A Formal Security Model for Verification of Automotive Embedded Applications

SAFA 2010

Gabriel Pedroza, Ludovic Apvrille & Renaud Pacalet

gabriel.pedroza@telecom-paristech.fr

Outline

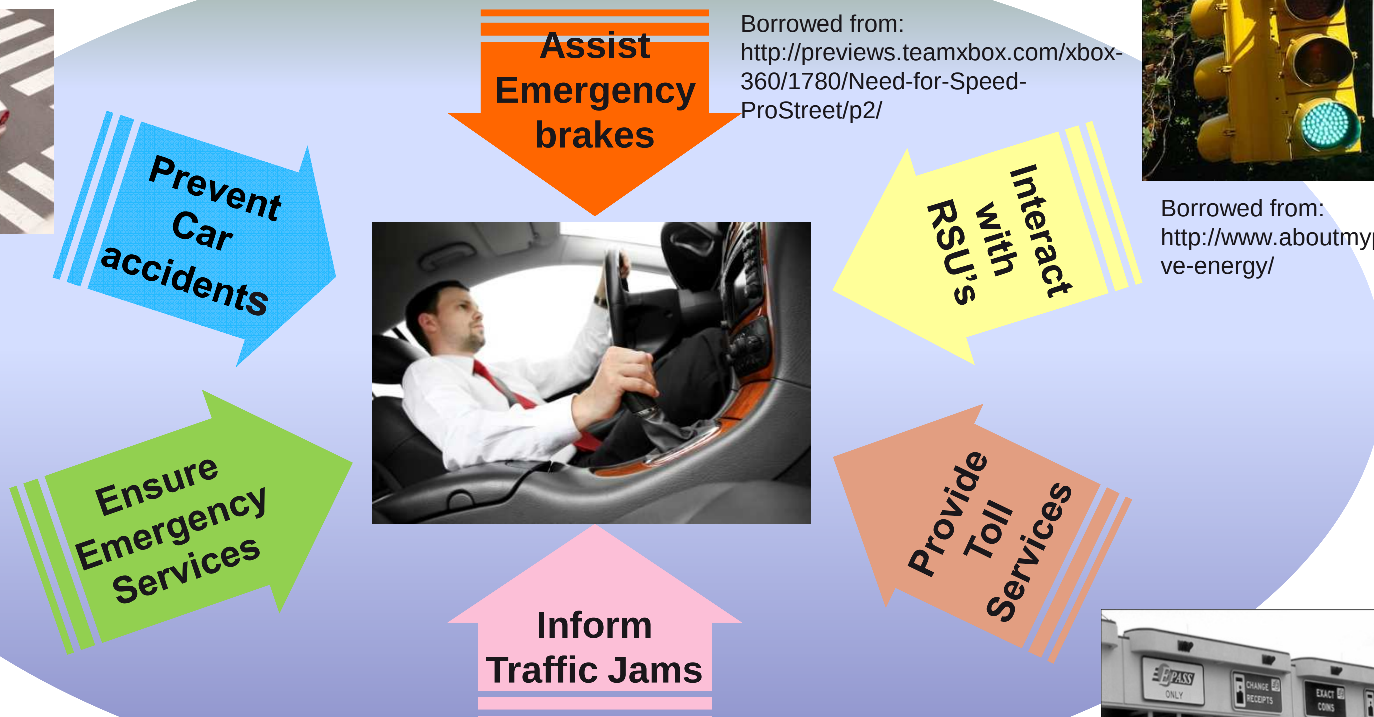
- Context
- Addressed problematic
- Proposed methodology
  - Formal Security Model
- Demos
- Conclusions and next steps

ITS aims to: Reduce fatalities!

Borrowed from: http://previews.teamxbox.com/xbox-360/1780/Need-for-Speed-ProStreet/p2/

