NICTA

Let's build secure systems on a correct kernel

June Andronick

© NICTA

Australian Government
Department of Broadband, Communications and the Digital Economy
Australian Research Council

NICTA Funding and Supporting Members and Partners

Credits

NICTA

Gerwin Klein

(part of) the Trustworthy Embedded Systems crowd

© NICTA 2010

Agenda

NICTA

Secure Systems "à la NICTA"

For 1 critical system:

- 1 desired security property P
- an interactive theorem prover
- a bit of patience

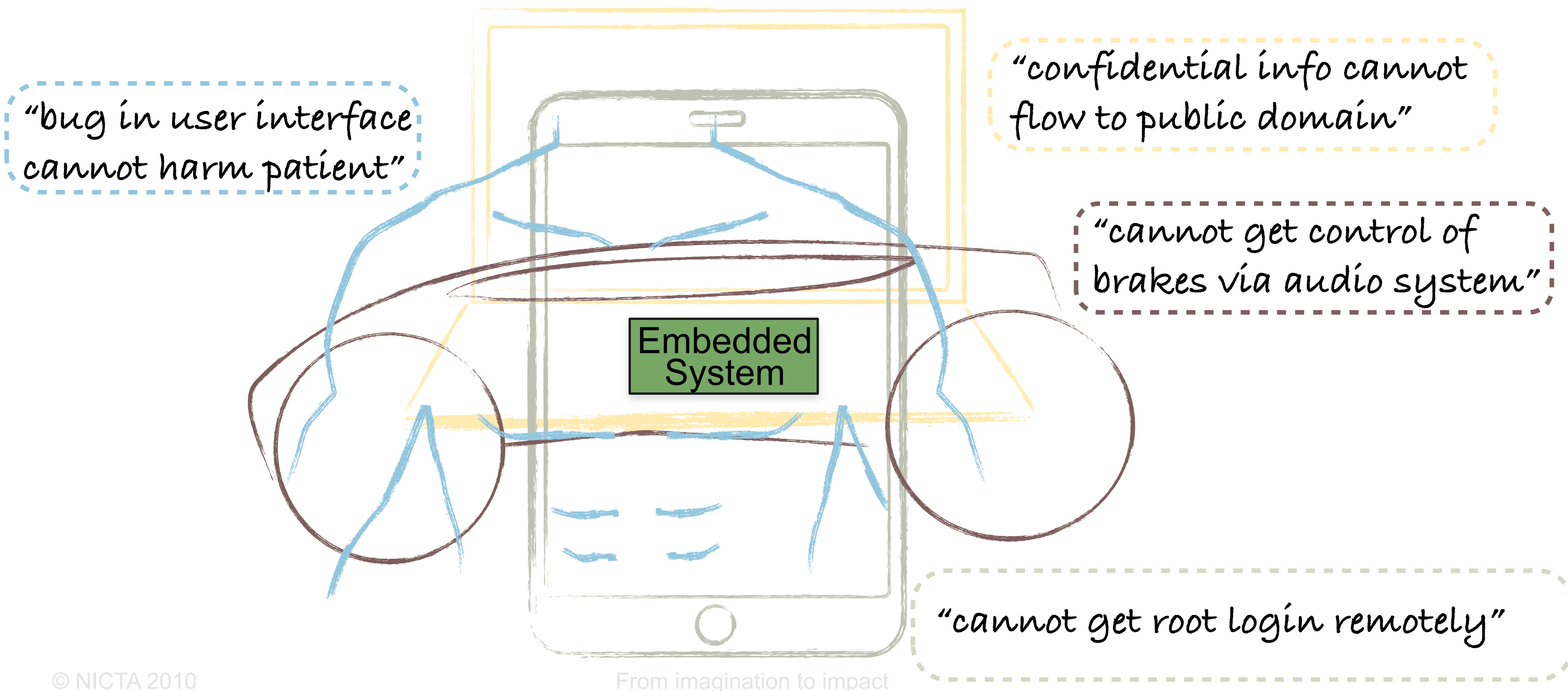
1. carefully design your system
2. prove that the design enforces P
3. prove correctness of the TCB
4. prove isolation

