UNIVERSITÄT
DES
SAARLANDES

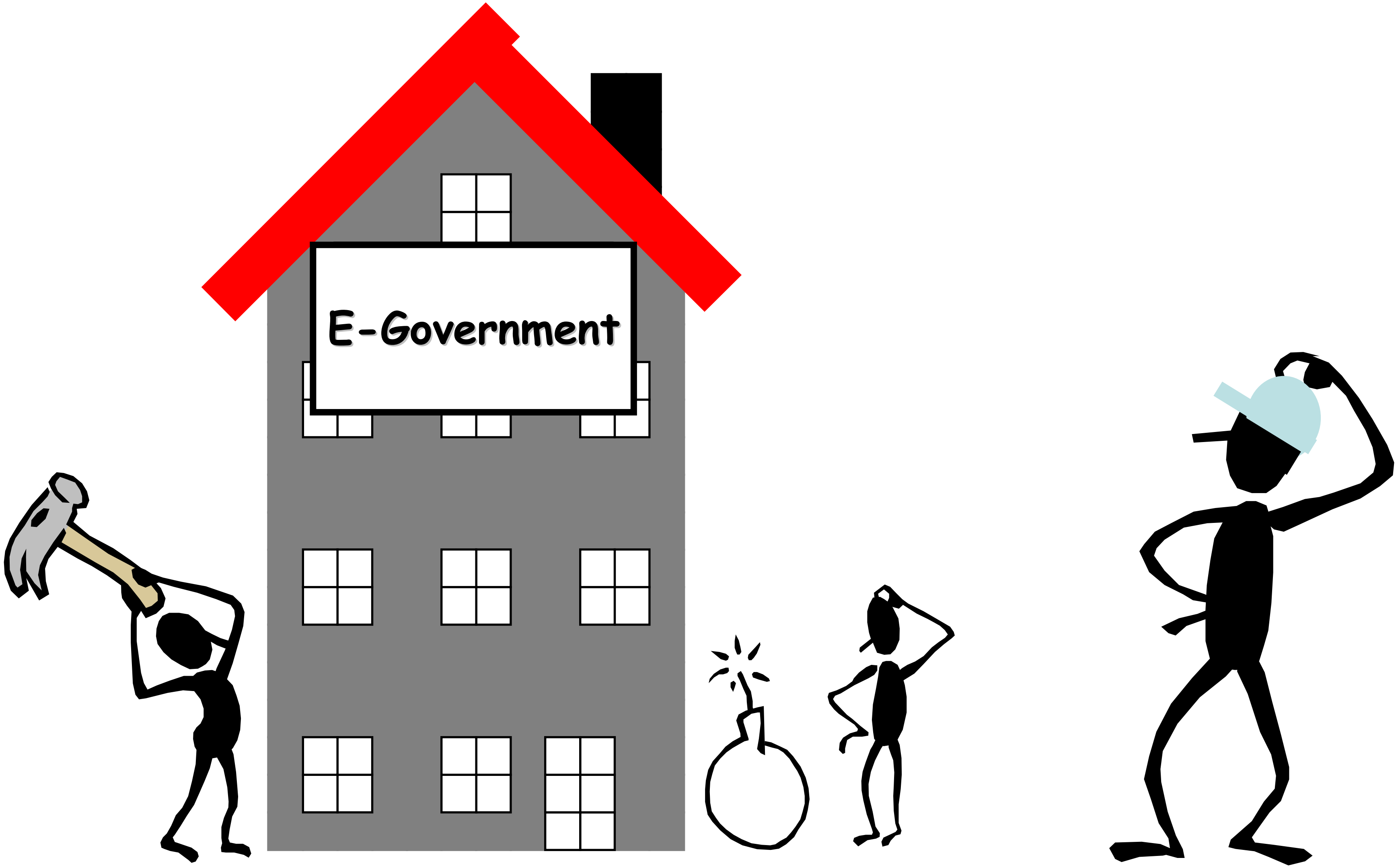
Max
Planck
Institute
for
Software Systems

On Reactive Simulatability

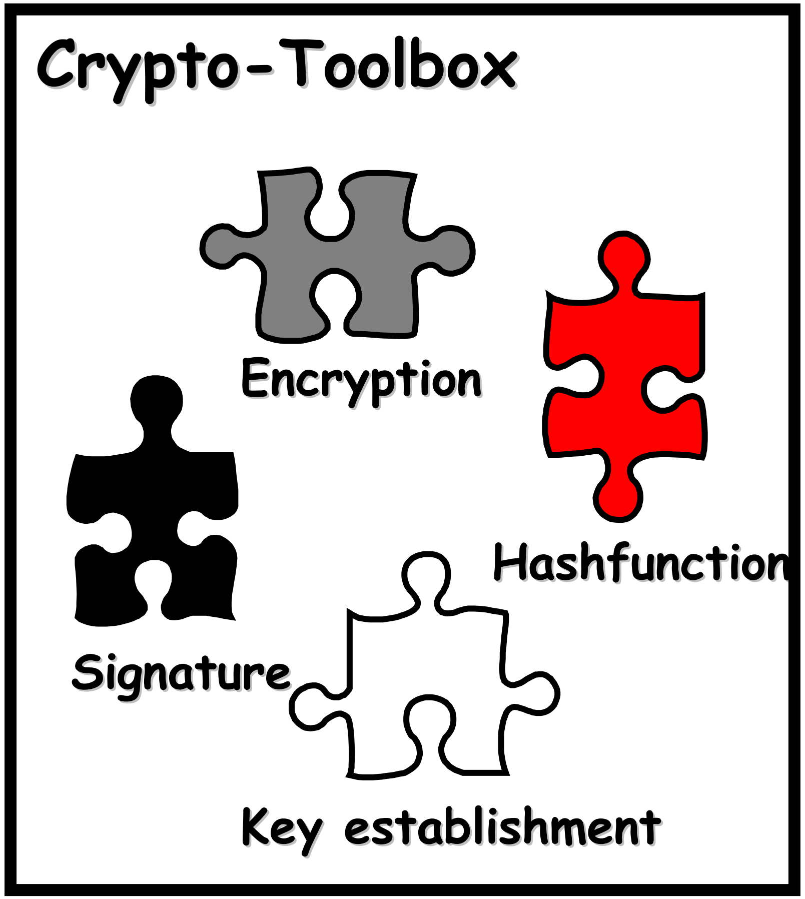
Matthias Berg / (Michael Backes)
Saarland U and MPI-SWS

CoSyProofs 2009

Building Systems on Open Networks



Cryptography: The Details



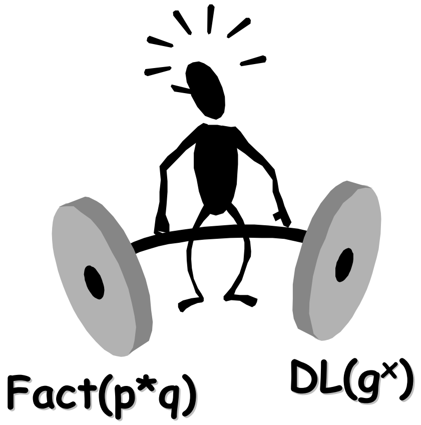
Crypto-Toolbox

Encryption

Hashfunction

Signature

Key establishment

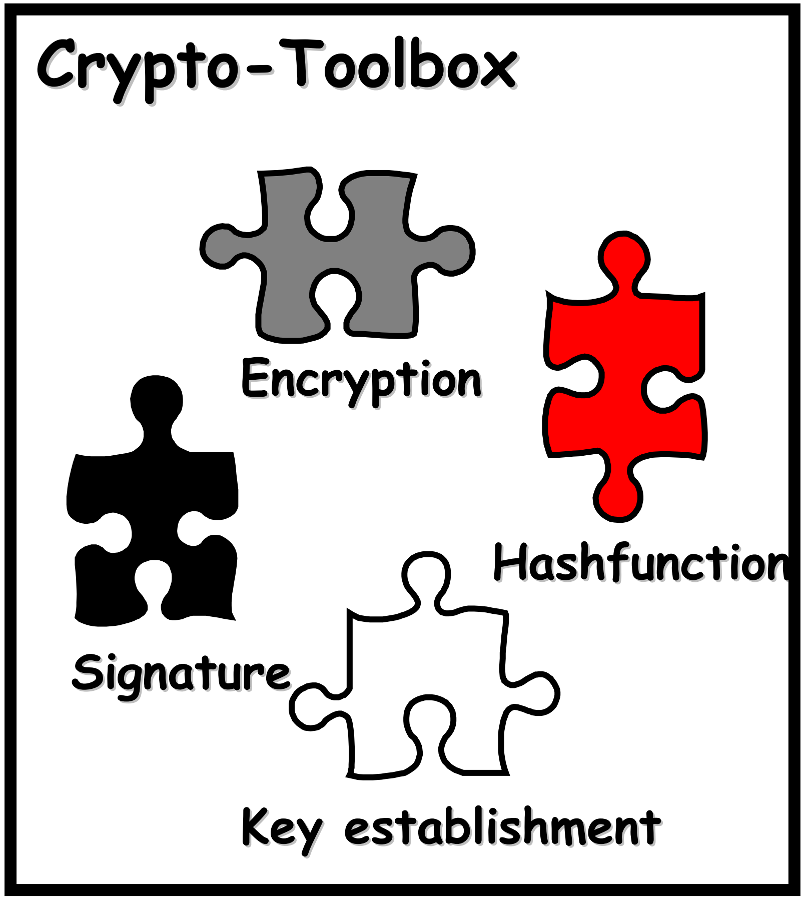


Fact(p*q)