Borrowed from: http://www.carinsurancecomparison.com/

Borrowed from: http://www.aboutmyplanet.com/alternative-energy/



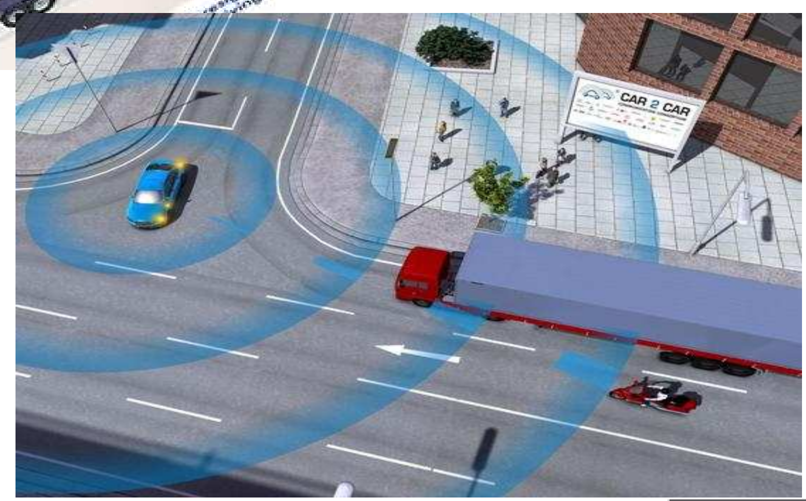
Borrowed from: http://www.mirror.co.uk/news/

Borrowed from: http://www.pbworld.com/news_events/

Borrowed from: http://wereviewyousave.com/some-tips-that-will-help-you-avoid-a-car-accident-3/

ITS paradigm

- Interactions between vehicles, road side infrastructure, and remote car-driver services would lead to safer and more efficient roads.



Images borrowed from Car2Car Communication Consortium. In http://www.car-to-car.org/

TELECOM ParisTech

ITS information chain

■ Correct ITS operation

Obstacle sensing

(1)

In-car reaction

(2)

Sending Alert
(Car2Car Communication)

Obstacle prevention:
- Slow down
- Shift way
- Emergency maneuver

■ Altering ITS operation

Sniff and store old alerts

(3)

Send faked alert

Virtual obstacle

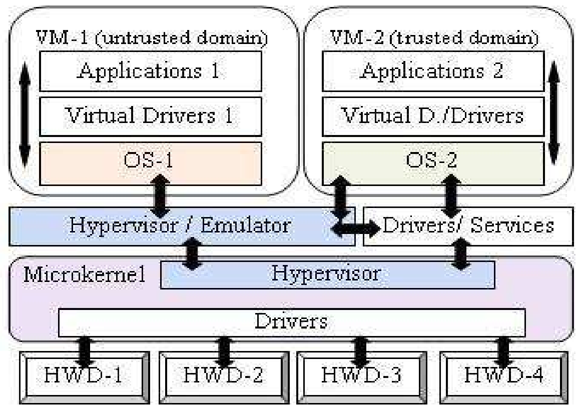
Unnecessary obstacle prevention may lead to traffic jams or accidents

Images borrowed from: http://booty-bootcamp.com/blog/ (1), http://www.khulsey.com/demo_howto_car.html (2), http://www.darkgovernment.com/news/tag/cyber-warfare/ (3).

