# PhD Defense

Morten Dahl Jørgensen

Department of Computer Science
Aarhus University

19 June 2013

# PhD Overview

## Symbolic Analysis of Cryptographic Protocols

Authenticity Privacy Universal Composability

## Secure Computation

Division Fresco

# Symbolic Analysis of Cryptographic Protocols

# Symbolic Analysis of Cryptographic Protocols

Develop methods to assist system designers and implementors in verifying that their creations do not contain security flaws

→ **in particular w.r.t. misuse of cryptographic techniques**

Main approach: simplify via abstraction

- aid manual efforts
- allow automated tools
- reduce required expertise

For which protocols & properties can this be done?

# Symbolic Analysis of **Cryptographic Protocols**

- A **protocol** is a "recipe" for a set of players that describes what steps they can take in order to perform a specific **task**
  - example: French Greeting

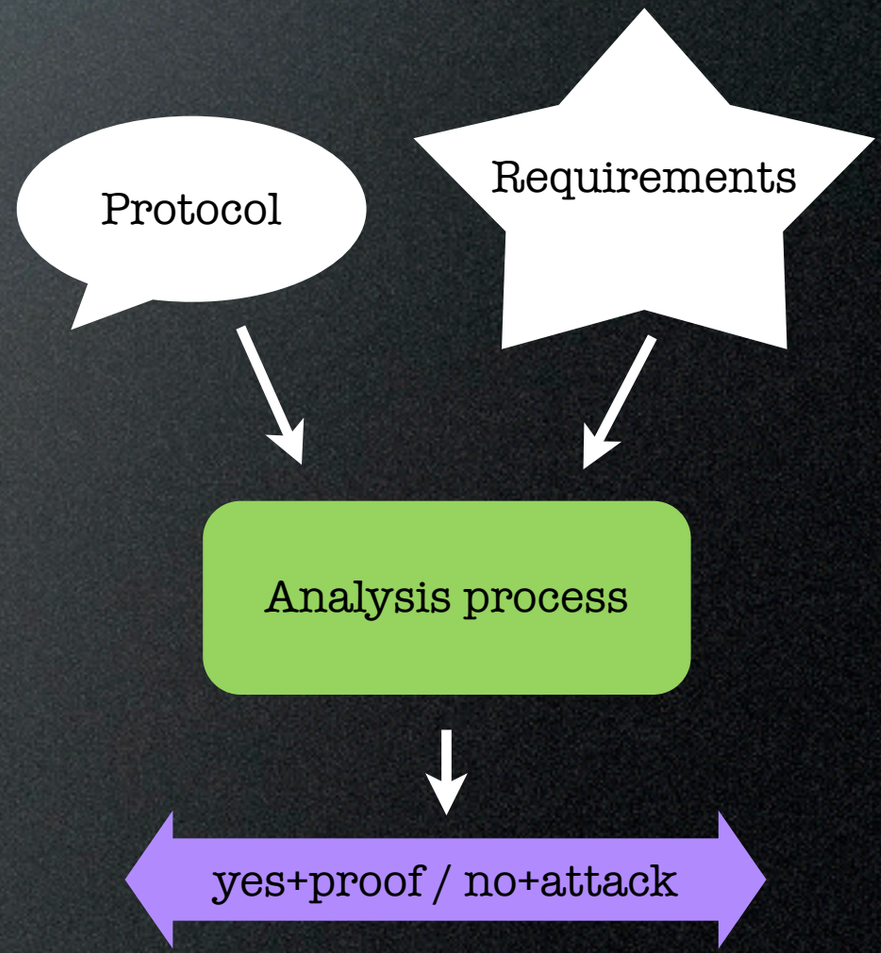
- A **cryptographic protocol** employs cryptographic primitives
  - example: Secure Email

- **Systems** use these protocols as sub-components:
  - online banking: "send secure email"
  - websites: "verify password"

# Symbolic **Analysis** of Cryptographic Protocols

Mathematical argument explaining why a protocol is “secure”

- a **security requirement** determines what secure means
- need mathematical model




Note, no focus on:

- social engineering (phishing)
- policy flaw
- physical properties of hardware
- software bug in implementation

# **Symbolic** Analysis of Cryptographic Protocols

Computational — Symbolic

- good model of the real world
- computation on bitstrings
- flexible operations
- complex analysis

$$2 \cdot \pi = 6.28318530718$$

$$\frac{2 \cdot \pi}{4} = 1.57079632679$$

- high abstraction level
- symbolic manipulation
- restricted operations
- simple analysis

$$2 \cdot \pi = 2 \cdot \pi$$

$$\frac{2 \cdot \pi}{4} = \frac{1}{2}\pi$$

# **Symbolic** Analysis of Cryptographic Protocols

Computational — Symbolic

| | | |
|---|---|---|
| keys, nonces, randomness | long random bitstrings | unguessable atomic symbols |
| ciphertexts, etc. | bitstrings: c<br>unlimited manipulation | terms: **enc**(**m**,**k**,**r**)<br>rules for manipulation |
| attacker | no restrictions besides limited computing power | only few selected operations |

# Our Work

| | Authenticity | Privacy | UC |
|---|---|---|---|
| properties | simple | intermediate | advanced |
| primitives | simple | simple | advanced |
| motivation | automated analysis | concrete system | computational sound |