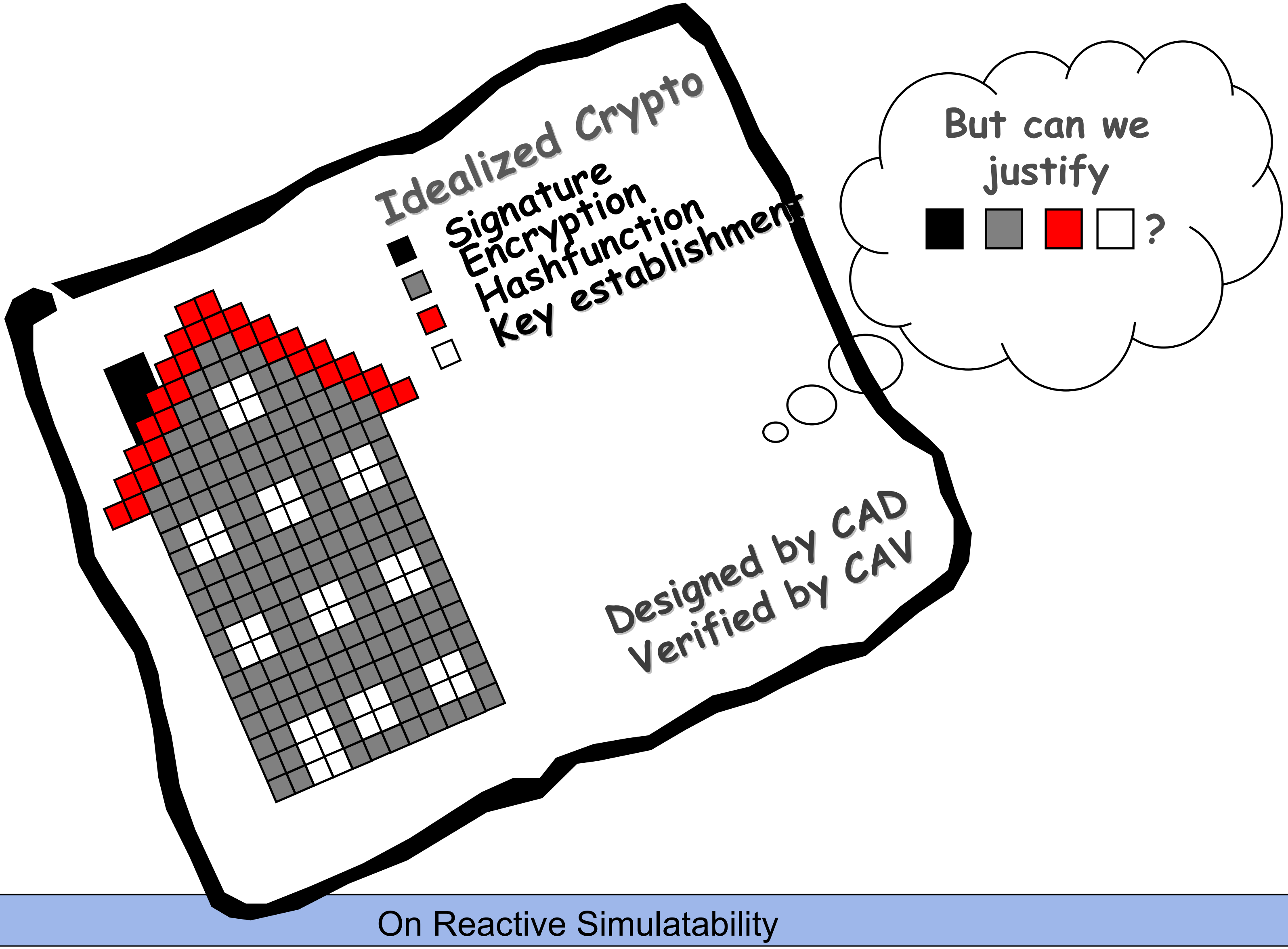
DL(g^x)

Prob[Attack] ≤ ...

Cryptography: The Details



Formal Methods: The Big Picture



Idealized Crypto

■ Signature

▨ Encryption

🟥 Hashfunction

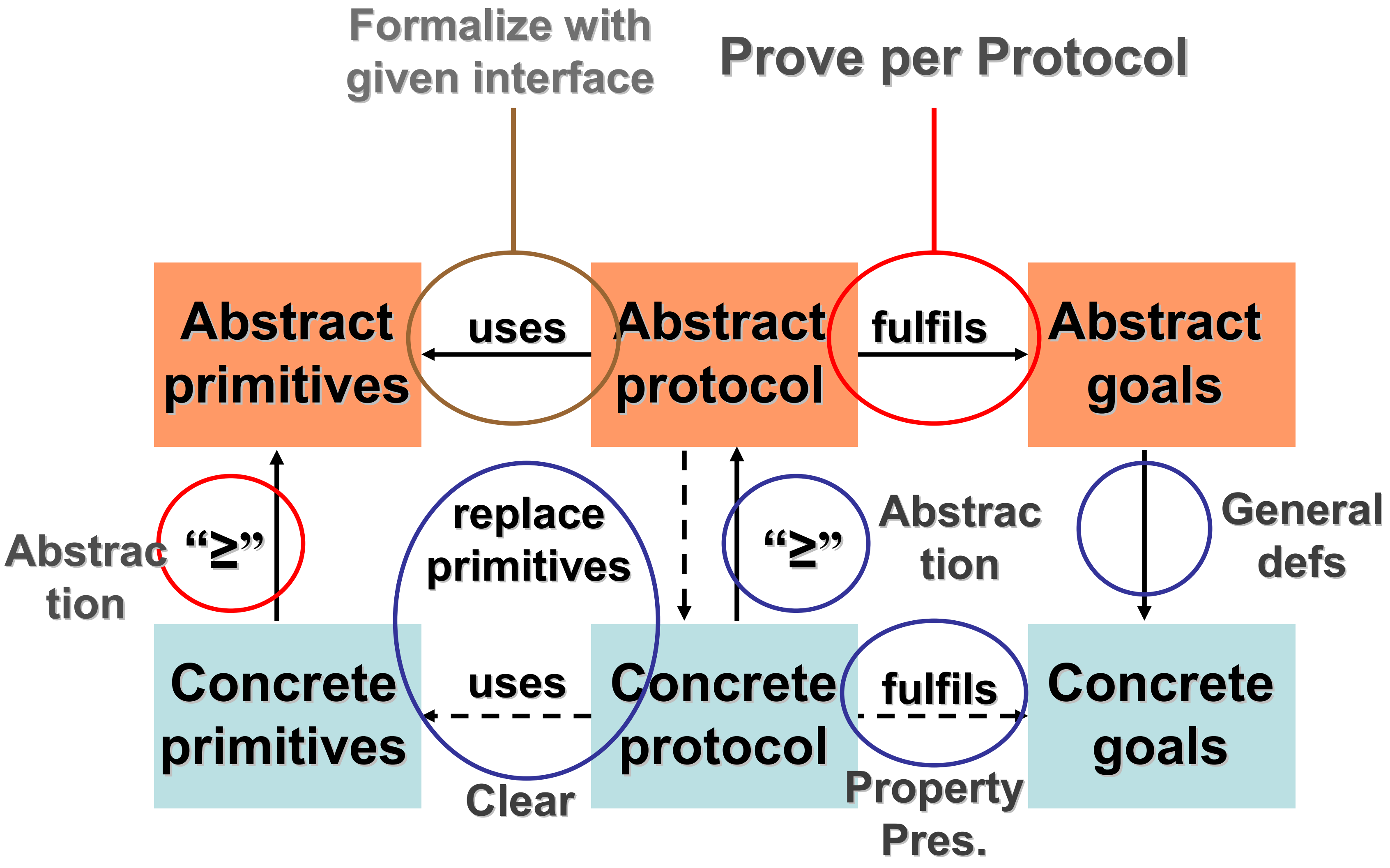
□ Key establishment

Designed by CAD

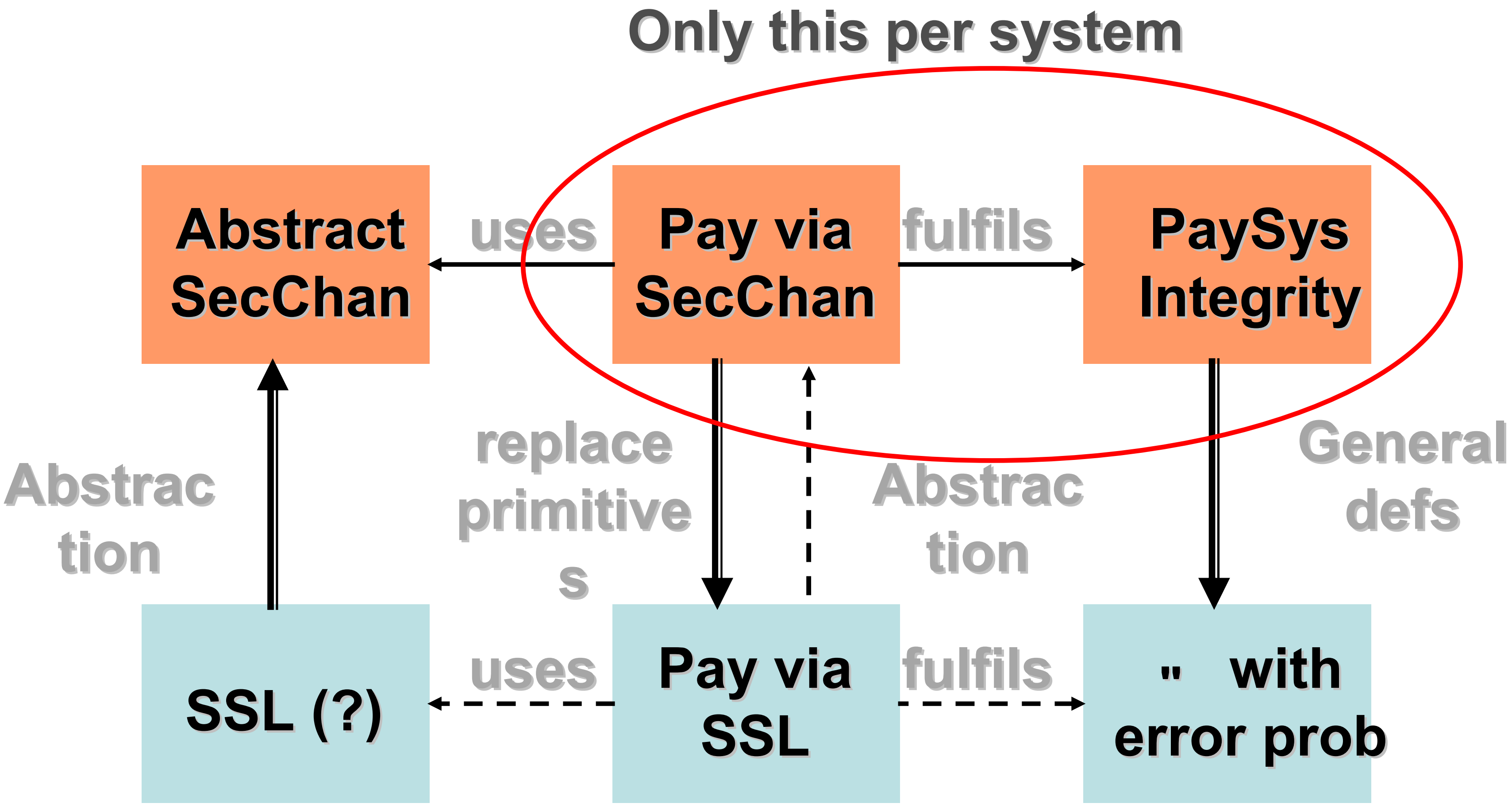
Verified by CAV

But can we justify ■ ▨ 🟥 □ ?

Idea: Sound Abstract Protocol Proofs



Example



Only this per system

Abstract SecChan

uses

Pay via SecChan

fulfils

PaySys Integrity

Abstrac tion

replace primitive s

Abstrac tion

General defs

SSL (?)

uses

Pay via SSL

fulfils

" with error prob

What do we need for soundly abstracting?

- Precise system model that permits all “realistic” attacks.
  - Network characteristics? synchr./asynchr., reliable, secure, etc.
  - Power of the adversary? Passive/active, static/dynamic, secure function evaluation / reactive (!)
  - Realistic scheduling
  - Which other protocols may run concurrently?
  - …

What do we need for soundly abstracting?

- Precise system model that permits all “realistic” attacks.
- Capable of reasoning about abstractions/realizations at the same time
  - Cryptographic issues: probabilism, error-probabilities, computationsl restrictions, etc.
  - Abstraction issues: Abstract transition functions, distributed-systems aspects, formal calculi, etc.

What do we need for soundly abstracting?

- Precise system model that permits all “realistic” attacks.
- Capable of reasoning about abstractions/realizations at the same time
- Mathematically rigorous definition of what a “good” abstraction is
  - Intuitive
  - Should fit to a variety of different abstractions/real protocol classes
  - Provable by convenient proof techniques

What do we need for soundly abstracting?

- Precise system model that permits all “realistic” attacks.
- Capable of reasoning about abstractions/realizations at the same time
- Mathematically rigorous definition of what a “good” abstraction is
- Not only hold in isolation but preserve security under composition
  - (Makes the definition “useful”)
  - Make modular analysis of larger protocols possible

What do we need for soundly abstracting?