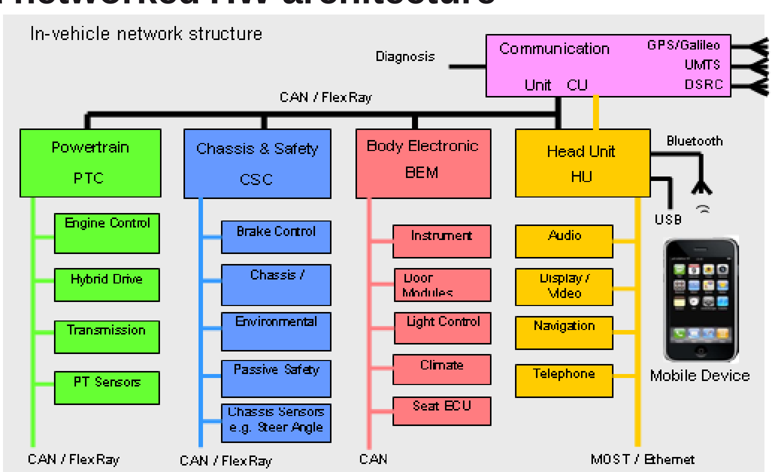
ITS' protection

- In-car applications should be protected to prevent attacks but …

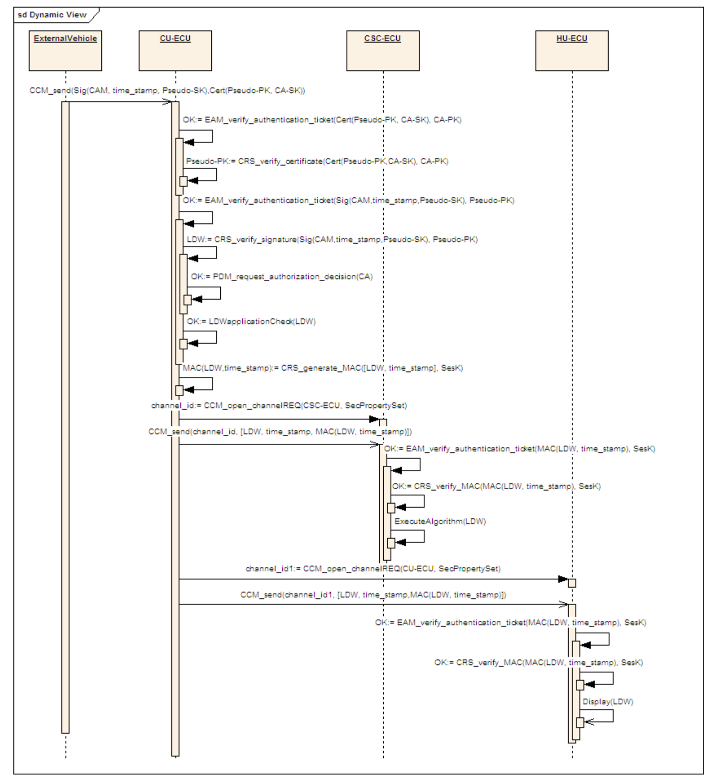
Complex SW architecture *



Distributed networked HW architecture *



Real time constrained execution of Security Protocols *



(Enlarge this figure)

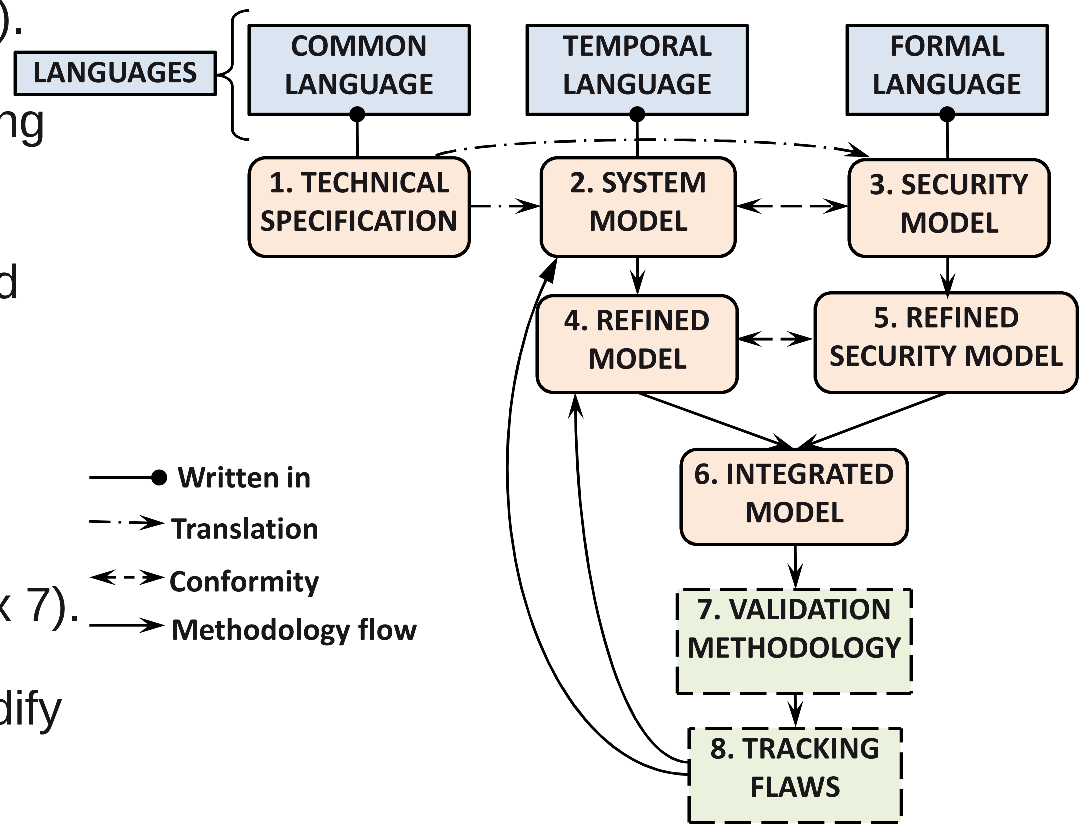
* Borrowed from EVITA technical reports D3.2 and D3.3. In http://www.evita-project.org

Main concerns with car applications protection

- We aim to address:
  1. Adequate specification and representation of security properties.
  2. Formal specification of security properties.
  3. Representation of automotive communication systems in a model (HW-SW).
  4. Verification of properties in automotive models.