© NICTA 2010

Motivation

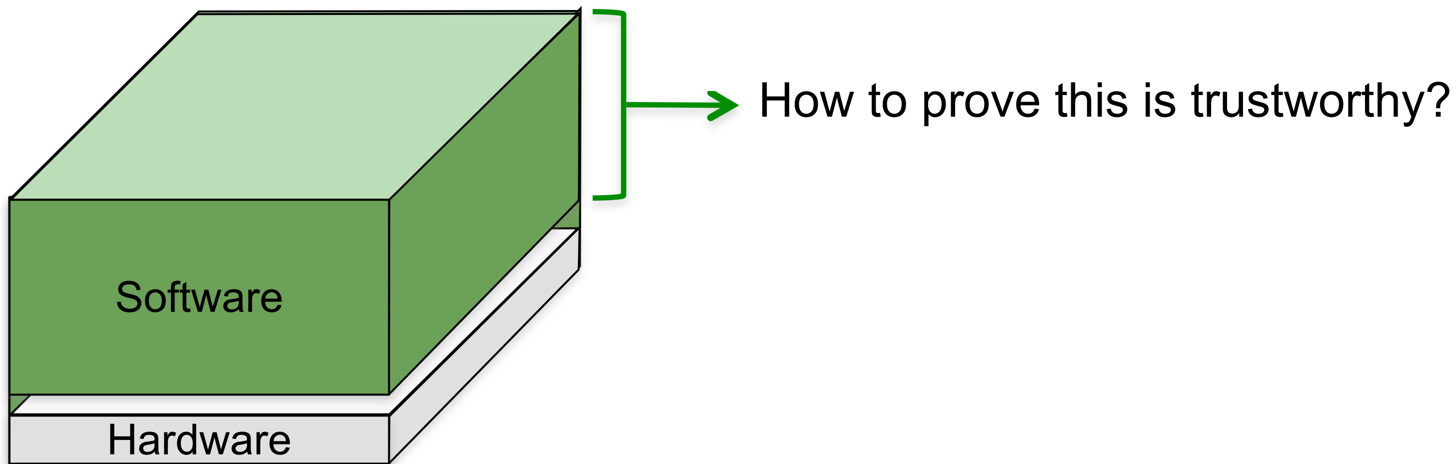
Aim: Trustworthy Embedded Systems

Target: code-level guarantee at reasonable cost — real-world, usable systems (low-level, performant)



© NICTA 2010

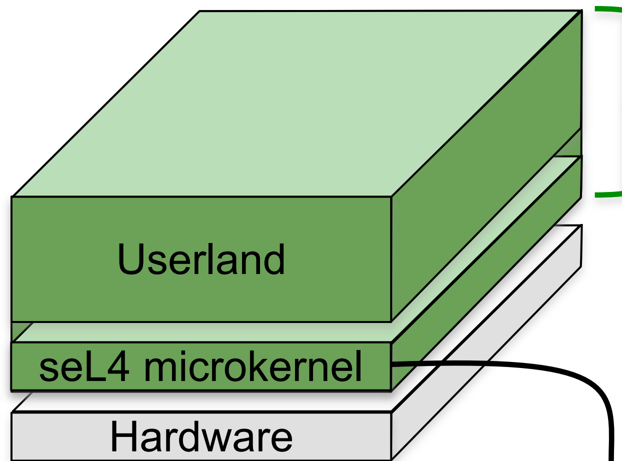
Approach



© NICTA 2010

Approach

NICTA



How to prove this is trustworthy?