- Precise system model that permits all “realistic” attacks.
- Capable of reasoning about abstractions/realizations at the same time
- Mathematically rigorous definition of what a “good” abstraction is
- Not only hold in isolation but preserve security under composition
- Should preserve essentially arbitrary security properties
  - Integrity, variants of confidentiality, non-interference, poly-time liveness
  - Tight links to properties shown for symbolic abstractions of crypto

What do we need for soundly abstracting?

- Precise system model that permits all “realistic” attacks.
- Capable of reasoning about abstractions/realizations at the same time
- Mathematically rigorous definition of what a “good” abstraction is
- Not only hold in isolation but preserve security under composition
- Should preserve essentially arbitrary security properties
- Abstractions should match the intuition for the requirements in mind
  - Intuitive abstractions, easy to read for non-specialist, thus enabling convenient use in larger protocols

What do we need for soundly abstracting?

- Precise system model that permits all “realistic” attacks.
- Capable of reasoning about abstractions/realizations at the same time
- Mathematically rigorous definition of what a “good” abstraction is
- Not only hold in isolation but preserve security under composition
- Should preserve essentially arbitrary security properties
- Abstractions should match the intuition for the requirements in mind
- Abstractions should be based on the functionality of the protocol, not on its structure.
- Good abstractions for many of useful protocol classes should exist

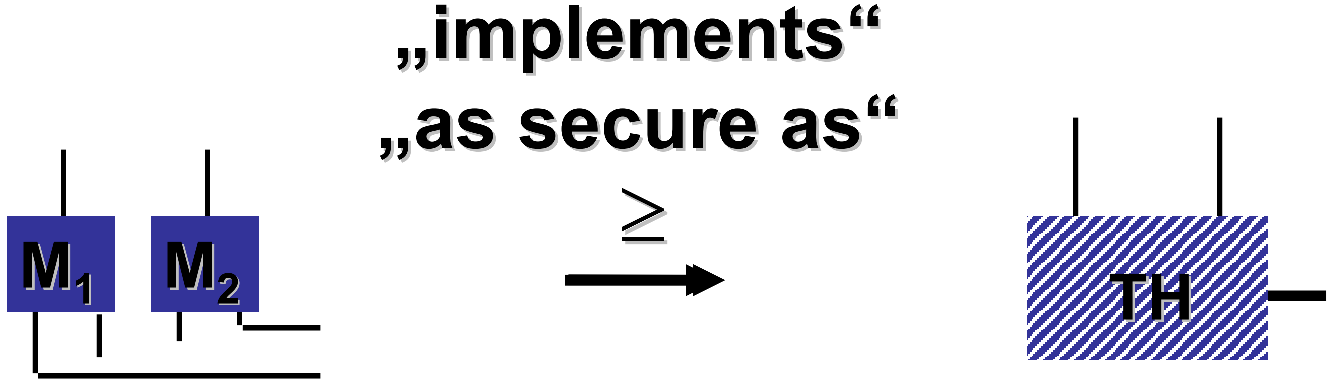
Overview: Reactive Simulatability Framework

- Precise system model allowing cryptographic and abstract operations
- Reactive simulatability with composition theorem
- Preservation theorems for security properties
- Concrete pairs of idealizations and secure realizations
- Sound symbolic abstractions (Dolev-Yao models) that are suitable for tool support
- Sound security proofs of security protocols: NSL, Otway-Rees, iKP, (parts of) Kerberos, etc.
- Detailed Proofs (Cryptographic bisimulations with static information flow analysis, …)

What do we need for soundly abstracting?

- Precise system model that permits all “realistic” attacks.
- Capable of reasoning about abstractions/realizations at the same time
- Mathematically rigorous definition of what a “good” abstraction is
- Not only hold in isolation but preserve security under composition
- Should preserve essentially arbitrary security properties
- Abstractions should match the intuition for the requirements in mind
- Abstractions should be based on the functionality of the protocol, not on its structure.
- Good abstractions for many of useful protocol classes should exist

Idea: Define Security relative to an ideal task



„implements“

„as secure as“

$\geq$

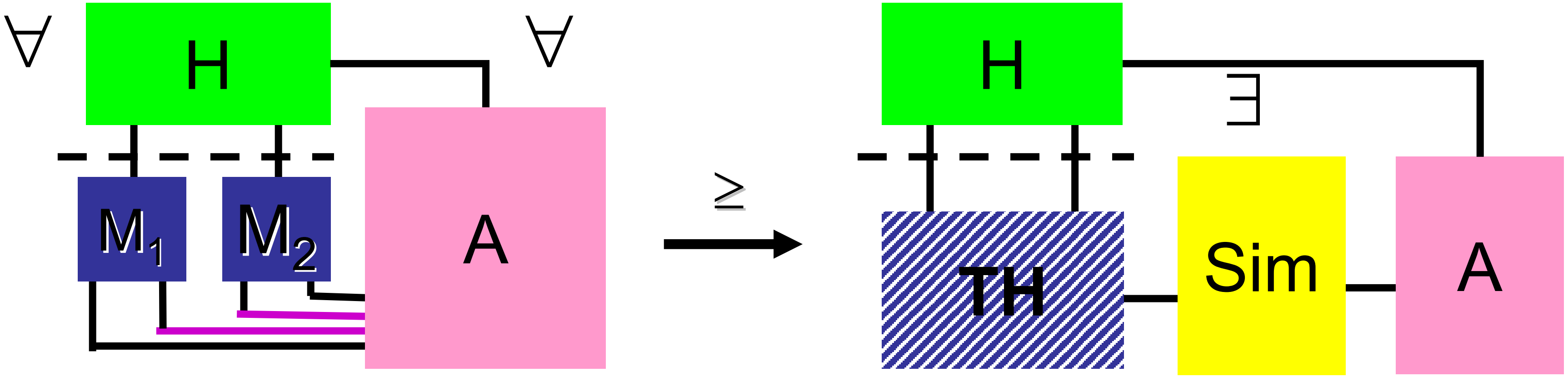
Real system

Ideal system (abstraction)

How to define that? What does “every attack” mean? “successfully converted”?

What are good ideal systems? What about concrete security properties, e.g., integrity or secrecy?

Reactive Simulatability – here blackbox



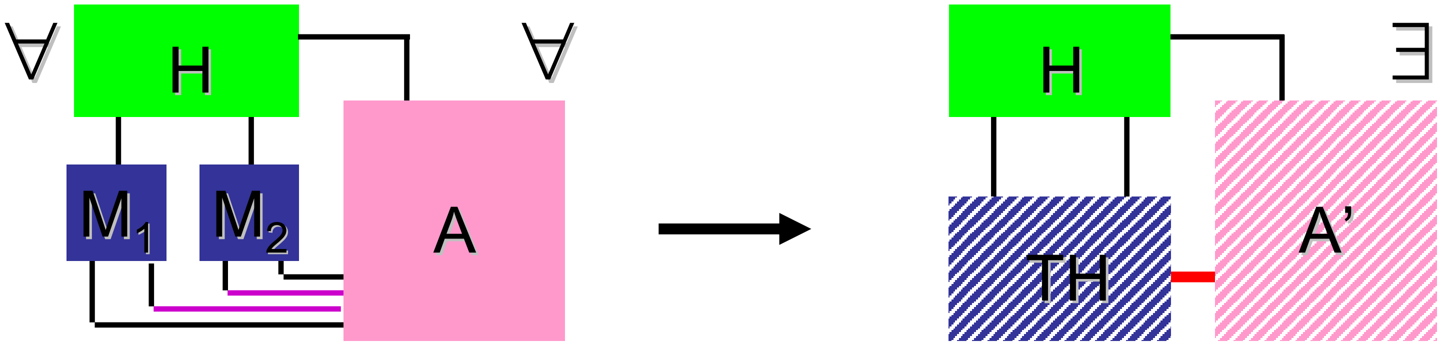
Real system

Ideal system

$$\text{view}_{\text{real}}(H) \approx \text{view}_{\text{ideal}}(H)$$

Indistinguishability of random variables

Reactive Simulatability Variants



- Standard simulatability: ∀A ∀H ∃A'
- Universal simulatability: ∀A ∃A ∀H
- Blackbox simulatability: ∃Sim ∀H ∀A A'=Sim&A
- Perfect / statistic / computational

Indistinguishability [Yao_82]

- Families of random variables:

$$(v_k)_{k\in \mathrm{IN}} \quad \approx_{\mathrm{poly}} \quad (v'_k)_{k\in \mathrm{IN}}$$

$\approx_{\mathrm{poly}} \Leftrightarrow \forall\, D$ (prob. poly. in first input):

$$|\ \Pr(D(1^k, v_k) = 1) - \Pr(D(1^k, v'_k) = 1)\ |$$

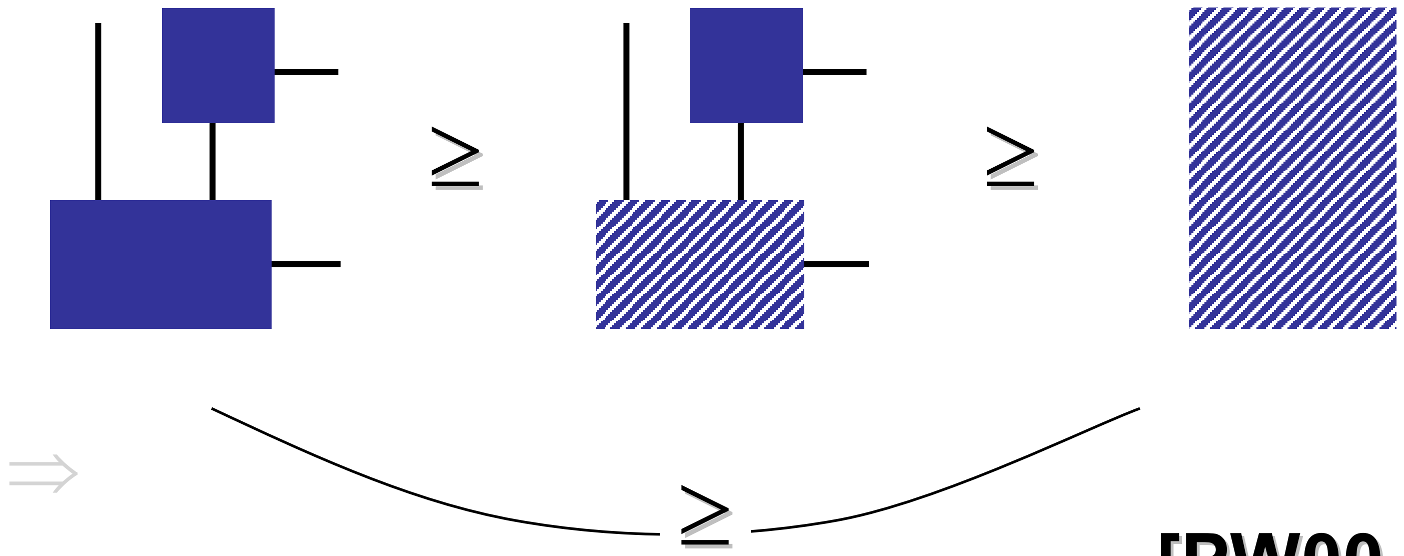
$$\leq 1\ /\ \mathrm{poly}(k).$$

What do we need for soundly abstracting?