Generic Verification Methodology Flow

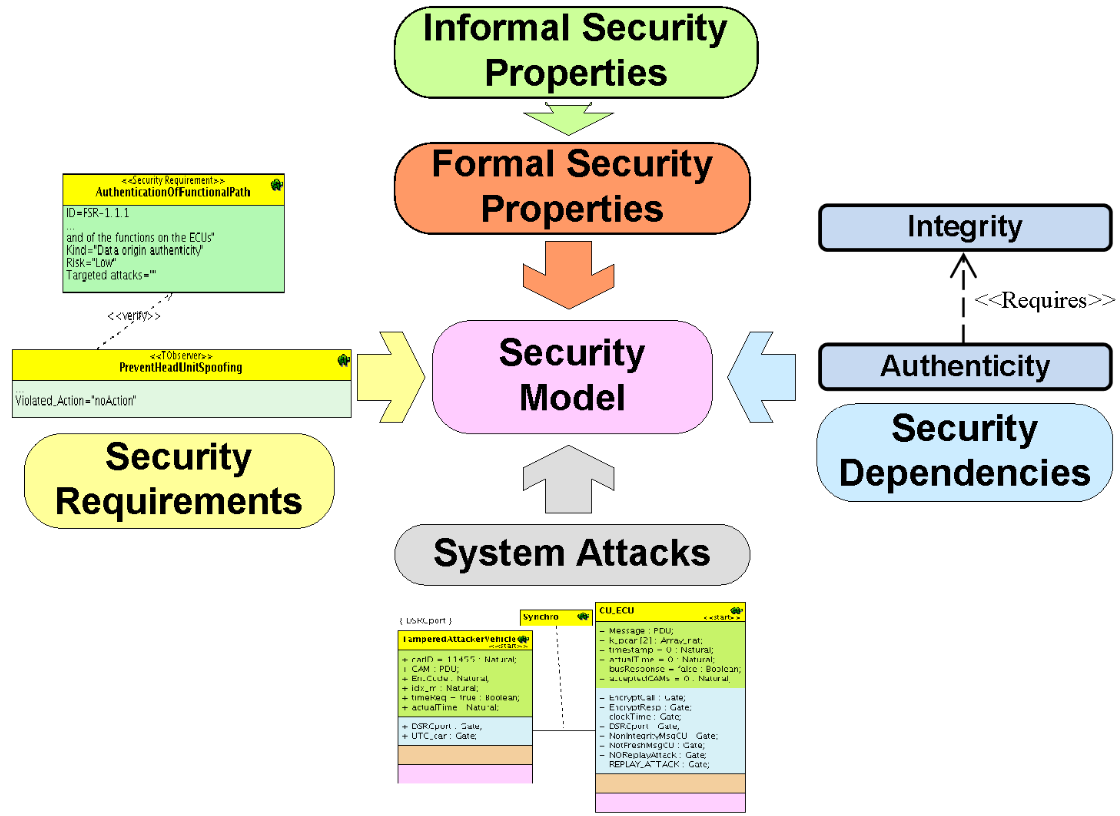
1. Translation of a technical specification to a System and Security Model (box 1).
2. System and security model decoupling (Boxes 2 and 3).
3. If possible, refinement of security and system models (Boxes 4 and 5).
4. Integration of Security and System models (Box 6).
5. Perform validation methodology (Box 7).
6. Identify and track flaws (Box 8). Modify system model up to achieve security properties accomplishment.



Formal Security Model (FSecM)

- FSecM targets model checking and integrates:

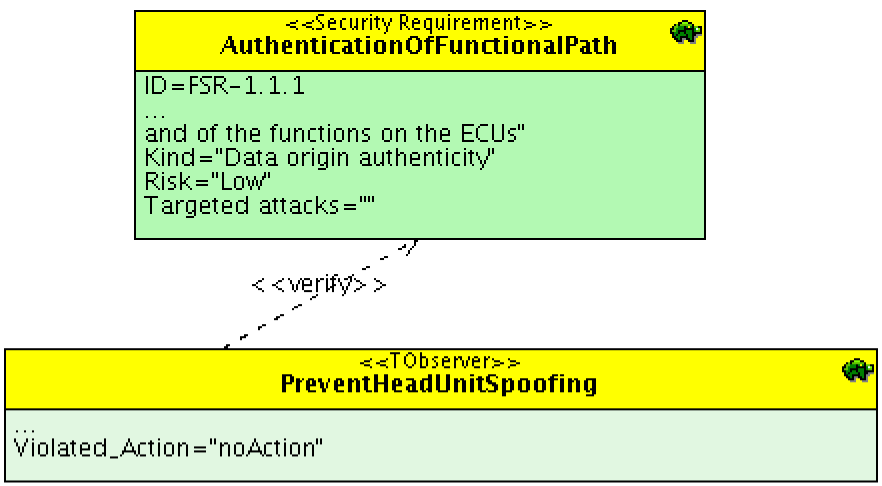
  1. Security Requirements.
  2. Formal security properties derived/synthesized from informal ones.
  3. Analysis and formalization of security properties dependencies.
  4. System attacks are written in the same language than system model.