# Authenticity Analysis

## [DKSH11]

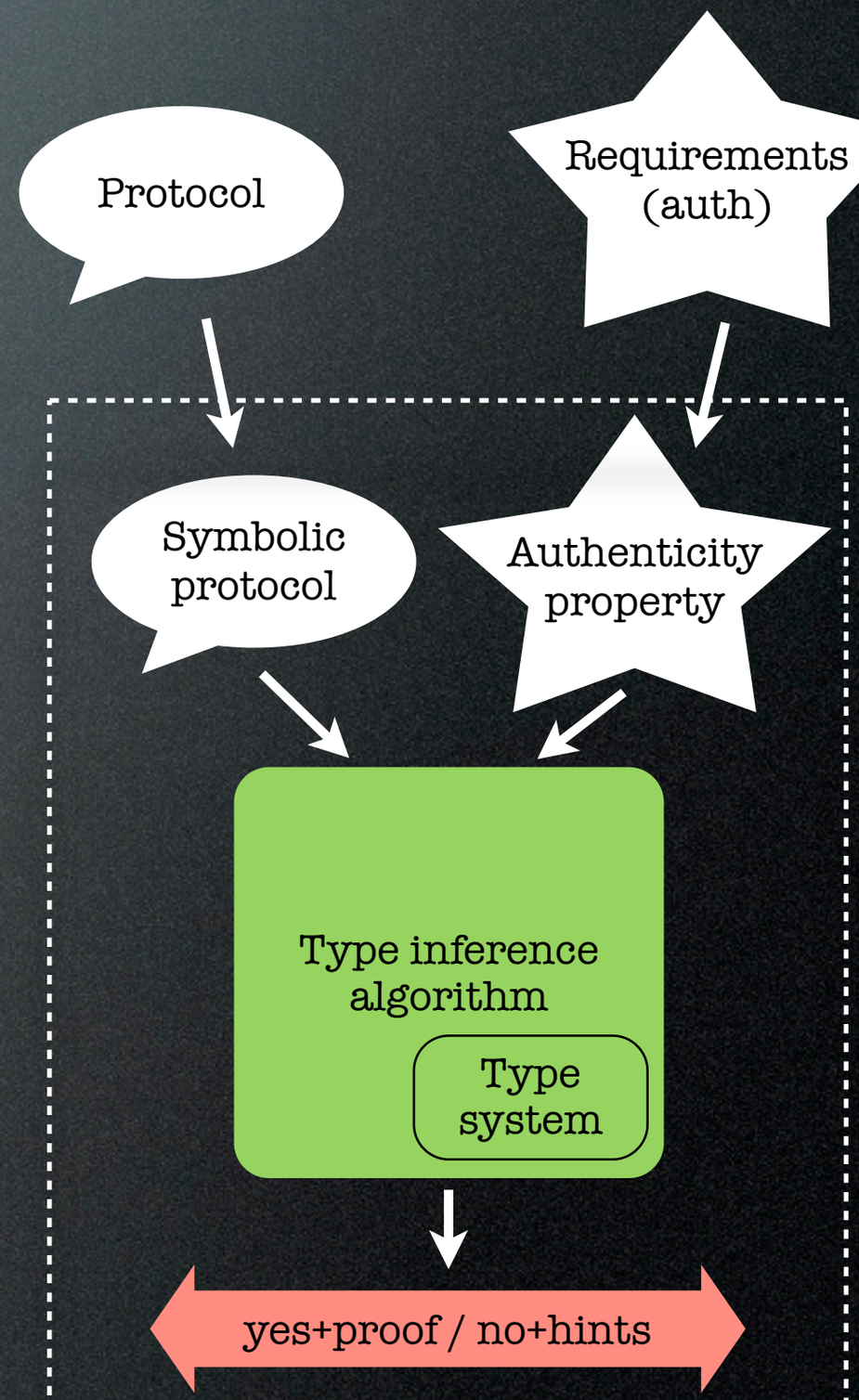
Joint work with Naoki Kobayashi, Yunde Sun, and Hans Hüttel; paper published at ATVA'11

In essence we:

- **develop automatic analysis method for authenticity properties**
- use type system to prove properties
- automate proof finding using type inference

Our main contributions:

- non-trivial modification of existing type system [GJ04] to support type inference
- bonus: capture multi-party protocols
- practical test of the algorithm's efficiency



# Authenticity Properties

- Informally: that data is of expected origin

- Formalised as correspondence assertions [WL93]
  - introduce **approve** and **expect** events
  - require that in all executions:
    - every **expect** must have been **approve**d
    - if so we say a correspondence exists

- Example: Authenticated Message

# Type System

Theorem: If a protocol type-checks then there always exists a correspondence

$$\frac{M : T \qquad \mathbf{Pub}(T)}{\mathbf{out}(ch, M); P} \qquad \frac{x : T \qquad \mathbf{Taint}(T)}{\mathbf{in}(ch, x); P}$$

$$\frac{\mathbf{Pub}(T) \qquad \mathbf{Taint}(T)}{\mathbf{Pub}(\mathbf{SKey}(T))}$$

$$\frac{\mathbf{Taint}(T)}{\mathbf{Pub}(\mathbf{EKey}(T))} \qquad \frac{\mathbf{Pub}(T)}{\mathbf{Pub}(\mathbf{DKey}(T))}$$

