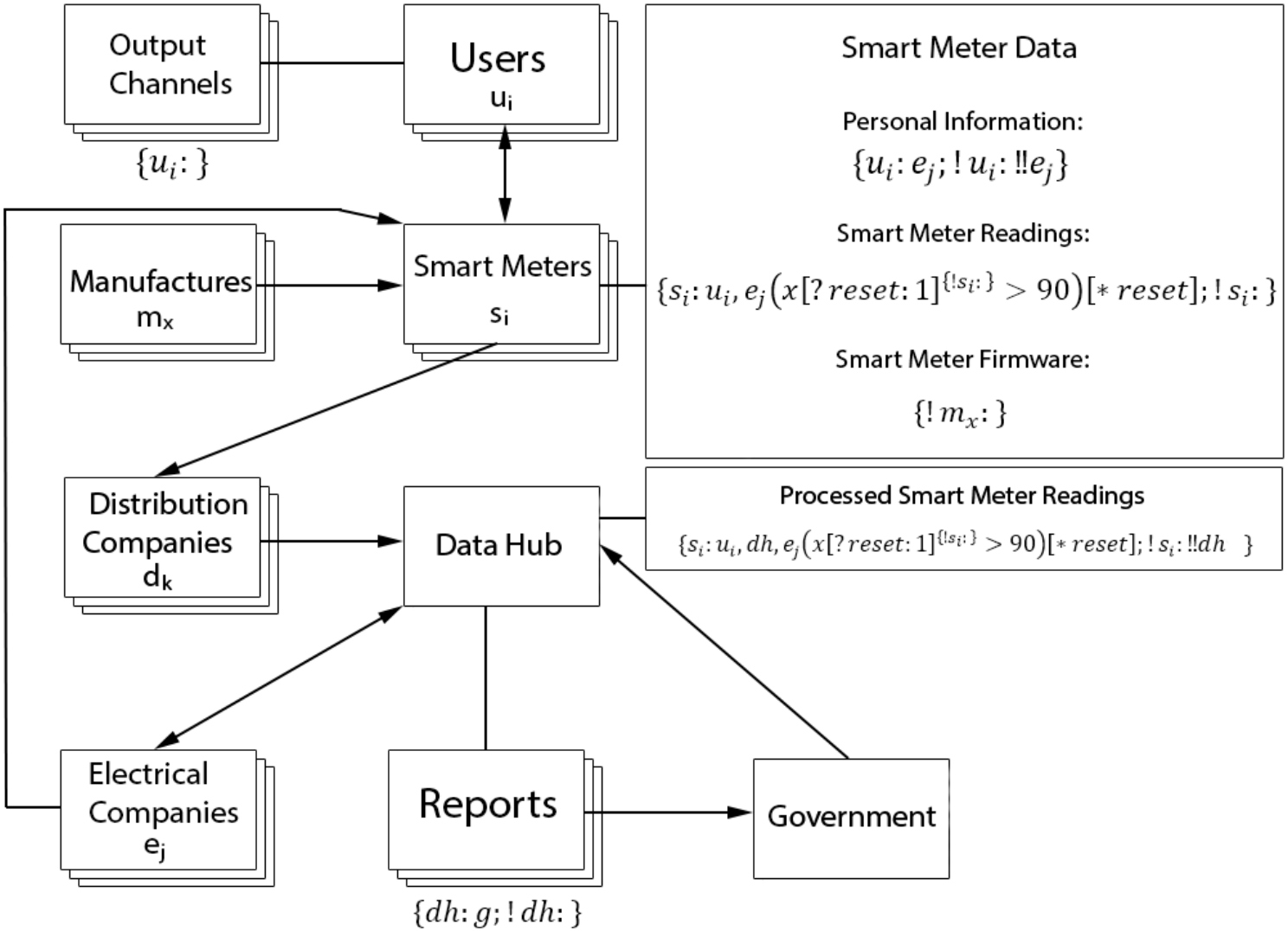

```
E<> P.data and x == 14
A[] P.data imply x >= 15
E<> P.data and x == 16
E<> P.data and x == 15
```

Figure : UPPAAL window of opportunity analysis where upper limit is set to 15.