Security Requirements Representation

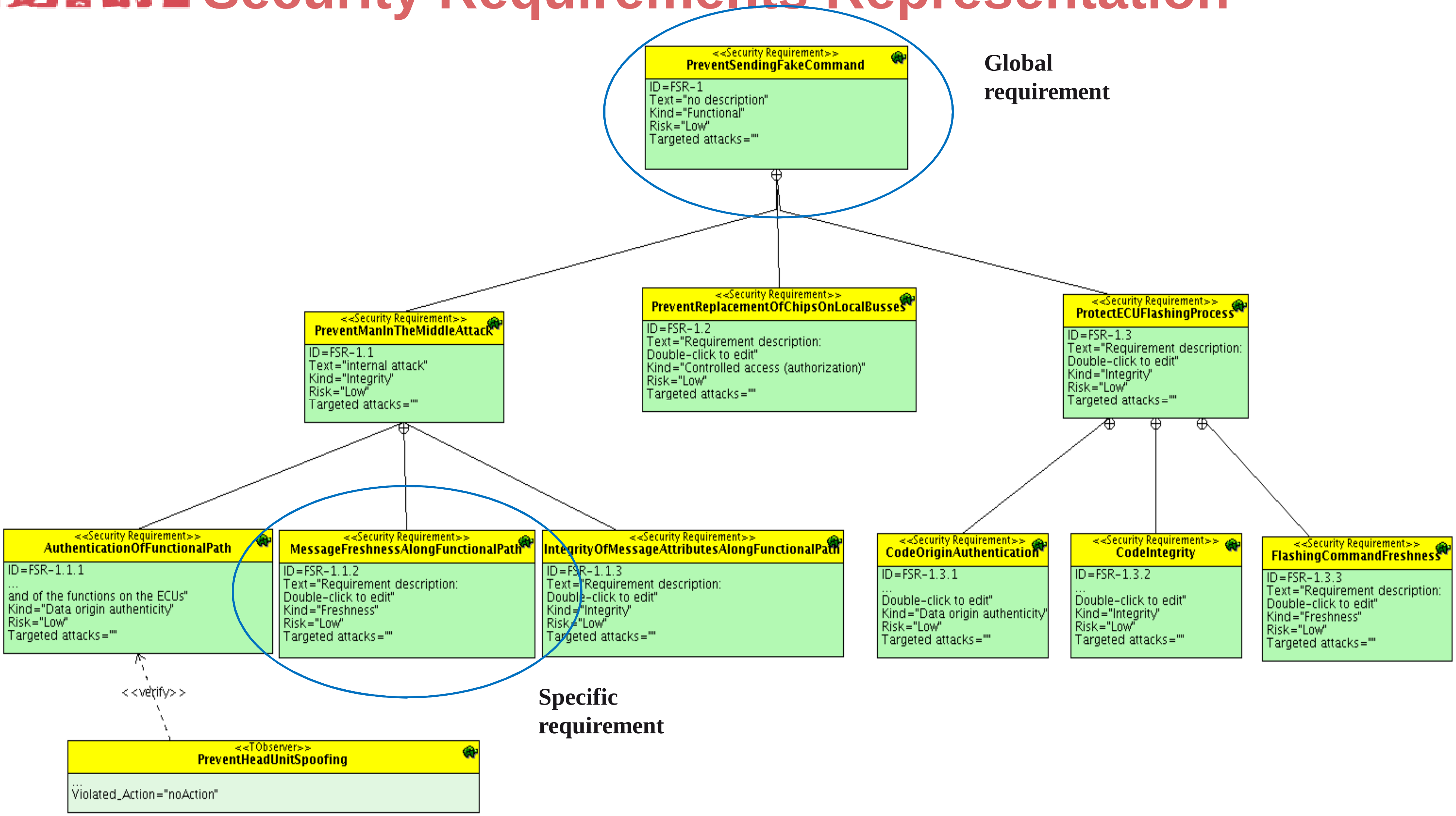
- *Security Requirements* :
  - » represented in an informal way
  - » SysML Security Requirement Diagrams (SRD)