- Precise system model that permits all “realistic” attacks.
- Capable of reasoning about abstractions/realizations at the same time
- Mathematically rigorous definition of what a “good” abstraction is
- Not only hold in isolation but preserve security under composition
- Should preserve essentially arbitrary security properties
- Abstractions should match the intuition for the requirements in mind
- Abstractions should be based on the functionality of the protocol, not on its structure.
- Good abstractions for many of useful protocol classes should exist

Base Lemmas of reactive simulatability

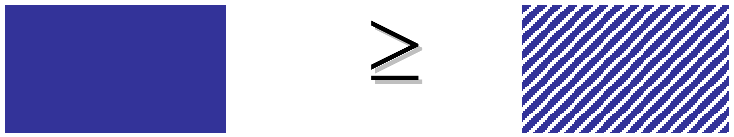
- Machine combination is defined and
  - is associative
  - retains poly-time (for strong version)
  - retains sub-machine views
- “As secure as” is transitive. E.g., with composition:



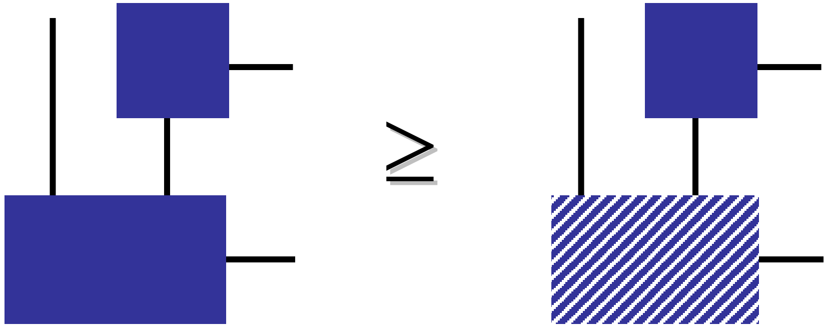
[PW00,PW01]

Composition – One System

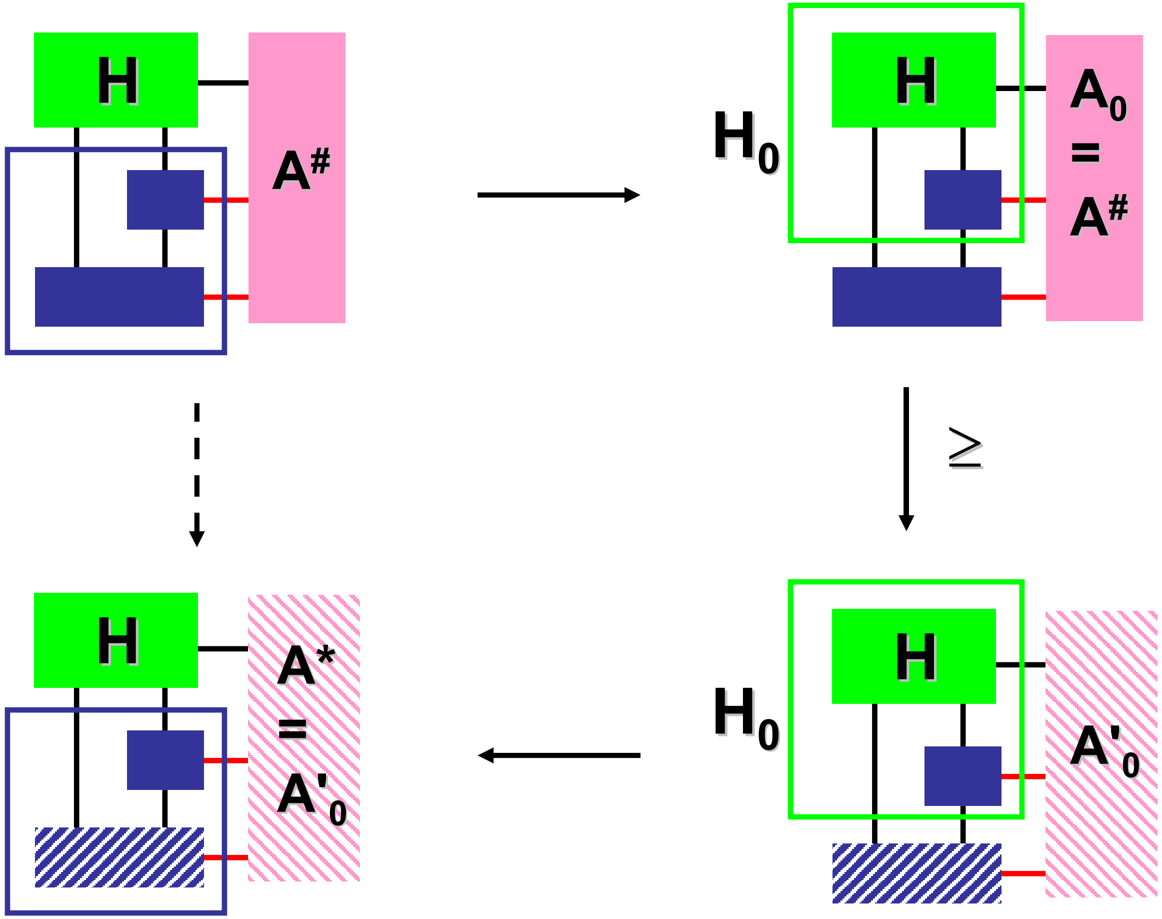
Given:



Then this holds:

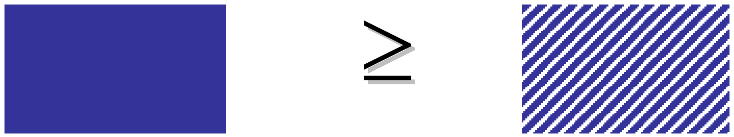


Proof Idea (Single Composition)

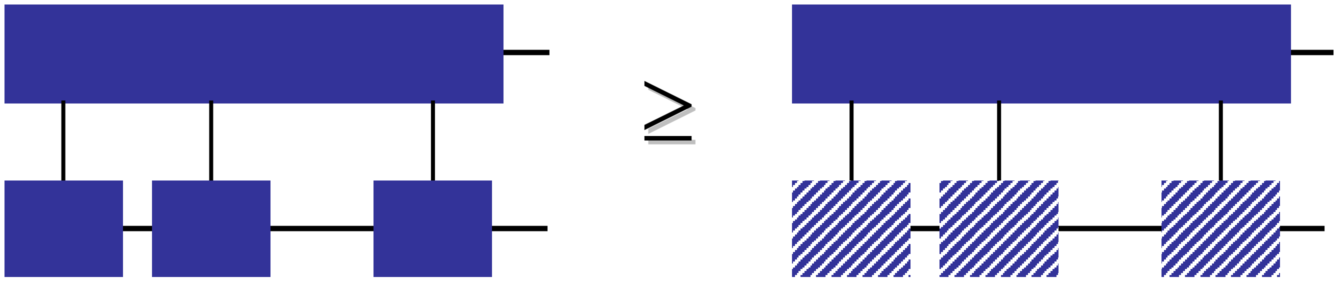


Composition – Multiple Systems

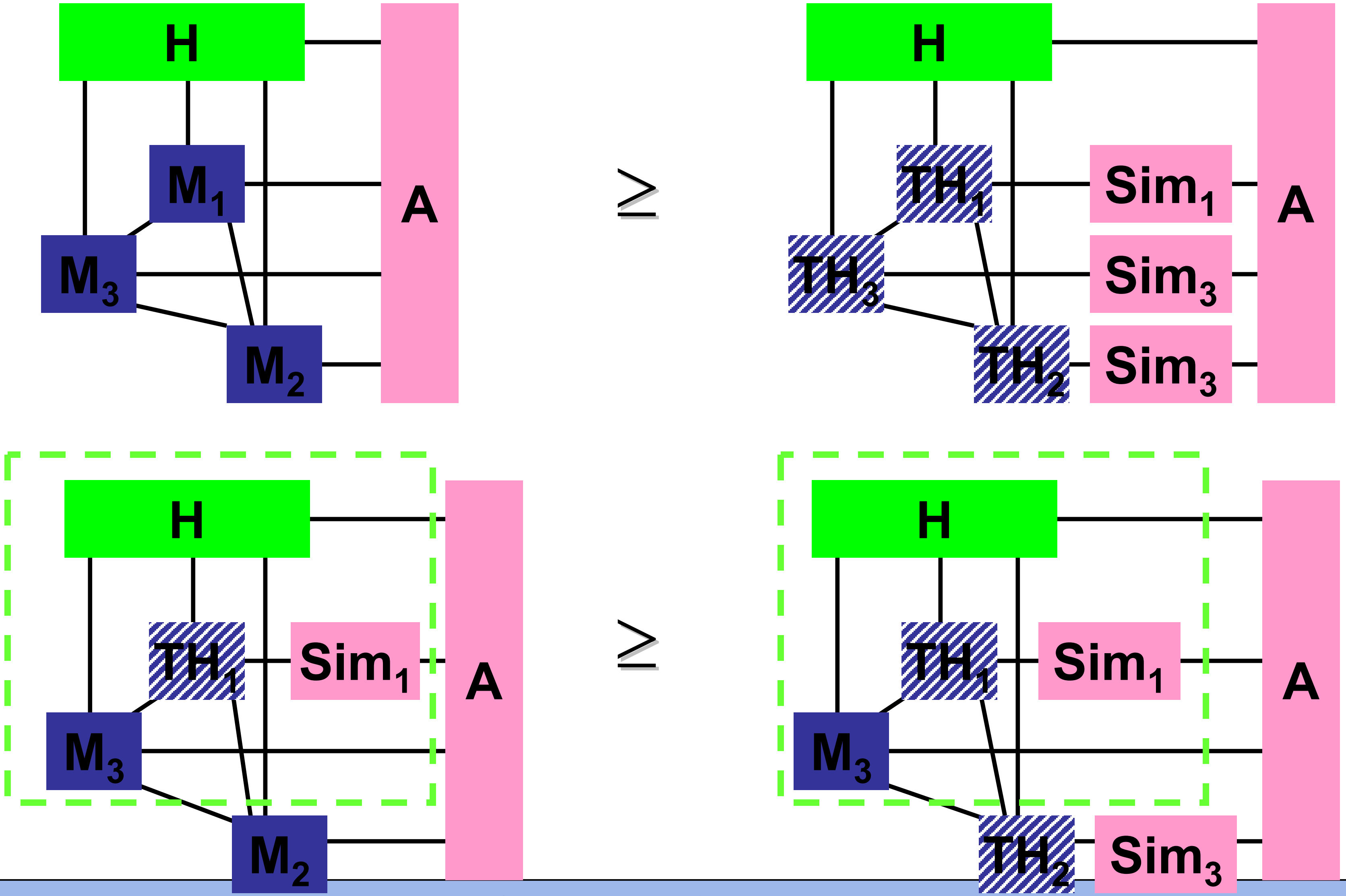
Given:



Also this holds:



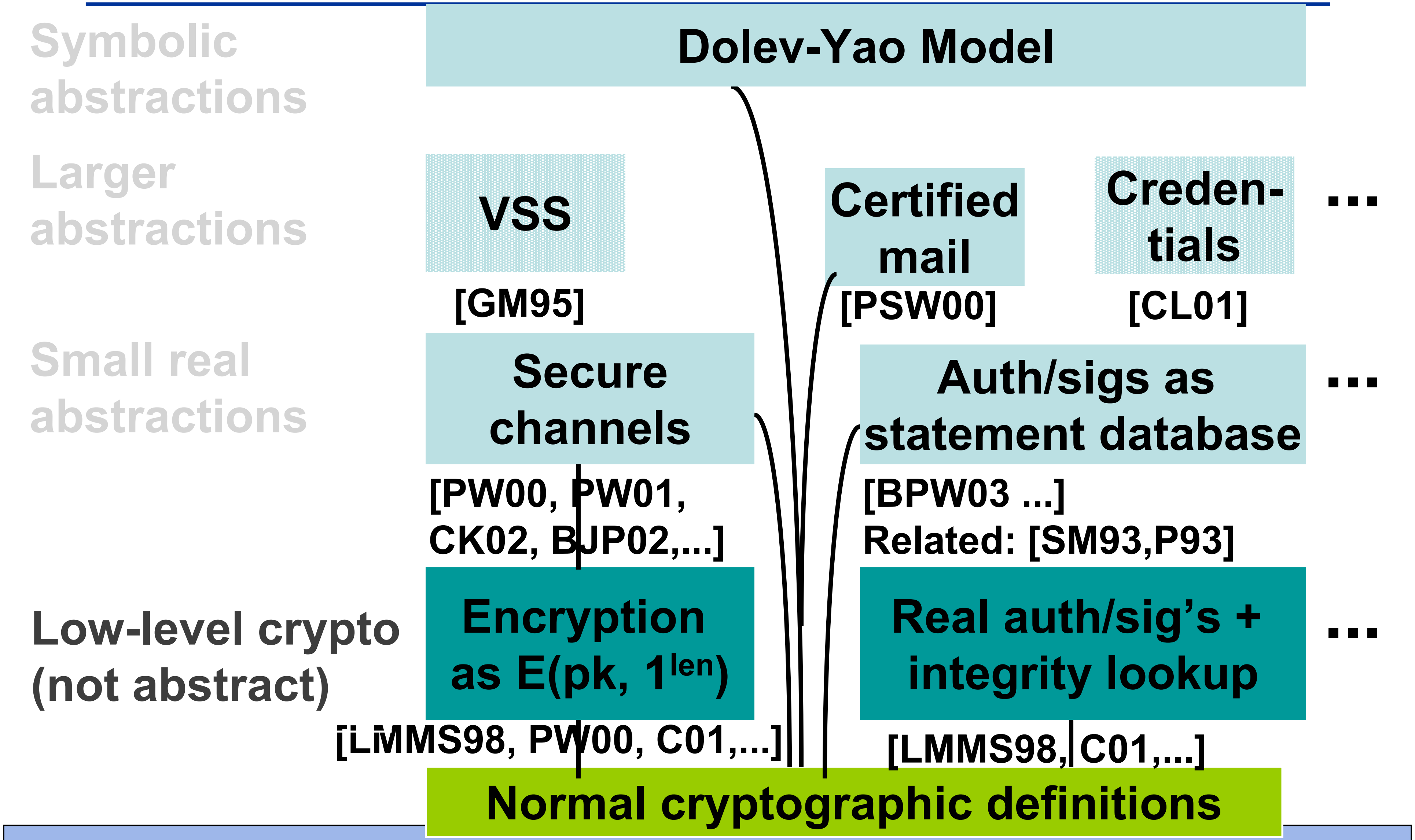
General Composition Proof via Hybrid Systems



What do we need for soundly abstracting?

- Precise system model that permits all “realistic” attacks.
- Capable of reasoning about abstractions/realizations at the same time
- Mathematically rigorous definition of what a “good” abstraction is
- Not only hold in isolation but preserve security under composition
- Should preserve essentially arbitrary security properties
- Abstractions should match the intuition for the requirements in mind
- Abstractions should be based on the functionality of the protocol, not on its structure.
- Good abstractions for many of useful protocol classes should exist

Cryptographic Idealization Layers



Symbolic abstractions

Dolev-Yao Model

Larger abstractions

VSS [GM95]

Certified mail [PSW00]

Creden-tials [CL01]

…

Small real abstractions

Secure channels [PW00, PW01, CK02, BJP02,…]

Auth/sigs as statement database [BPW03 …] Related: [SM93,P93]

…

Low-level crypto (not abstract)

Encryption as E(pk, 1^{len}) [LMMS98, PW00, C01,…]

Real auth/sig's + integrity lookup [LMMS98, C01,…]

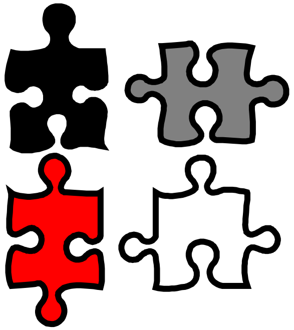
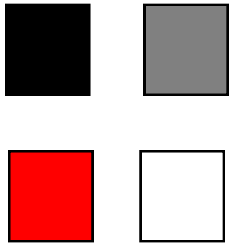
…

Normal cryptographic definitions

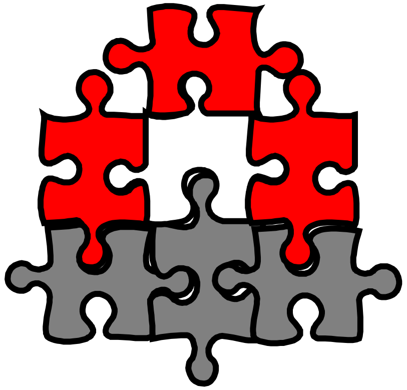
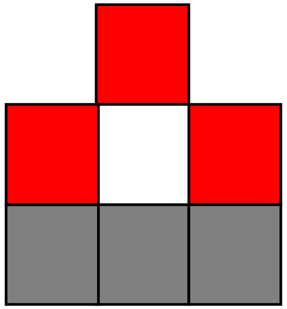
What do we need for soundly abstracting?

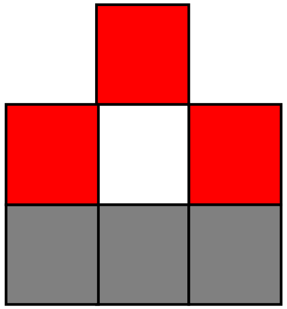
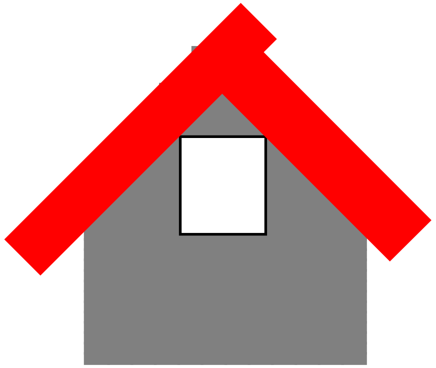
- Precise system model that permits all “realistic” attacks.
- Capable of reasoning about abstractions/realizations at the same time
- Mathematically rigorous definition of what a “good” abstraction is
- Not only hold in isolation but preserve security under composition
- Should preserve essentially arbitrary security properties
- Abstractions should match the intuition for the requirements in mind
- Abstractions should be based on the functionality of the protocol, not on its structure.
- Good abstractions for many of useful protocol classes should exist

Recall Prior Result

- “as secure as” (reactive simulatability)
- for certain versions of  and 

Specification Styles

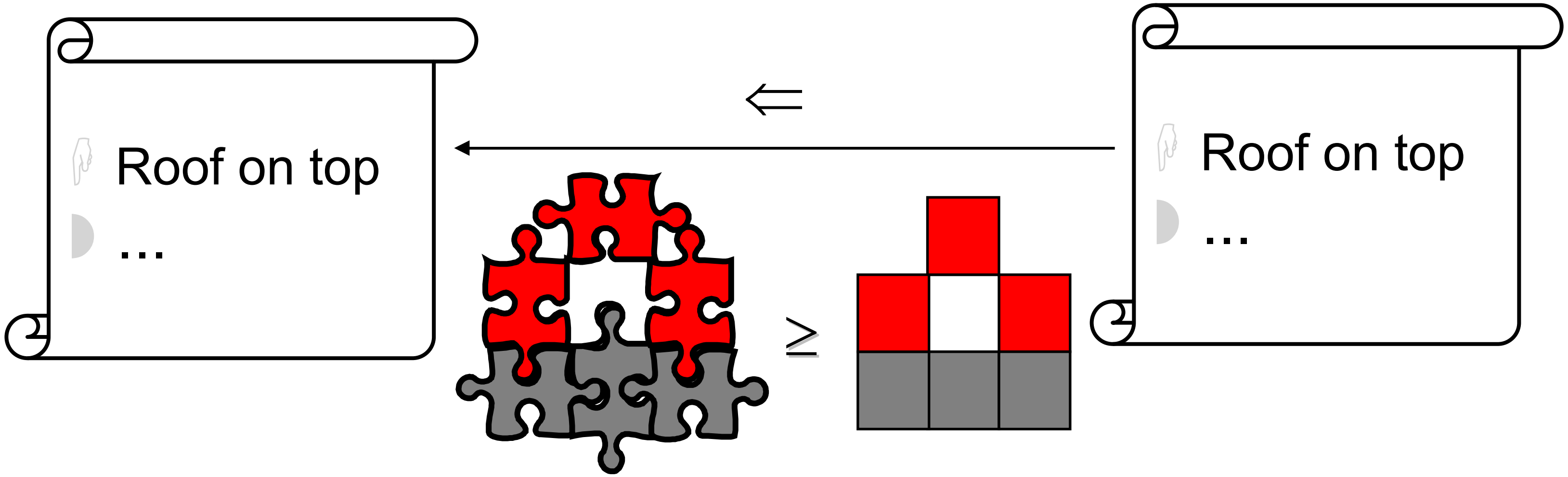
- Is  ≥  what people want?
- Often yes, in particular together with

 ≥ 

  – E.g., secure channels (see also spi calculus), certified mail
- But not always ...

Alternative: Property-based spec.

- E.g., “I want a tight roof on top”: integrity
  - Preserved by “≥”:



Roof on top

...