... leads to an accumulation of constraints

# Plus and Minus

Strengths:

- efficient algorithms and modular analysis
- moderate expert knowledge; programmer familiarity
- explicit verifiable proofs
- extendable to implementation-level analysis

Weaknesses:

- simple primitives and properties
  - many details hidden in the typing rules; expert-task to extend
- overly conservative (price of simplicity)
- may not be able to provide an explicit attack
- no real-world world guarantees

# Privacy Analysis
## [DDS10] and [DDS11]

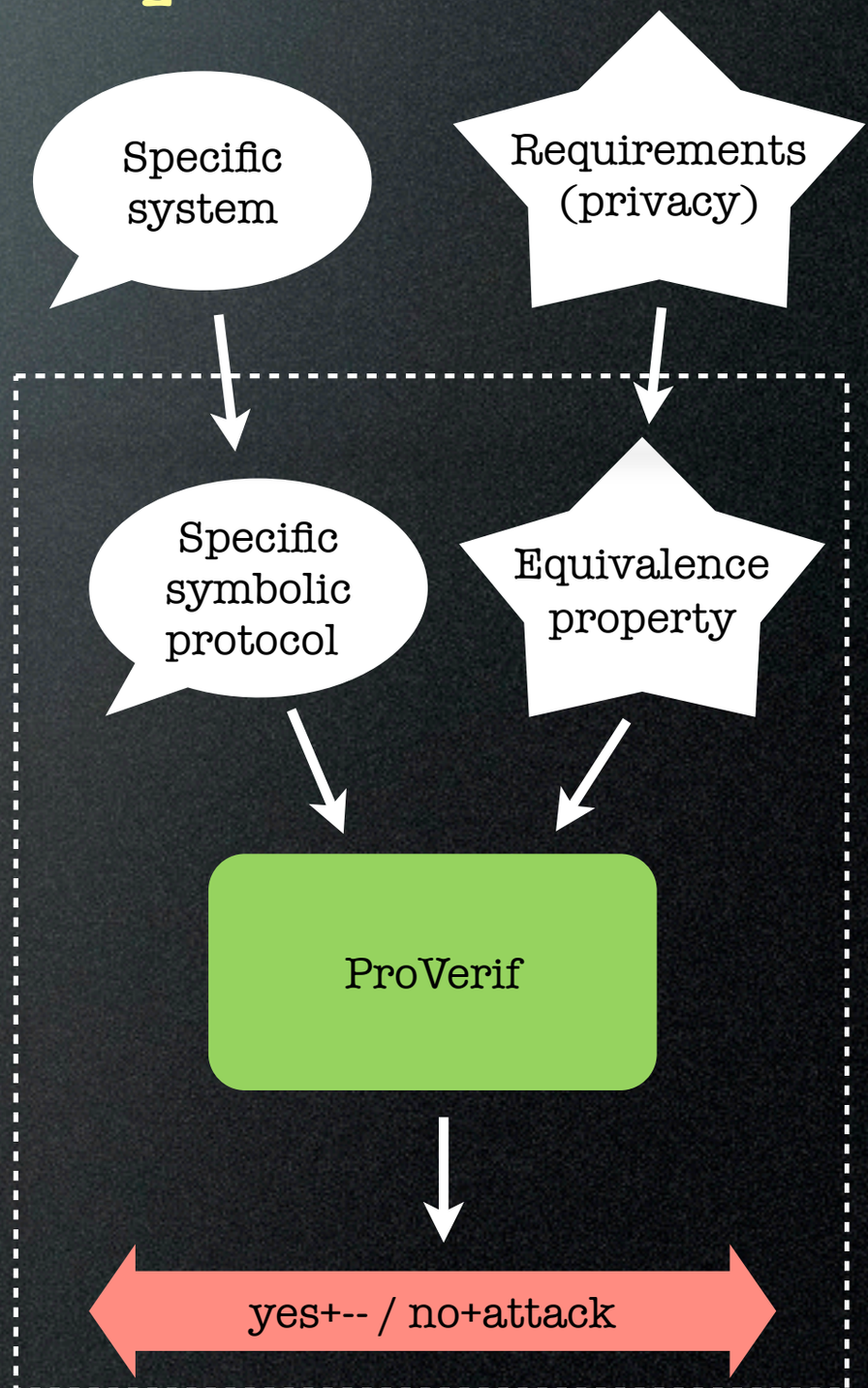
Joint work with Stéphanie Delaune and Graham Steel; papers published at ESORICS'10 and TOSCA'11

In essence we:

- **formally analyse two concrete systems w.r.t. privacy**
- formally express the two systems
- formulate suitable notions of privacy
- carry our analysis using the ProVerif tool

Our main contributions:

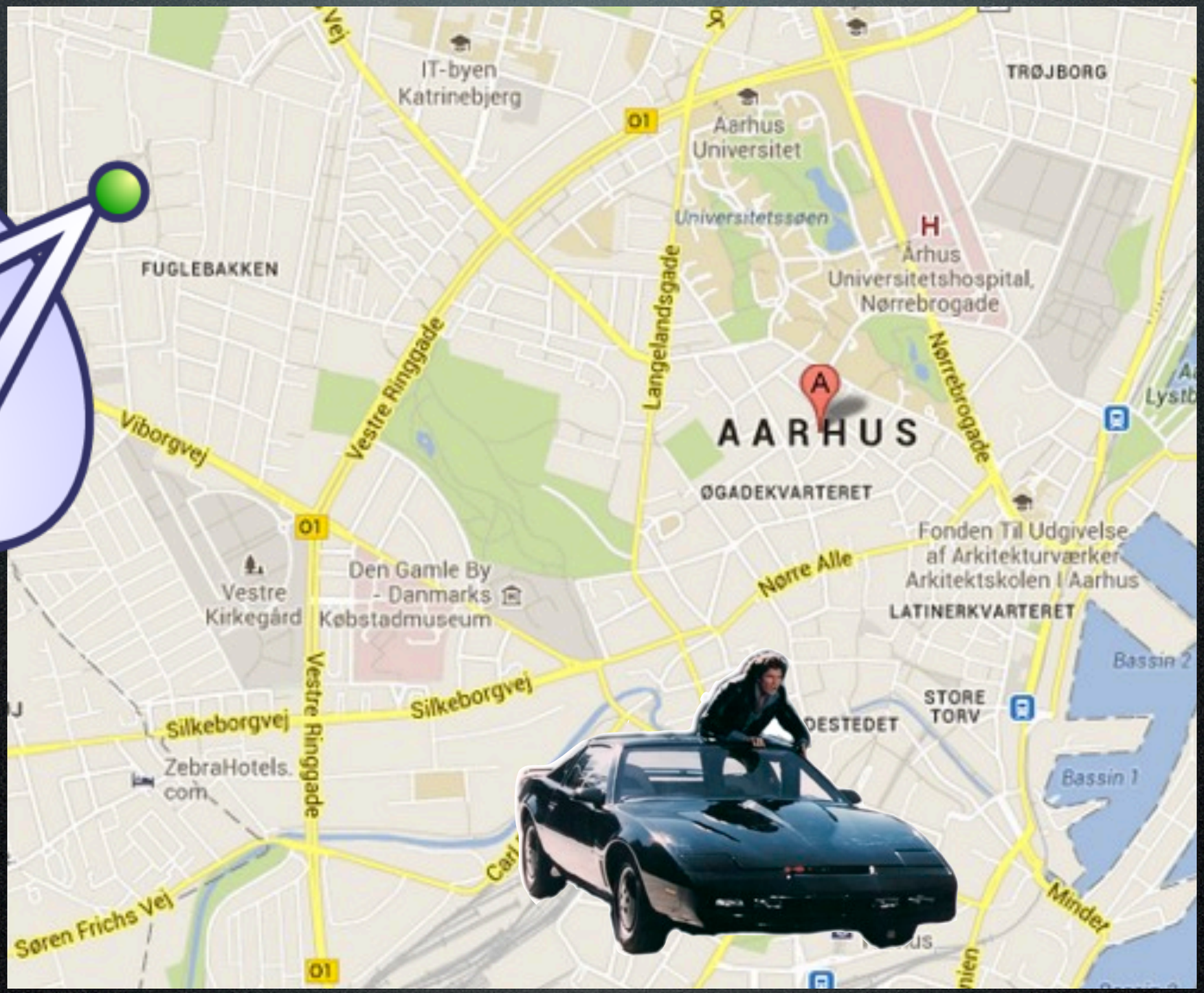
- further investigate the modelling of privacy by indistinguishability (also voting + RFID tags)
- report on analysis results
- investigate current level of tool support