- *SysML Diagrams:*
  - » conform directed hierarchy graphs
  - » nodes are security requirements
  - » edges adopt several compositional semantics



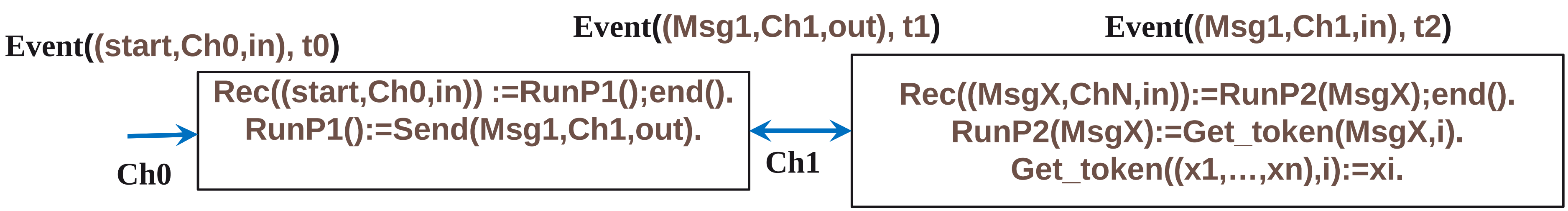
Edge	Semantics
A —⊕ B	reqB composed by reqA
A deriveReq B	reqB derive reqA
A copy B	reqA is a copy of reqB
O verify A	O verifies reqA.

Security Requirements Representation



Formalizing Security Properties

- Functional Path is a pair *(F,C)**:



- *C* is the set of channels and events in the use case; *C*:={Ch0, Ch1, Event((start,Ch0,in), t0), Event((MsgX,Ch1,out), t1), Event((MsgY,Ch1,in), t2)}.
- *F:=Fc∪Fo*; *Fc* is the set of functions that produce commands; *Fc*:={ Rec(), RunP1(), RunP2() }
- *Fo* is the set of the functions that output data in channels of *C*; *Fo*:={ Send() }

* Borrowed from EVITA technical report D2.3.

Formalizing Security Properties

- Security Requirements:
  - relate elements in a functional path with an (informal) Security Property definition.

- Security Requirement: *"Integrity of message attributes along functional path."*

- Definition of Integrity: *"An integrity property applies to a quantum of information between two observations (defined, e.g., by a time and a location in the system). The property is satisfied when the quantum of information has not been modified between the two observations."*

TELECOM ParisTech

Formalizing Security Properties

- First Approach:
  - Use a formal language
  - directly represent the security property semantics (e.g. CTL, LTL, TLA, etc.)

- Second Approach:
  - Represent the security property as a System Observer
  - Use a framework that allows translation to a formal specification (e.g. TURTLE)

Experience with UPPAAL and CTL (1st)

- Security properties as CTL queries:
  - AG(A1.sendM → AF(A2.receiveM1 and M1=M))
  - ECU1__5.id1094==1 -- > (ECUKM__3.id939==1 and ECU1__5.Message1__data0== ECUKM__3.Message__data0)
  - Semantics:
    1. For all paths from the initial state of the system …
    2. Whenever A1 sends M …
    3. Implies that always A2 eventually receives M1 …
    4. and M1=M.

Experience with UPPAAL and CTL (1st)

- Main limitations:
  - Level of abstraction:
    - Queries are related to specific states;
      - » E.g. activation of the state id1094 in ECU1: ECU1__5.id1094==1
    - The queries should be repeated;
      - » Different events associated to different states
  - Level of refinement:
    - Verification of events during an specific period of time;
      - » E.g. freshness
    - BUT: not provided by CTL;
      - » Temporal quantifiers: unable to express specific time