1. Trustworthy foundation → **seL4**

functional correctness for 10,000 loc

Formal functional spec

Formal proof of refinement

Code

L4.verified

Result: "Every behavior of the code is a behavior of the spec"

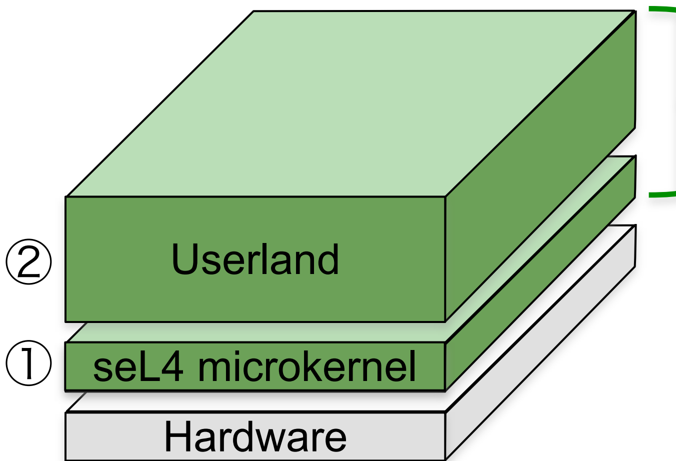
Corollary: "execution always defined" (no buffer overflows, …)

Assumptions:
- compiler + linker
- assembly code (600 loc)
- hardware (ARMv6)
- cache/TLB
- boot code (1,200 loc)

WORK IN PROGRESS

© NICTA 2010

Approach



How to prove this is trustworthy?

1. Trustworthy foundation → **seL4**

   functional correctness for 10,000 loc

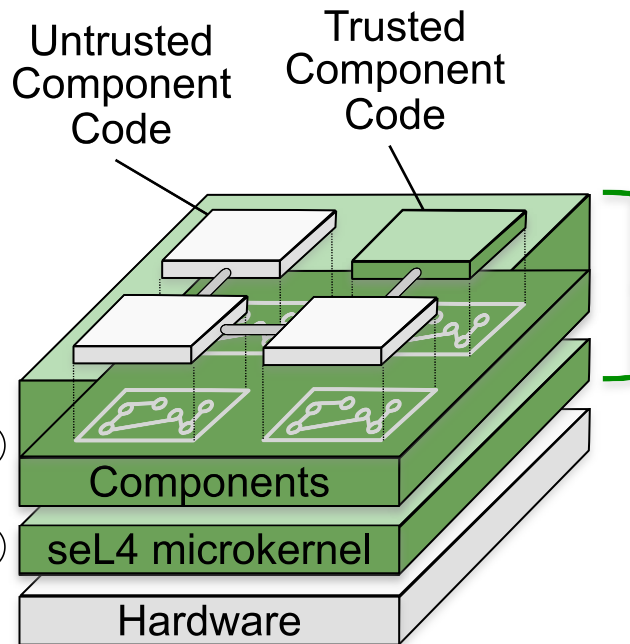
2. Strategic componentized security architecture

   formal guarantees for >1,000,000 loc

Idea: Strong *guarantees* about *whole system* without needing to reason about all of its code

© NICTA 2010

Approach



How to prove this is trustworthy?

1. Trustworthy foundation → **seL4**
   functional correctness for 10,000 loc

2. Strategic componentized security architecture
   formal guarantees for >1,000,000 loc

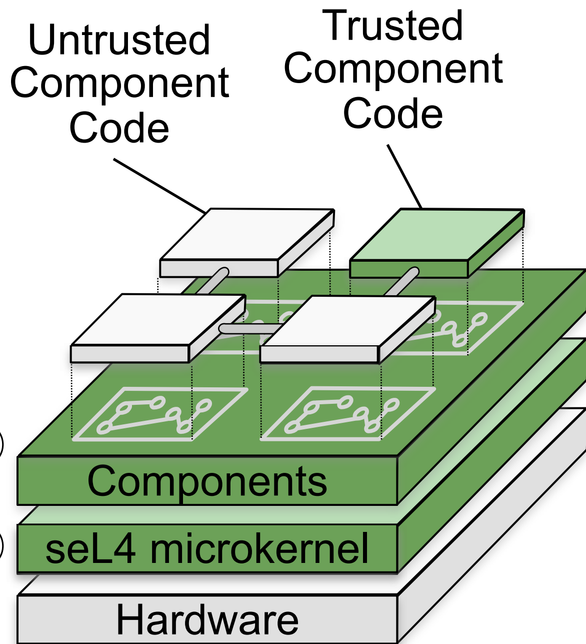
Idea: Strong *guarantees* about *whole system* without needing to reason about all of its code