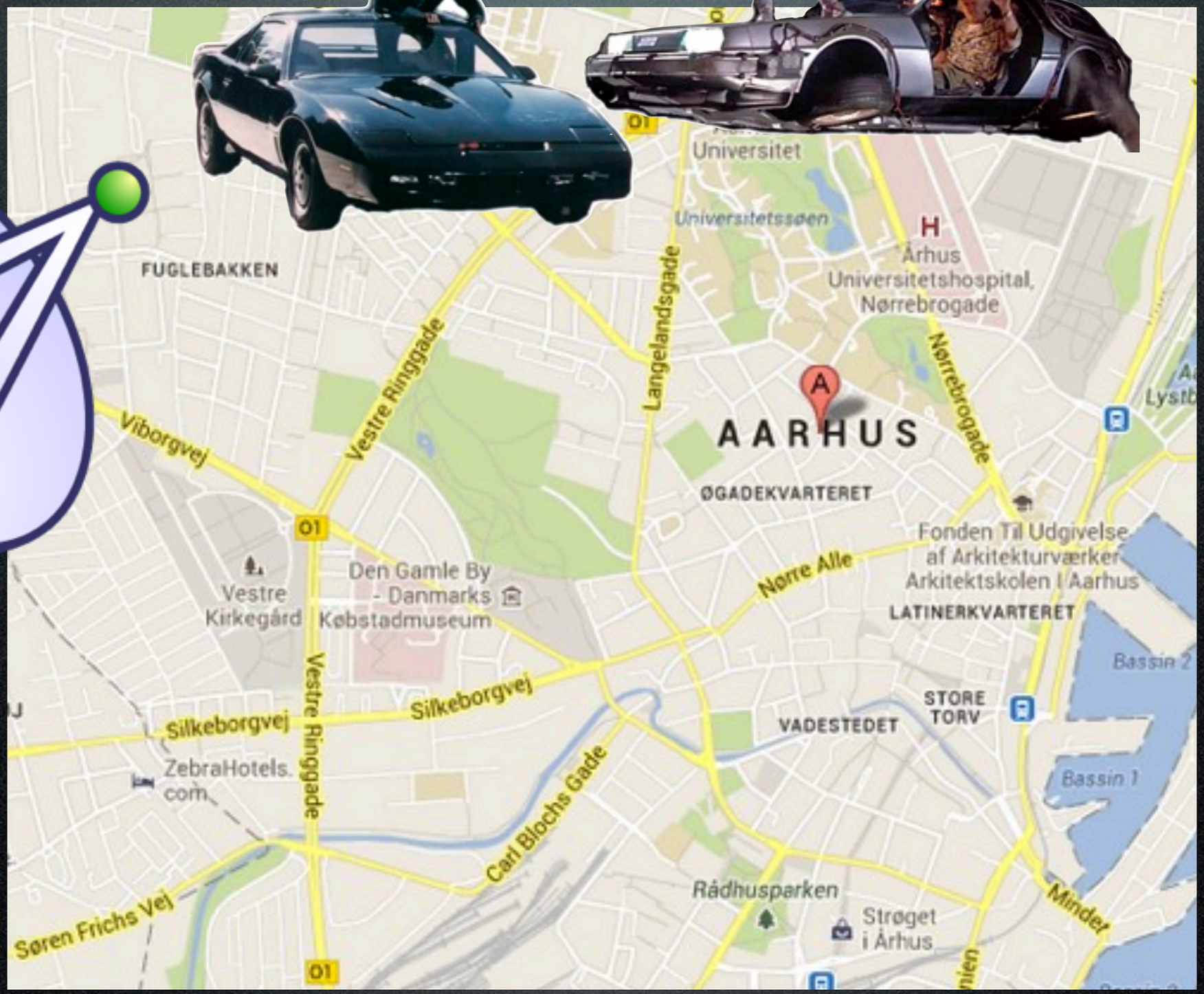
# The VPriv System

# The VPriv System

#746ab
#mcj48
#j37ck

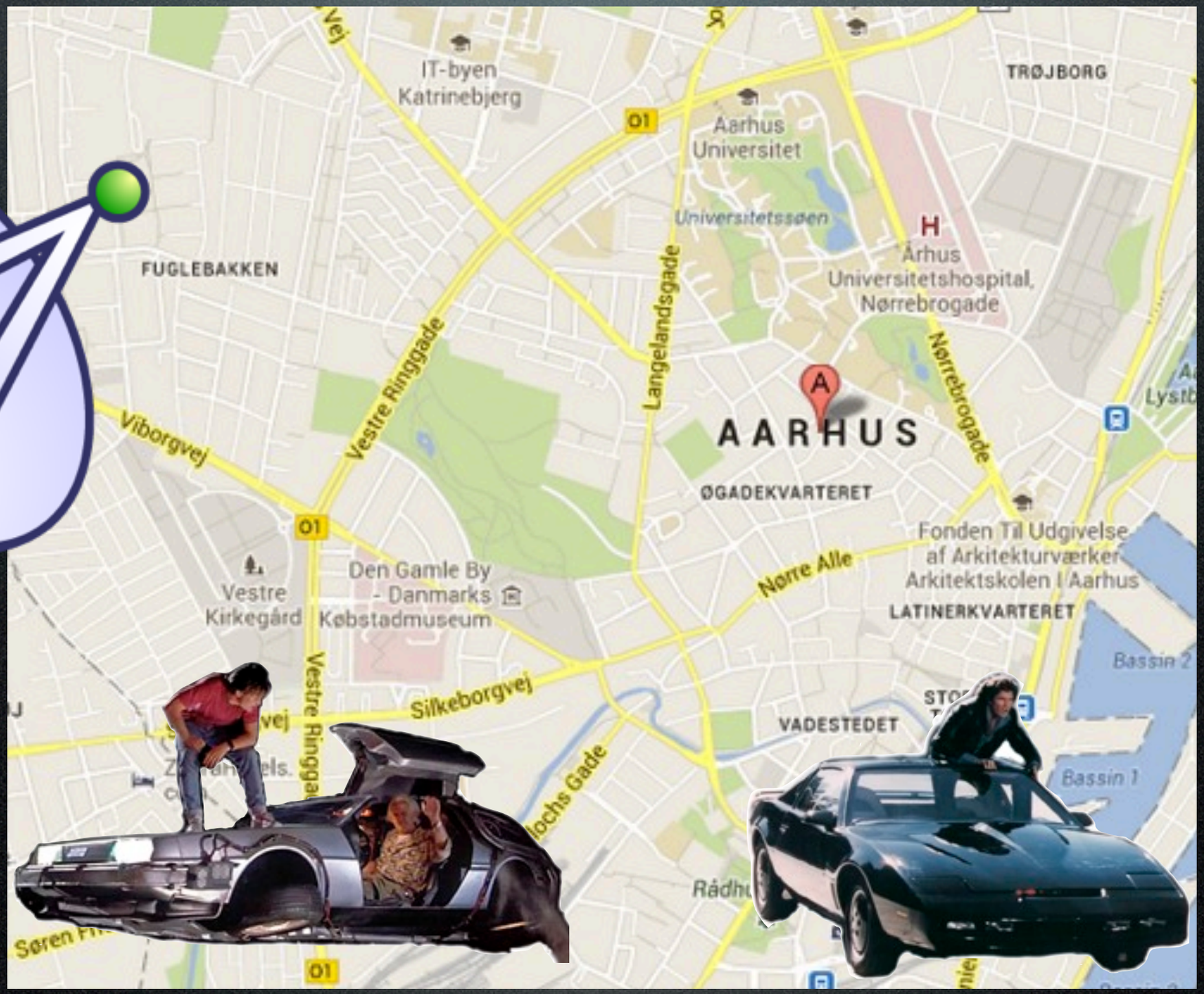


# Route Privacy



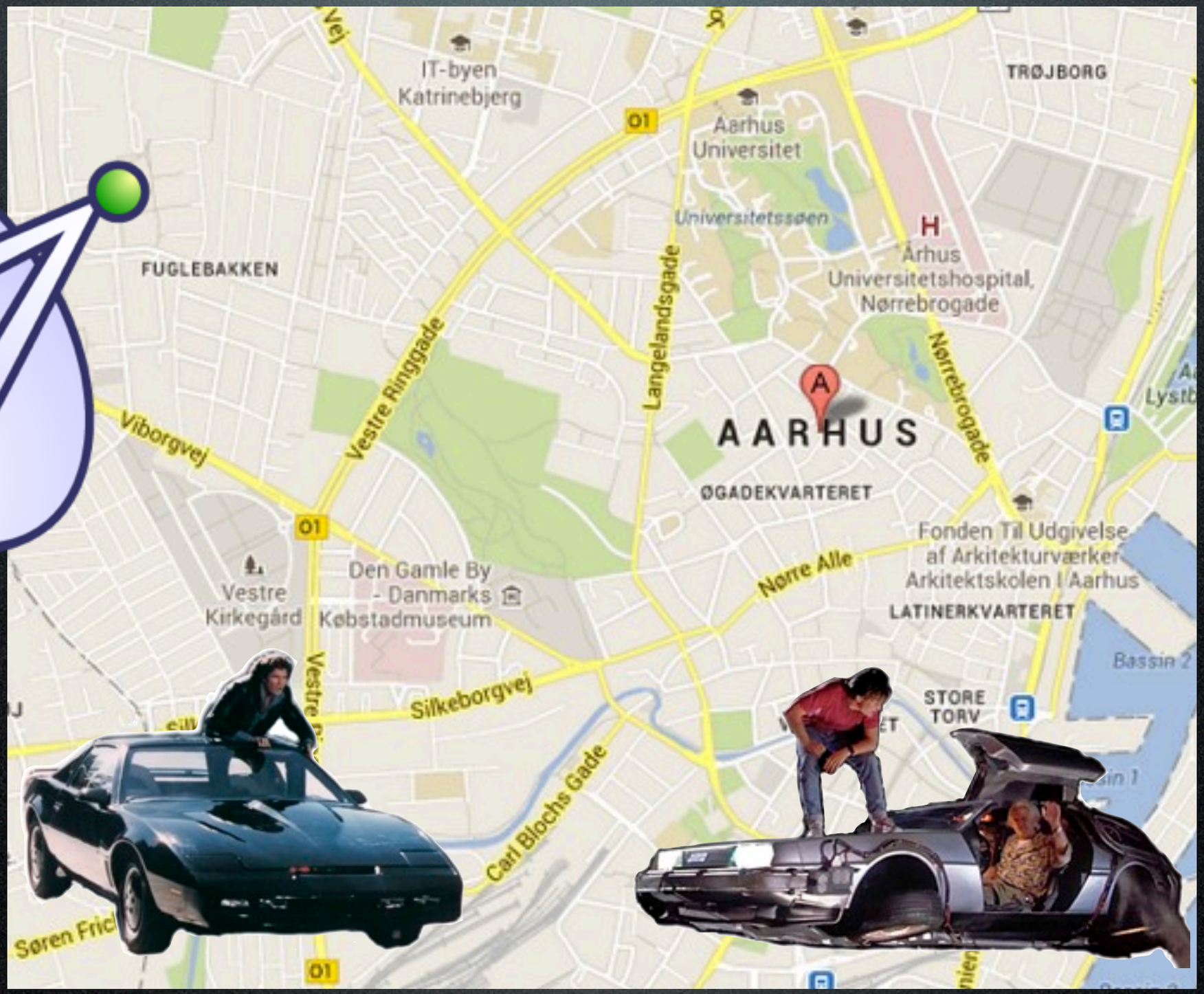
# Route Privacy

#83ncs
#9bv3d
#48cnl



# Route Privacy

# Route Privacy




#jdn24
#jn104
#103nc

# Privacy as an Equivalence

Privacy modelled as equivalence between two different behaviours

$$\mathcal{C}_{setup}\left[V_{kitt}(route_{left}) \mid V_{delorean}(route_{right})\right]$$

$$\sim$$

$$\mathcal{C}_{setup}\left[V_{delorean}(route_{left}) \mid V_{kitt}(route_{right})\right]$$