Dependencies in Security Properties

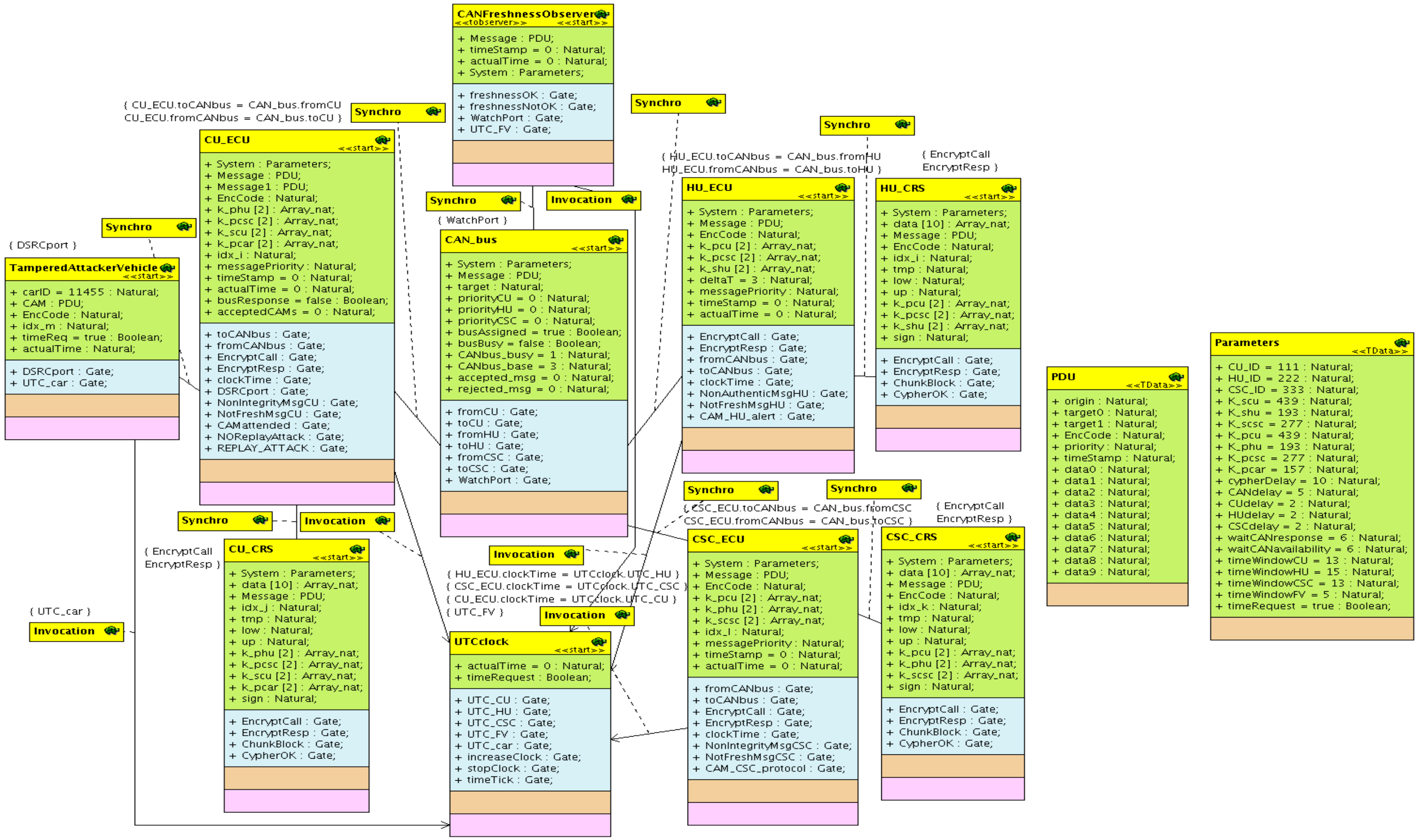
- Aim to provide an order for property verification
- Dependencies take into account:
- *Verifier viewpoint*:
  - » Events ignored by the verifier
  - » e.g. in a local view
- *Model hypotheses*:
  - » Properties assured by assumptions
  - » e.g. a secure channel
- *Target of the property*:
  - » e.g. data origin authenticity →integrity
- Different schemes of dependencies; they are Use Case oriented

Attacker Paradigm

- Based upon the Dolev-Yao approach
- Direct interaction between the attacker and the system model
- Attacker represented in the same language as the system
- Attacker behavior based upon an specific attack goal
- Addressed attacks; the result of a threat and risk analysis
- Threat and risk analysis based upon specific HW-SW architecture