Smart meter system with communication channels outlined



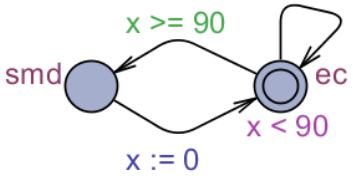
Smart meter system with TDLM security labels

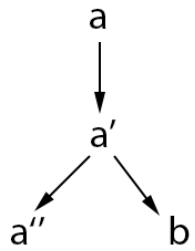


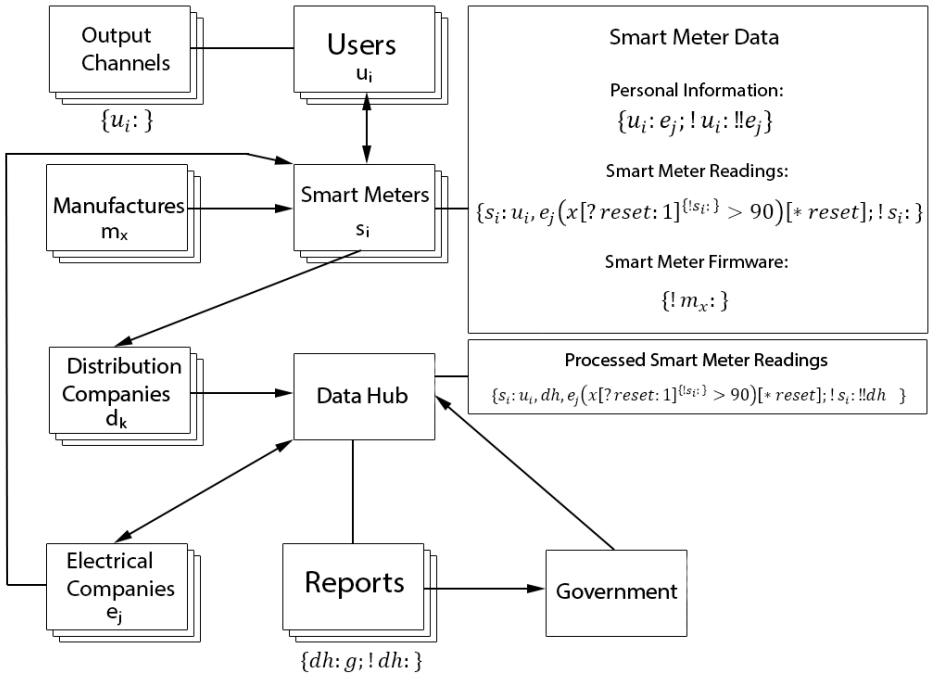
Conclusion

- Our model is decentralised
- Time is hard!
- This is initial work.
- Verification is a nice side effect (for now).
- What should we do next?

Questions?


x >= 90
smd
ec
x < 90
x := 0


a
a′
a″
b


Output Channels
{u_i: }
Users
u_i
Smart Meter Data
Personal Information:
{u_i: e_j; ! u_i: !!e_j}
Smart Meter Readings:
{s_i: u_i, e_j(x[? reset: 1]^{!s_i:} > 90)[* reset]; ! s_i: }
Smart Meter Firmware:
{! m_x: }
Manufactures
m_x
Smart Meters
s_i
Distribution Companies
d_k
Data Hub
Processed Smart Meter Readings
{s_i: u_i, dh, e_j(x[? reset: 1]^{!s_i:} > 90)[* reset]; ! s_i: !!dh }
Electrical Companies
e_j
Reports
{dh: g; ! dh: }
Government

Conclusion

- Our model is decentralised
- Time is hard!
- This is initial work.
- Verification is a nice side effect (for now).
- What should we do next?
  - Flow locks?