How: Using seL4's access control (*capabilities*)

© NICTA 2010

Approach

NICTA

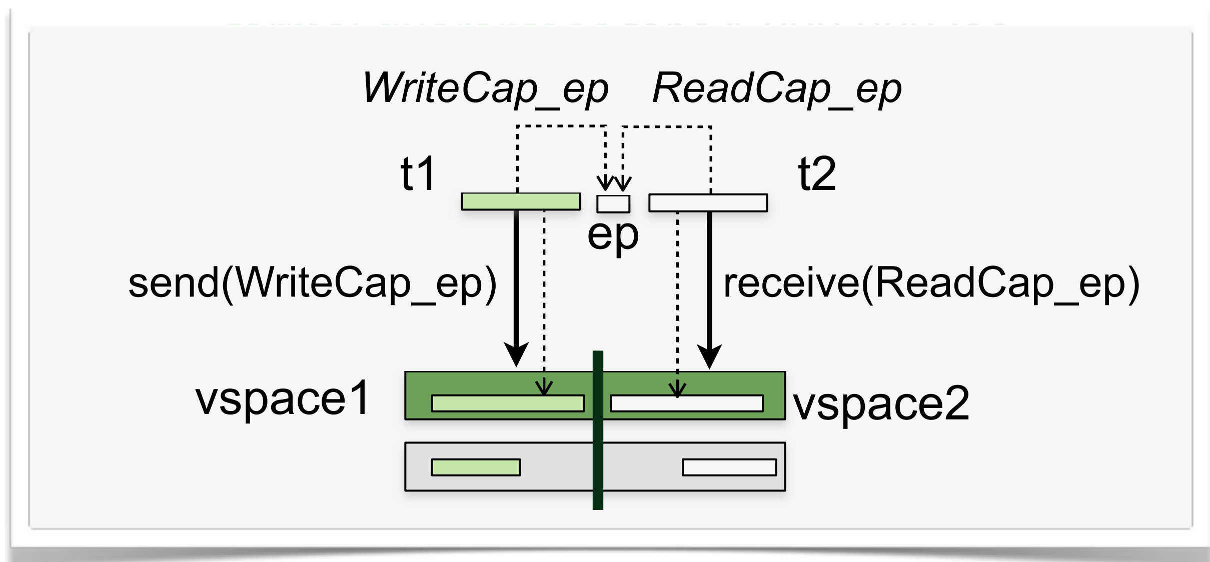


How to prove this is trustworthy?