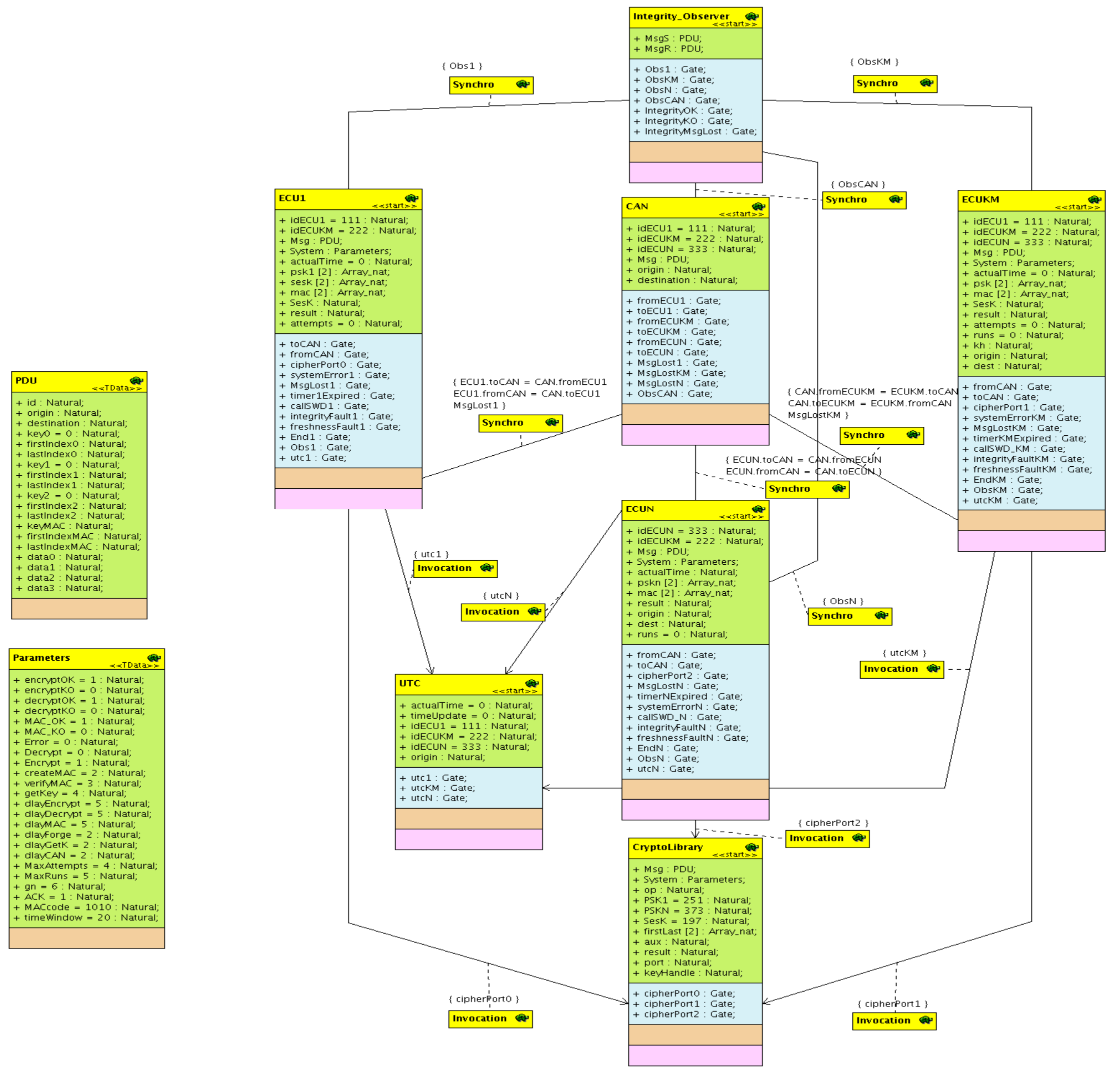
Demo1: FSecM integration, 1st Approach

- A Target of Verification (ToV) in the TURTLE framework



Demo2: FSecM integration, 2nd Approach

- Verification of security protocol in the TURTLE framework using Security Observers



Conclusions and Next Steps

- 1st Approach:
  - Security requirements represented in CTL
  - Verified with UPPAAL
  - BUT: Insufficient CTL semantics
- 2nd Approach:
  - Security Properties represented as Observers
  - Using the TURTLE framework
  - More flexible representation
  - BUT: need for a more rigorous methodology

Conclusions and Next Steps

- Specific attacks represented in the TURTLE framework
- BUT: Generic Dolev-Yao attackers require more elements:
  - e.g. For exhaustively exploring attacker-protocol interactions space.
- New formal languages are explored:
  - pi-calculus, spi-calculus.
- And automated tools:
  - ProVerif
- New insights in the research are expected