# Plus and Minus

Strengths:

- more powerful properties
- more flexible on primitives; easier to extend; easier to understand
  - nonces, symmetric encryption, asymmetric encryption, and signatures
  - nonces, commitments, hashing, and list permutations
- often we get a concrete attack trace

Weaknesses:

- requires more expert knowledge (modelling + tool operation)
- no explicit proof
- overly conservative (price of tool support for equivalence)
- no real-world world guarantees

# UC Analysis
## [DD13]

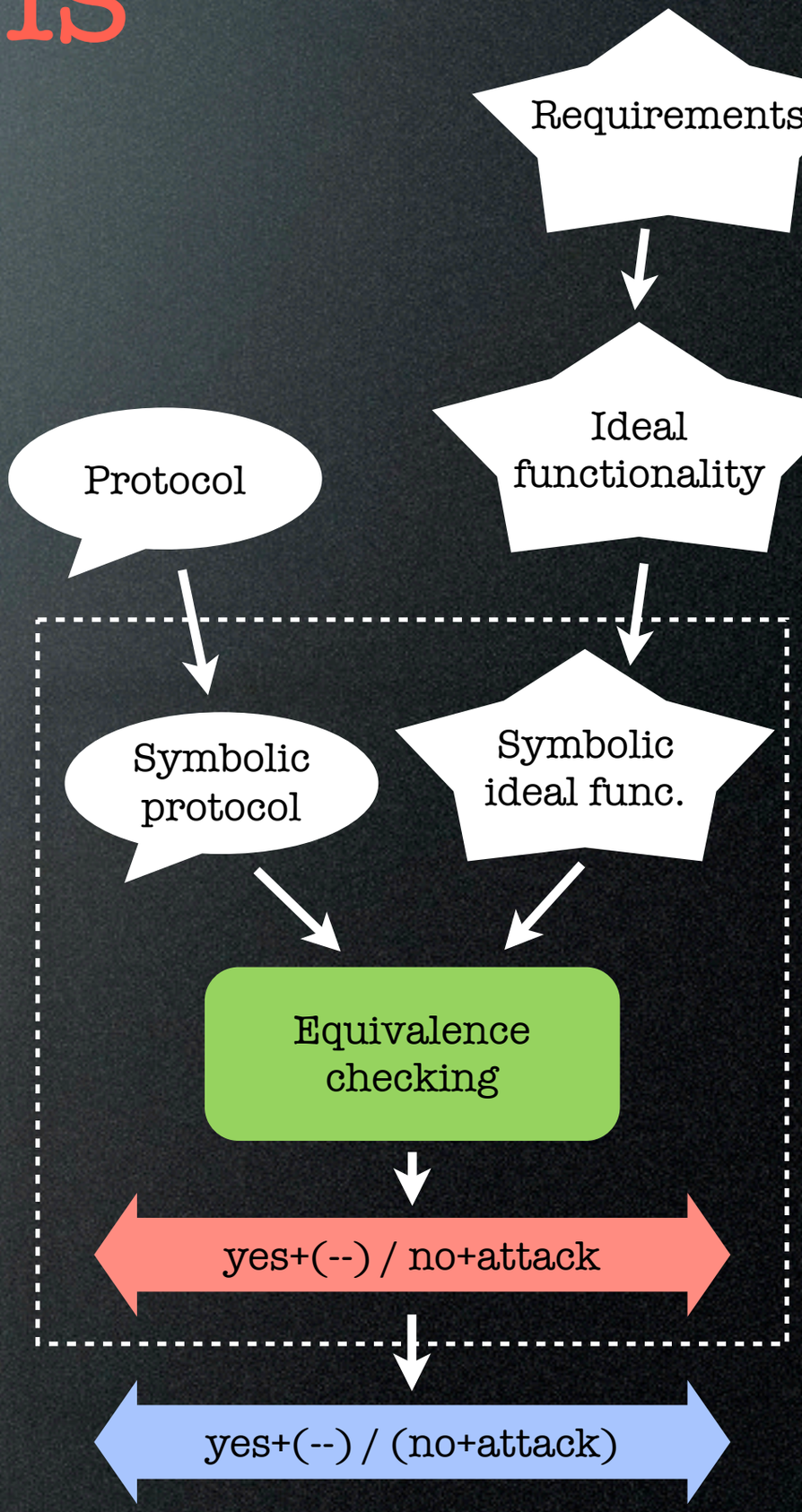
Joint work with Ivan Damgård; unpublished

In essence we:

- **develop framework for simplifying/automating the analysis of advanced protocols and properties in a sound and composable manner**
- formulate a class of powerful protocols
- give a general computational soundness result
- illustrate the method on a few examples

Our main contributions:

- show computational soundness of powerful primitives
- motivate the use of Universal Composability [Can05] in the symbolic setting
- analyse a concrete protocol using ProVerif
- list heuristics for automating the analysis