1. Trustworthy foundation → **seL4**

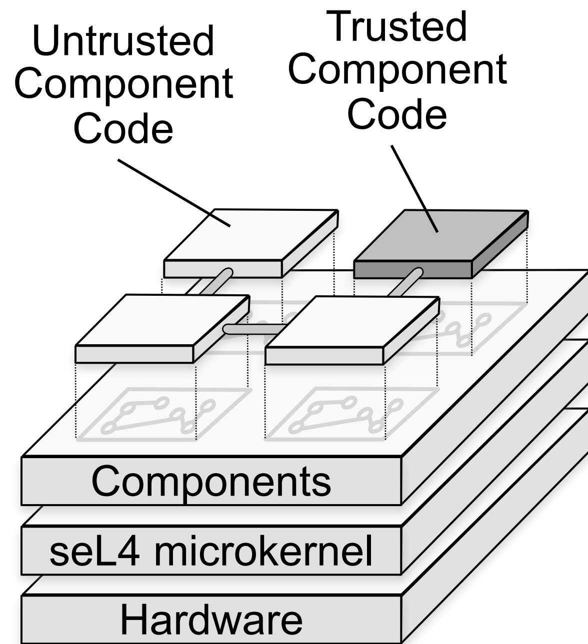
functional correctness for 10,000 loc

2. Strategic componentized security architecture



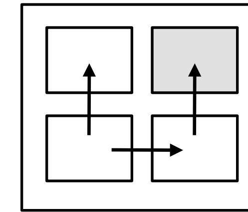
© NICTA 2010

Careful design



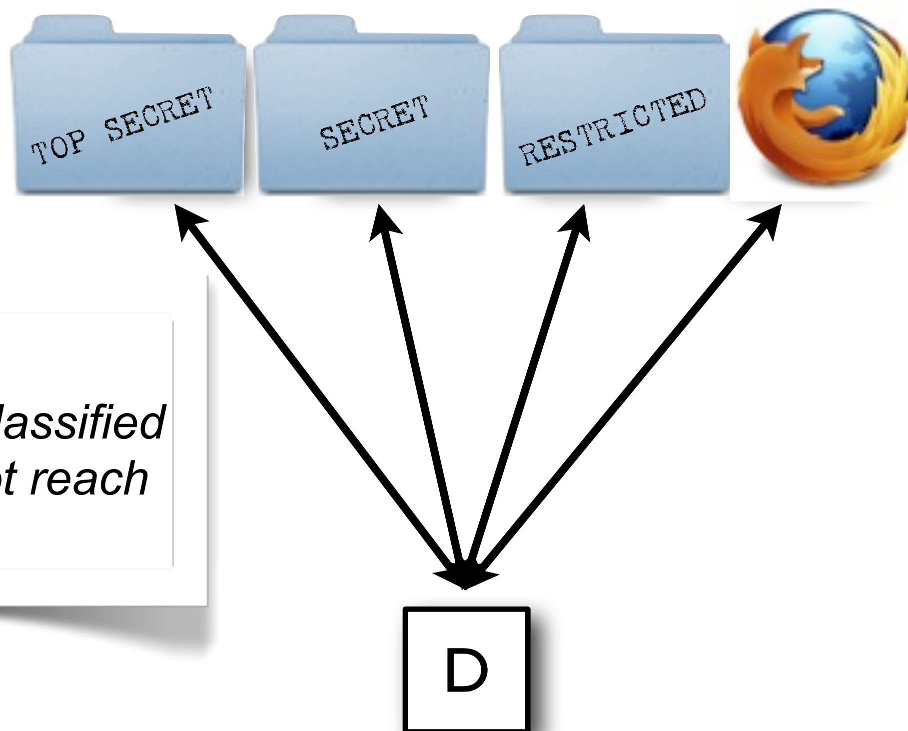
System Implementation

Security Architecture



1a. minimal Trusted Computing Base

Case-study



Goal:

Data from one classified network must not reach another

Secure Access Controller (SAC)