⇐

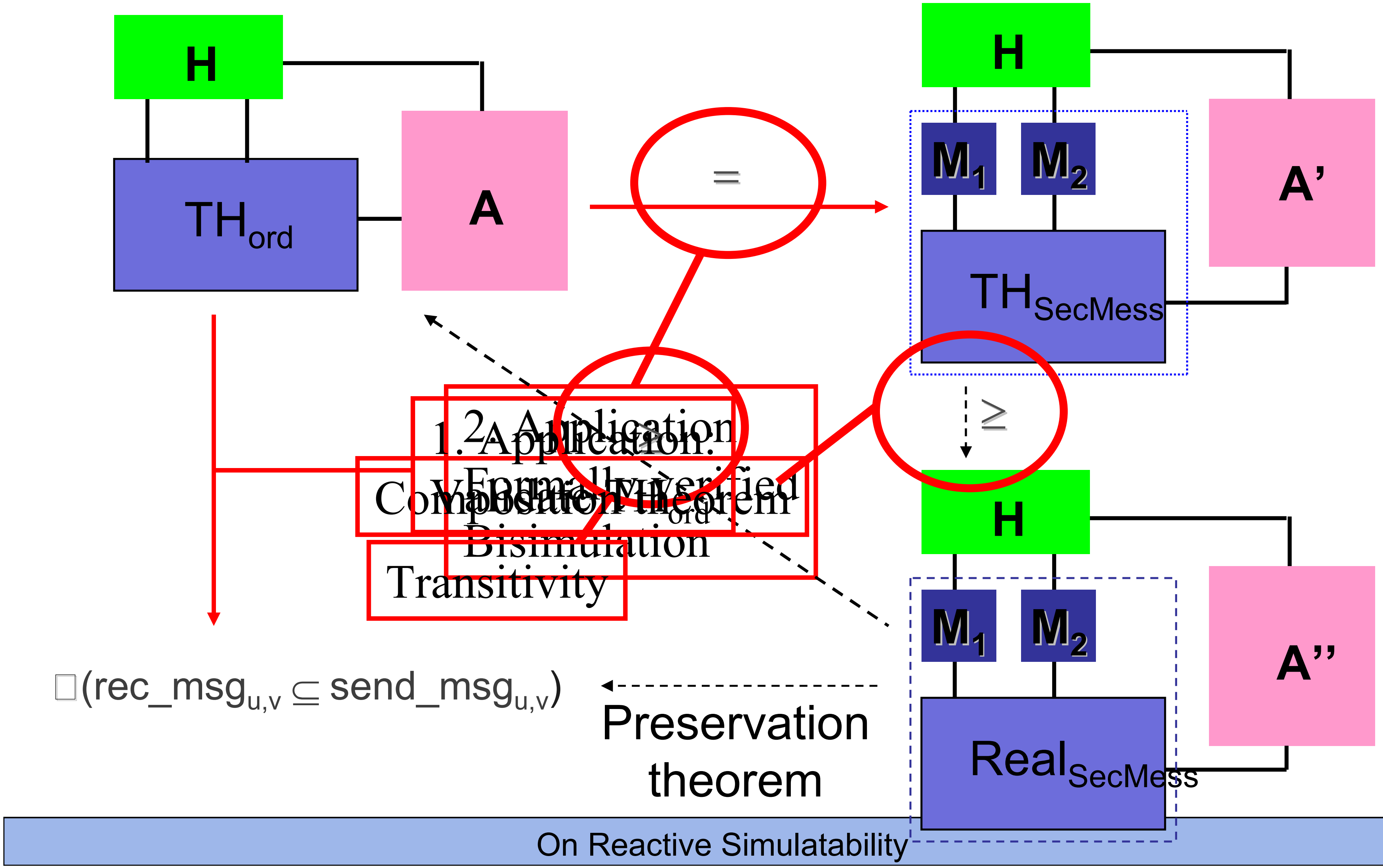
≥

Roof on top

...

- In the RSIM framework: Preservation theorems for integrity, non-interference, poly-time liveness, etc.

Example: Ordered Channels over Unordered Ones



1. Application: Validation of TH_{ord}

2. Application: Formally verified Bisimulation

Composition theorem

Transitivity

$\Box(\text{rec\_msg}_{u,v} \subseteq \text{send\_msg}_{u,v})$

Preservation theorem

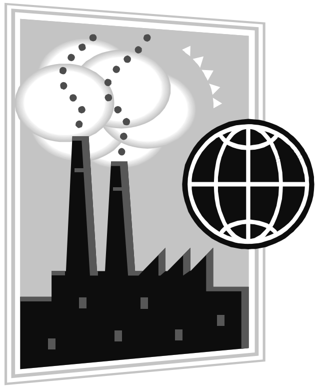
What do we need for soundly abstracting?

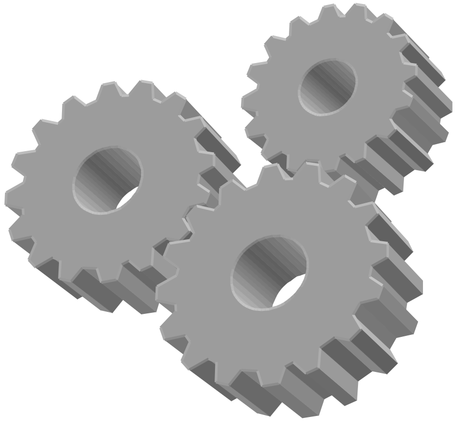
- Precise system model that permits all “realistic” attacks.
- Capable of reasoning about abstractions/realizations at the same time
- Mathematically rigorous definition of what a “good” abstraction is
- Not only hold in isolation but preserve security under composition
- Should preserve essentially arbitrary security properties
- Abstractions should match the intuition for the requirements in mind
- Abstractions should be based on the functionality of the protocol, not on its structure.
- Good abstractions for many of useful protocol classes should exist

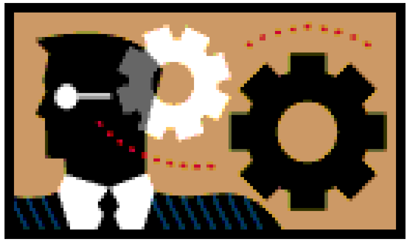
Now: the BPW model (sound Dolev-Yao style library)

Why Formal Methods?

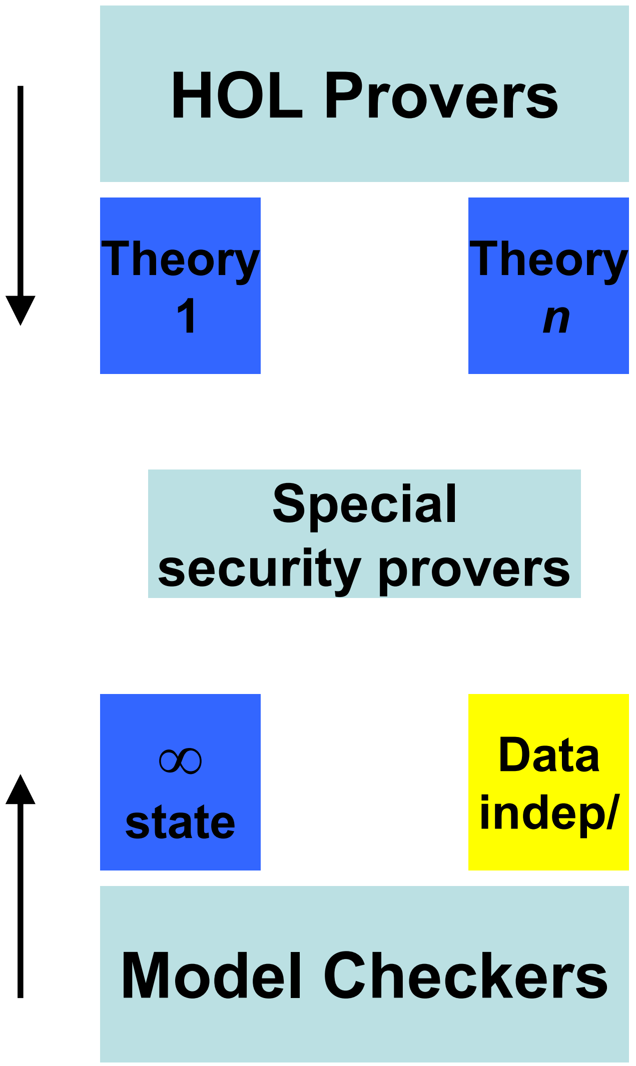
- Automation if
  - Repetitive
  - Tedious
  - Prone to human errors
  - Critical application
- A top candidate: Distributed protocols
- Security variants for 20 years







Protocol Proof Tools



- Almost anything
- Much human interaction

- Special logic fragments for security
- Approximations: correct, not complete

- Fully automatic
- State exploration

Automating Security Protocol Proofs

- Even simple protocol classes & properties undecidable
  - Robust protocol design helps
- Full arithmetic is out
- Probability theory just developing for reasoning about larger protocols

So how do current tools handle cryptography?

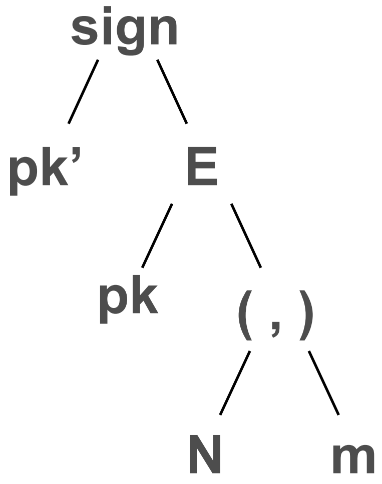
Dolev-Yao Model

- Idea [DY81]
  - Abstraction as term algebras, e.g., $D_x(E_x(E_x(m)))$
  - Cancellation rules, e.g., $D_xE_x = \varepsilon$
- Well-developed proof theories
  - Abstract data types
  - Equational 1st-order logic
- Important for security proofs:
  - Inequalities! (Everything that cannot be derived.)
  - Known as “initial model”

Important goal: Justify or replace

Dolev-Yao Model – Variants [BPW]

- Operators and equations **[EG82, M83, EGS85 ...]**
  - Enc, sigs, nonce, payload, pairing, ...
  - Inequalities assumed across operators!
- Untyped or typed
- Destructors explicit or implicit
- Abstraction from probabilism
  - Finite selection, counting, multisets
- Surrounding protocol language
  - Special-purpose, CSP, pi calculus, ... [any]



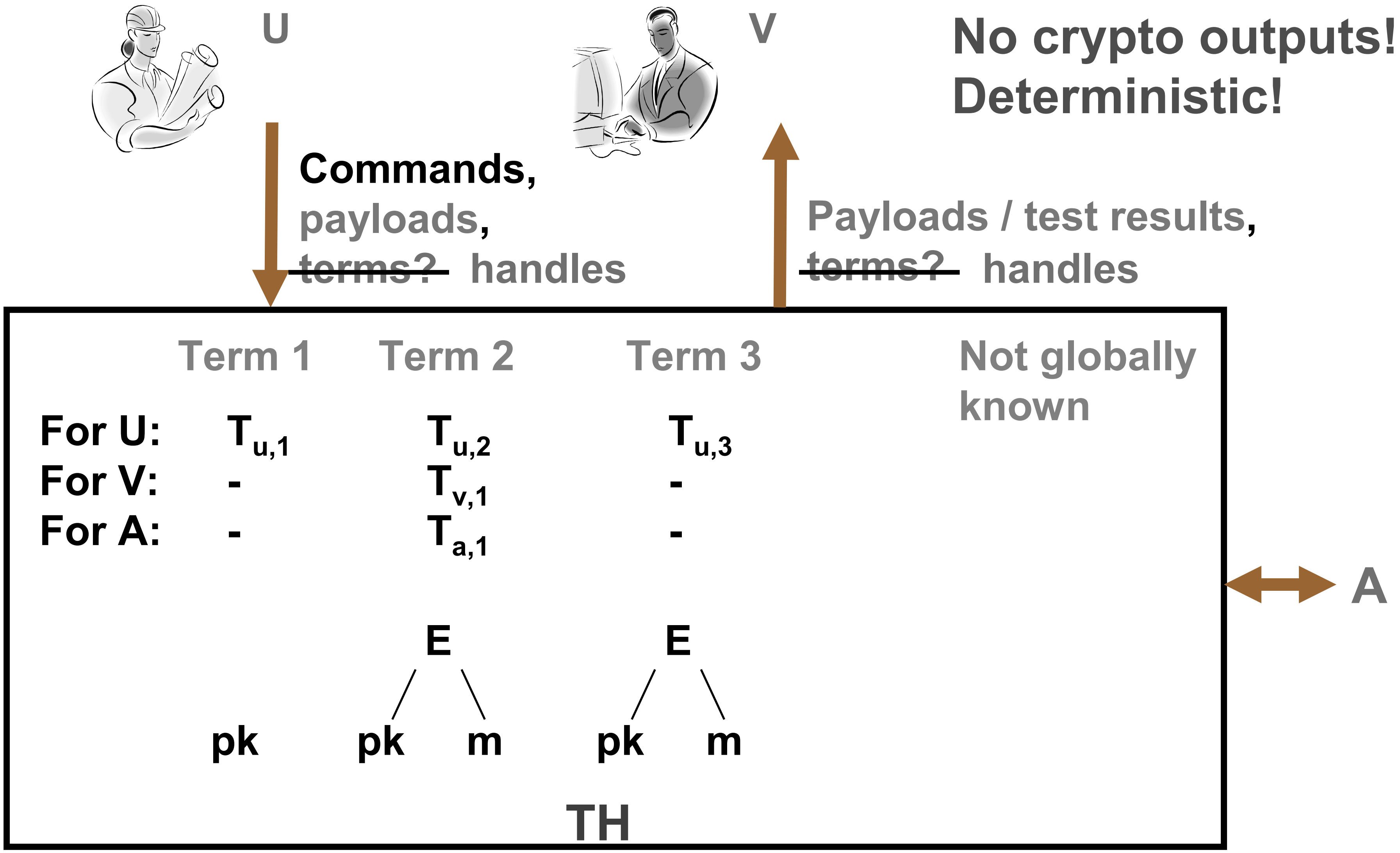
The BPW model – major challenges

- Recall: Term algebra, inequalities
- Major tasks:
  - Represent ideal and real library in the same way to higher protocols
  - Prevent honest users from stupidity with real crypto objects, but don’t restrict adversary
    - E.g., sending a bitstring that’s almost a signature
  - What imperfections are tolerable / must be allowed?