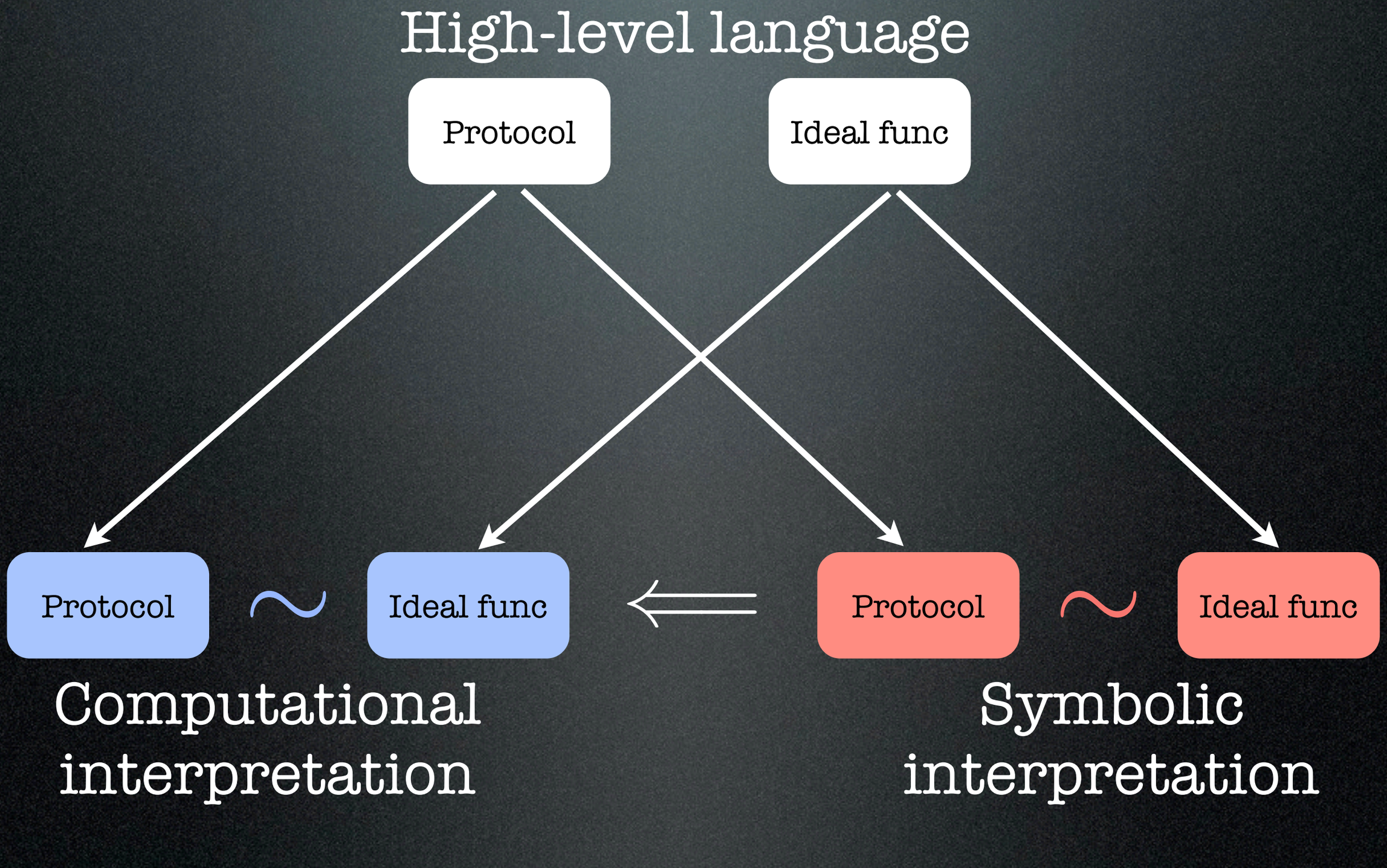
# Ideal Functionalities

- A **magic box** that players may use instead of a protocol
  - protocol: **how** a task is performed
  - ideal functionality: **what** a task does
    - including its security guarantees
  - verifying a protocol boils down to
    - checking an equivalence
    - constructing a simulator

- Example: Authenticated Message and Coin-Flipping

- Ideal functionalities for **compositional analysis**
  - and compositional design

# Computational Soundness



# Plus and Minus

Strengths:

- even more powerful properties
- powerful primitives:
  - homomorphic encryption, commitments, and zero-knowledge proofs
  - coin-flip, oblivious transfer, multiplication-triple generation
- real-world world guarantees
- modular and composable analysis
- (in some cases) suitable for current tools (ProVerif)

Weaknesses:

- requires expert knowledge
  - formulating ideal functionalities
  - partial proof construction (simulator)
  - tool operation
- fixed on primitives and two-party function evaluation protocols; expert-task to extend

# Summary

| | Authenticity | Privacy | Universal Comp. |
|---|---|---|---|
| properties | correspondence | equivalence | ideal functionality |
| primitives | encryption, signatures | encryption, signatures, commitments, hashing | homomorphic encryption, commitments, zero-knowledge proofs |
| expertise | automatic + efficient | modelling; tool support | ideal func. + simulator; some tool support |
| real-world | (extendable to source code) | real-world case study | computational sound |

# Thank you