The BPW model - characteristics

- Characteristics
  - "library" of standard crypto primitives
  - tracks content (messages) and knowledge (who knows what)
  - *N* honest users and adversary manipulate messages indirectly using handles
- Functionality
  - local functions for message construction and access
    (e.g. nonce & key generation, encryption/decryption, sign/verify, pairing/projection)
  - send functions for message transmission
    (user to adversary and vice versa)
  - adversary interface has additional capabilities
    (e.g. create garbage messages, invalid ciphertexts transform signatures)

The BPW model



No crypto outputs!
Deterministic!

U

Commands,
payloads,
~~terms?~~ handles

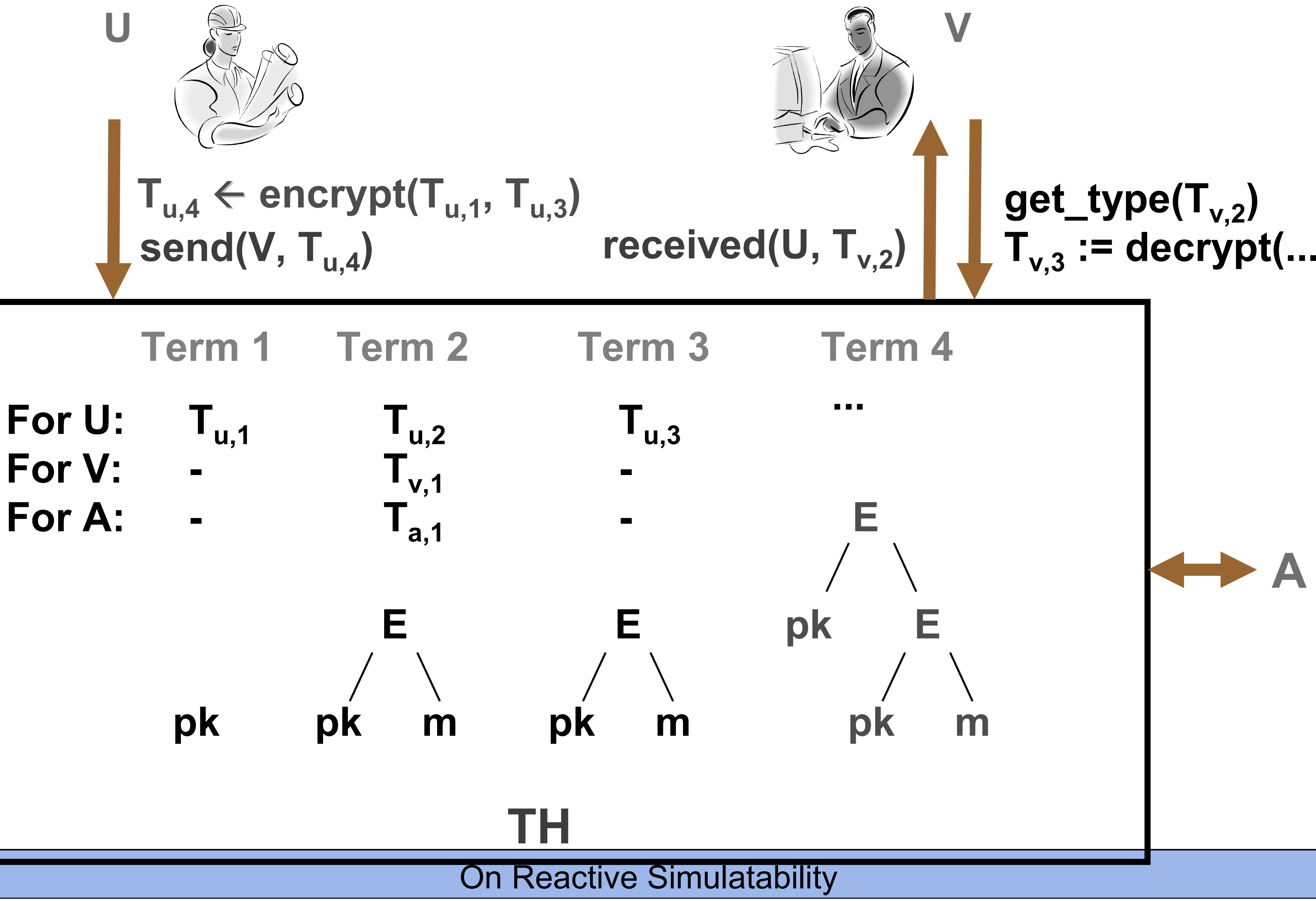
V

Payloads / test results,
~~terms?~~ handles

	Term 1	Term 2	Term 3
For U:	$T_{u,1}$	$T_{u,2}$	$T_{u,3}$
For V:	-	$T_{v,1}$	-
For A:	-	$T_{a,1}$	-

Not globally known

The BPW model



U

$T_{u,4} \leftarrow$ encrypt($T_{u,1}$, $T_{u,3}$)
send(V, $T_{u,4}$)

V

received(U, $T_{v,2}$)

get_type($T_{v,2}$)
$T_{v,3}$:= decrypt(...)

	Term 1	Term 2	Term 3	Term 4
For U:	$T_{u,1}$	$T_{u,2}$	$T_{u,3}$	...
For V:	-	$T_{v,1}$	-	
For A:	-	$T_{a,1}$	-	

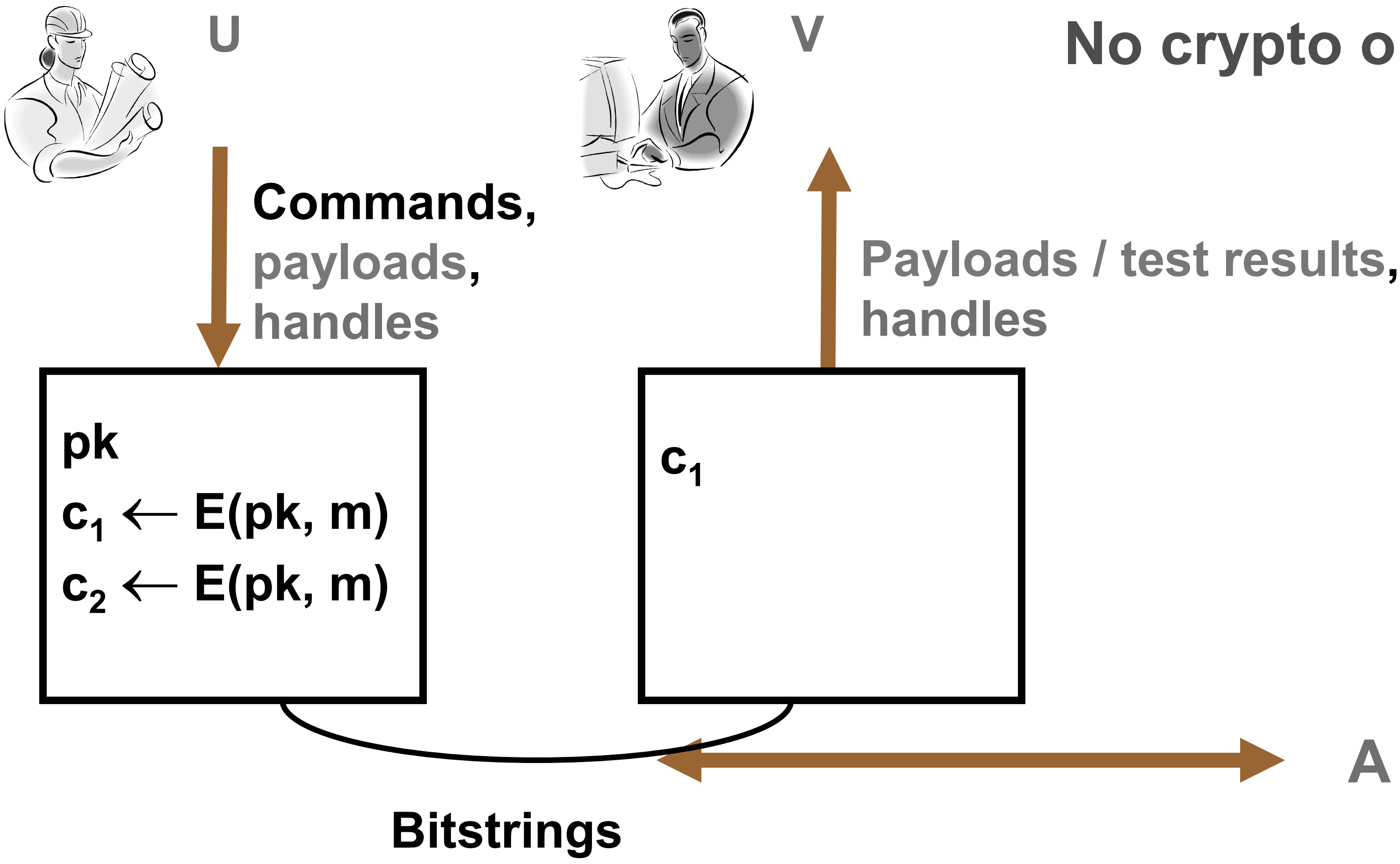
Main Differences to Standard Dolev-Yao

- Tolerable imperfections:
  - – Lengths of encrypted messages cannot be kept secret
  - – Adversary may include incorrect messages inside encryptions
  - – Signature schemes can have memory
  - – Slightly restricted key usage for symmetric encryption

Most imperfections avoidable
for more restricted cases

Real Implementation

No crypto outputs!



U

V

Commands, payloads, handles

Payloads / test results, handles

pk

$c_1 \leftarrow E(pk, m)$

$c_2 \leftarrow E(pk, m)$

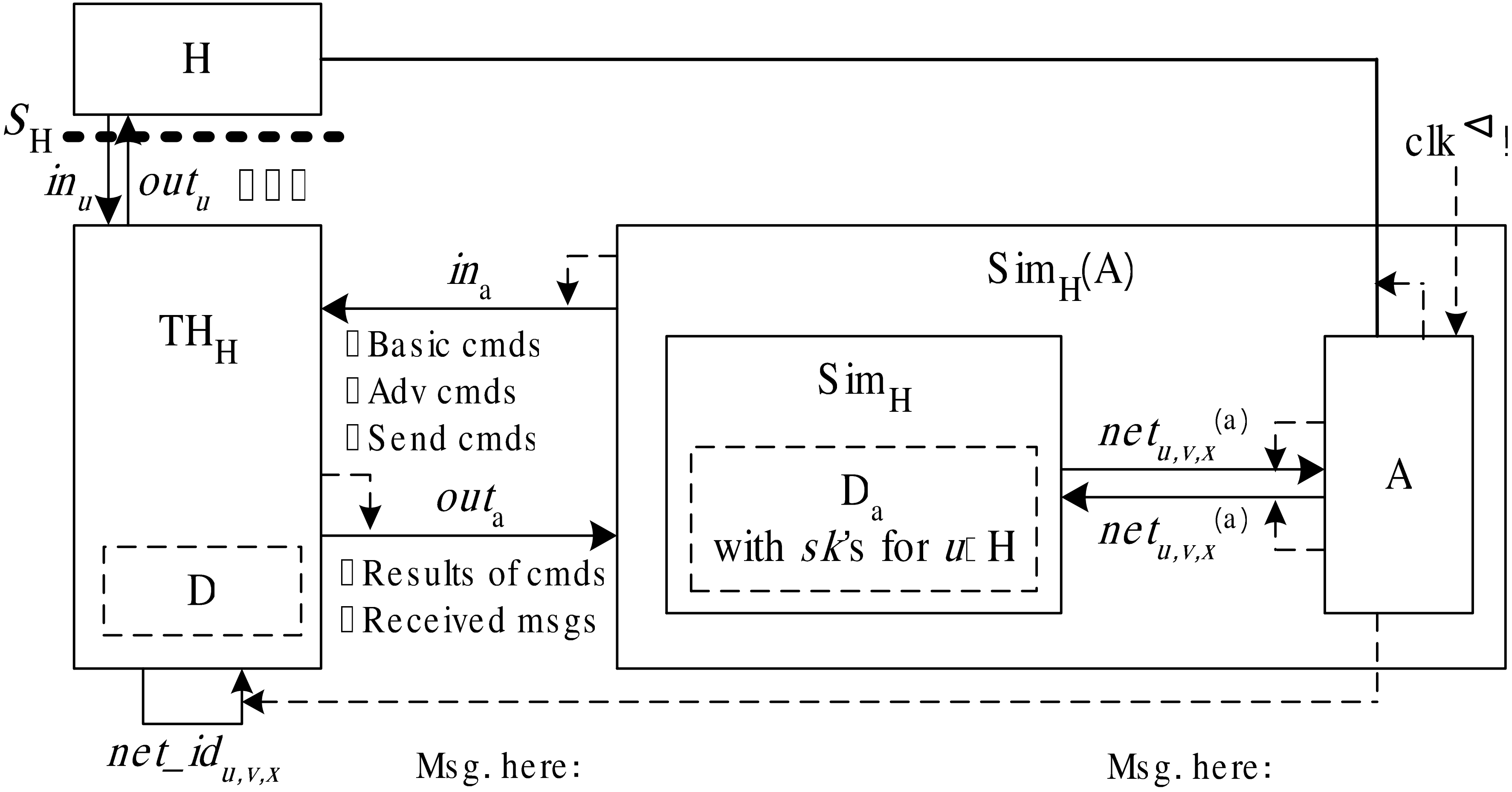
c_1

A

Bitstrings

Real system

The Simulator



H

S_H

in_u

out_u □□□

clk ⊲ !

TH_H

D

in_a

□Basic cmds

□Adv cmds

□Send cmds

out_a

□Results of cmds

□Received msgs

$Sim_H(A)$

Sim_H

D_a
with sk's for u□ H

$net_{u,v,x}^{(a)}$

$net_{u,v,x}^{(a)}$

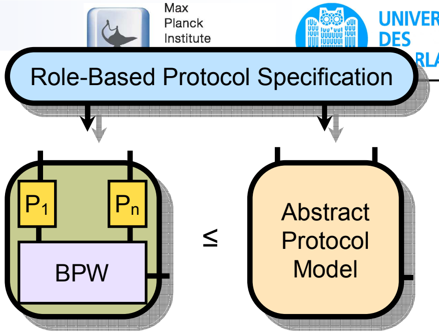
A

$net\_id_{u,v,x}$

Msg. here: index l^{nd}

Msg. here:
(u, v, x, l^{nd})

Msg. here:
word l



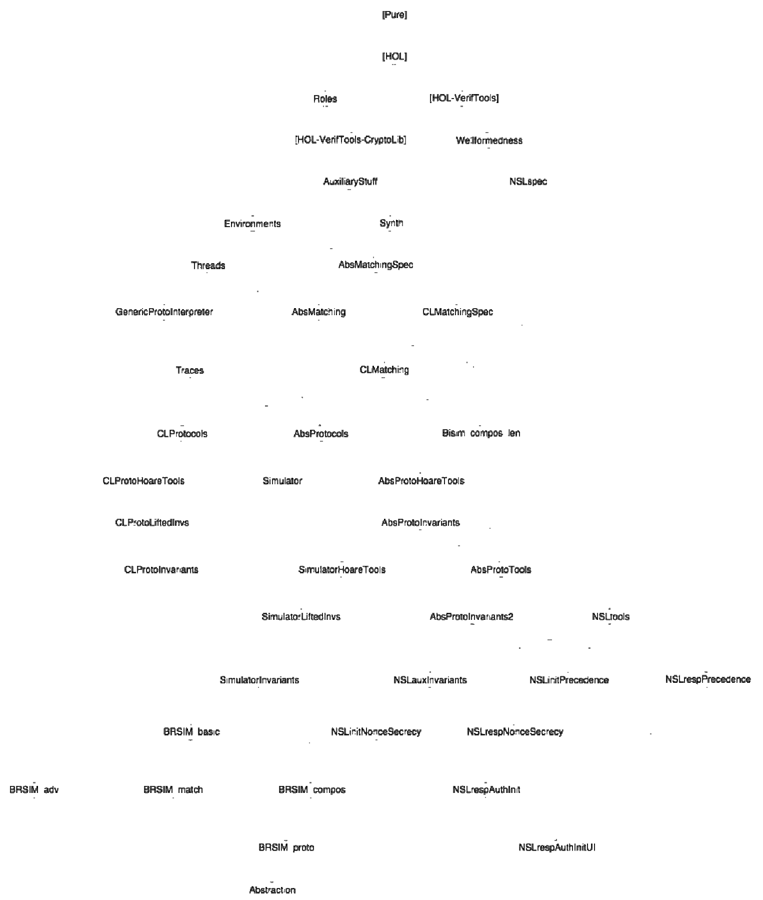
The BPW model in Isabelle

APM/BRSIM proof & NSL case study

- 40 theories (boxes in graph)
- 150 definitions
- 1200 lemmas and theorems
- 18k lines of Isabelle/HOL
- 360 pages PDF documentation

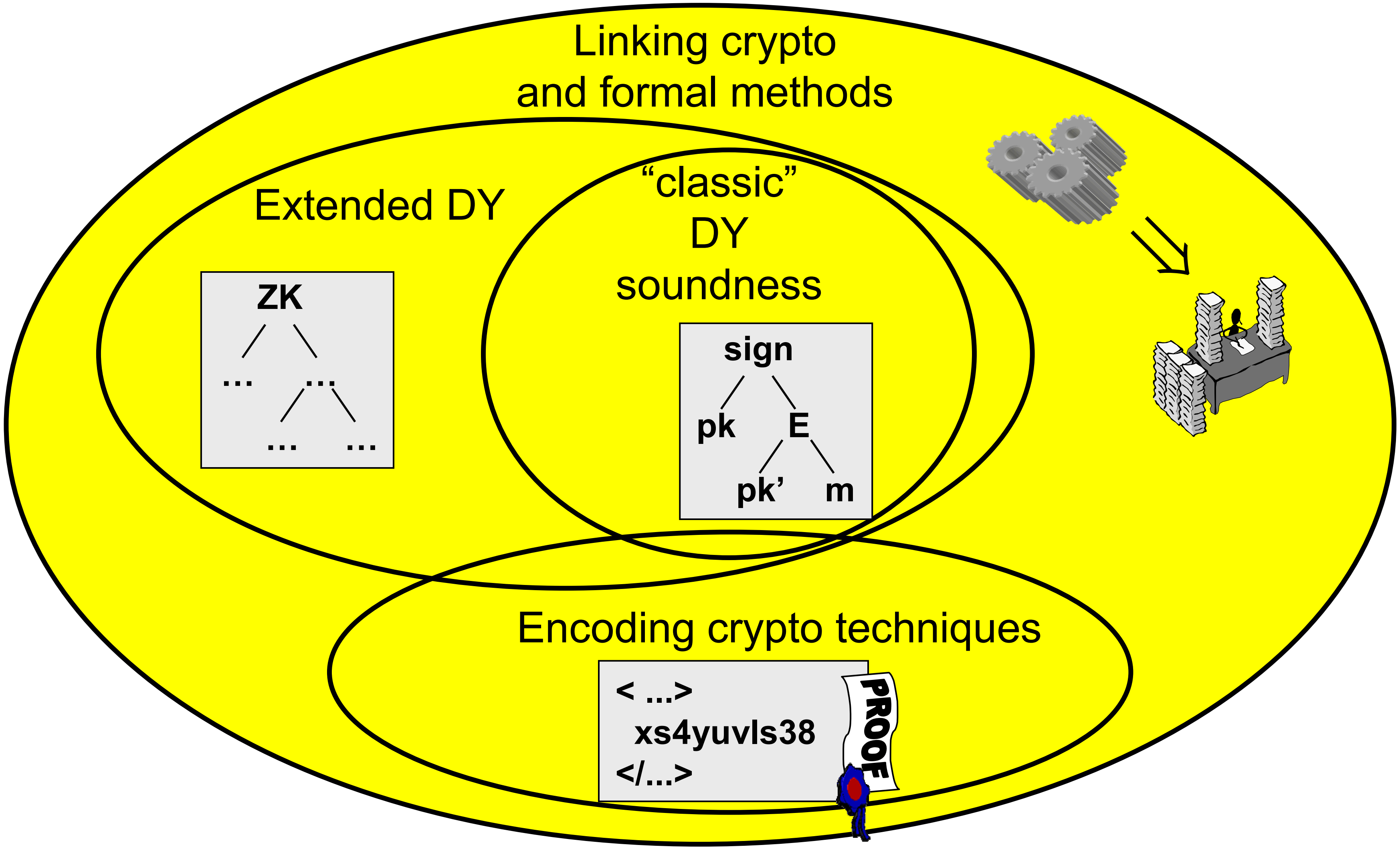
Overall

Module	Theories	Pages	Lines	%
Modeling & verification tools	9	54	2.7k	7
DAG-based model + BRSIM	10	119	6k	16
Term-based model	17	96	4.8k	13
Term-based NSL	14	136	6.8k	18
APM + BRSIM	31	266	13.3k	35
APM-based NSL	9	83	4.2k	11
Total	90	754	37.8k	100



Abstract protocol model, RSIM proof, and NSL case study

Bigger picture and some possible next steps


Linking crypto
and formal methods
Extended DY
ZK
…
…
…
…
"classic"
DY
soundness
sign
pk
E
pk'
m
Encoding crypto techniques
< …>
xs4yuvls38
</…>
PROOF