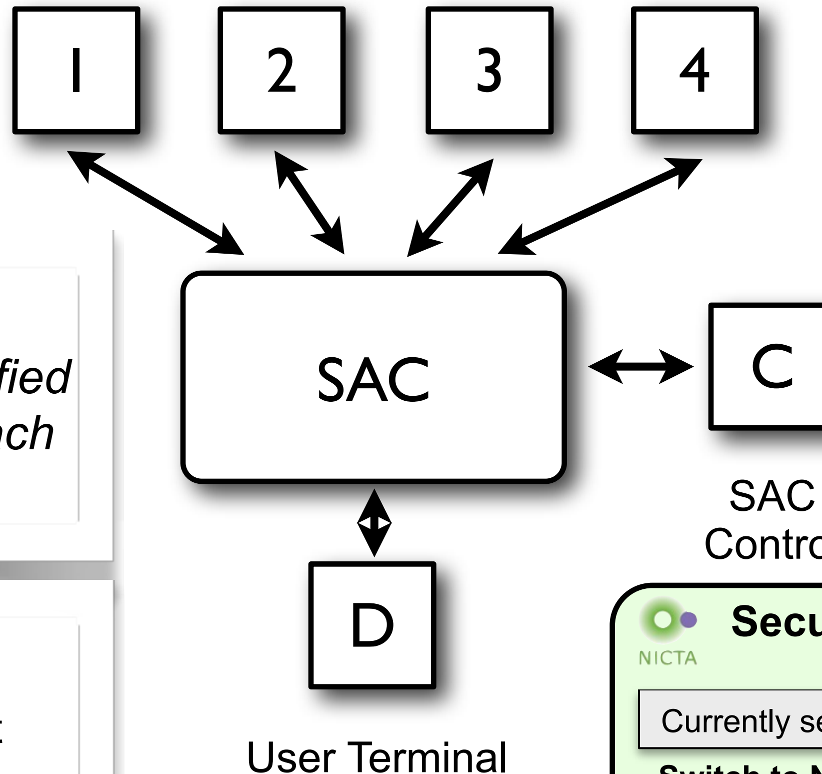
NICTA

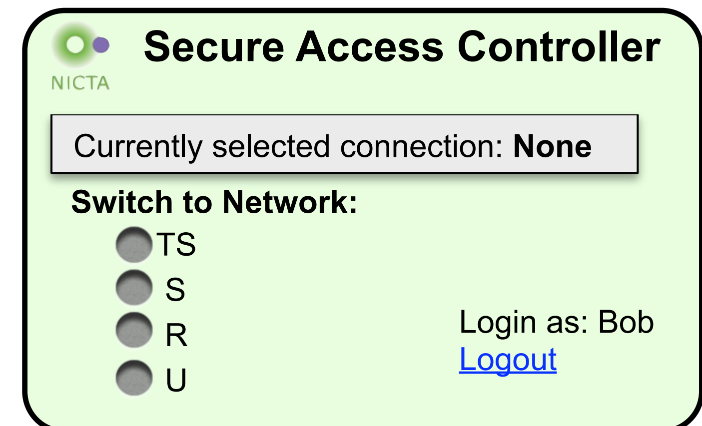
Goal:

Data from one classified network must not reach another

Assumptions:

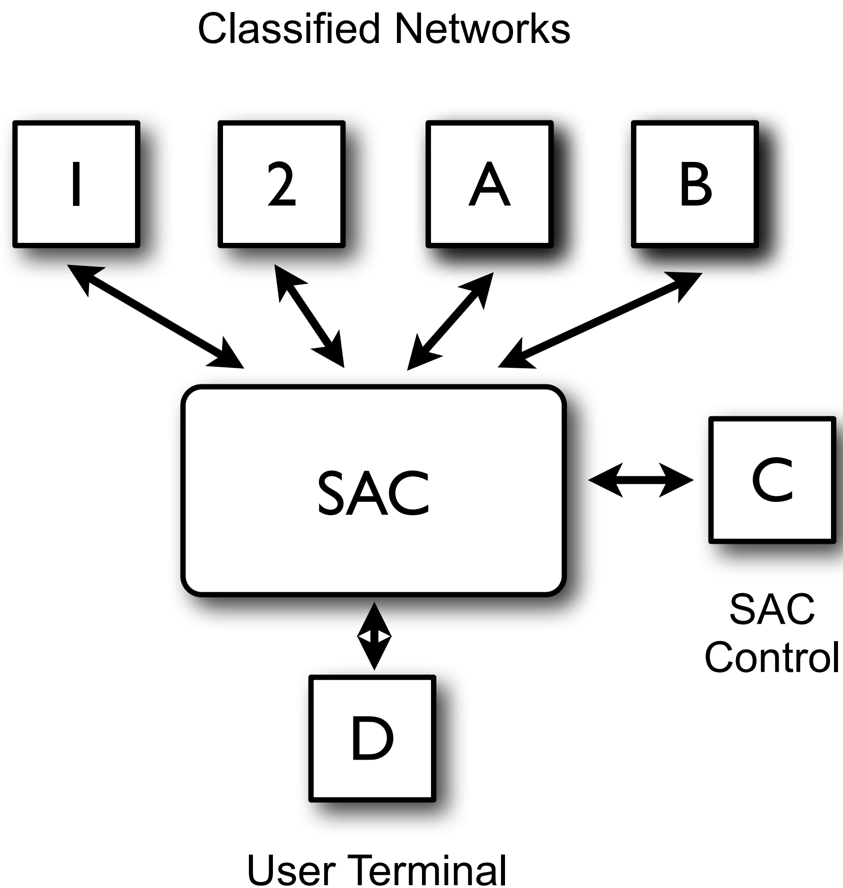
- User terminal will not leak data
- All networks are otherwise malicious





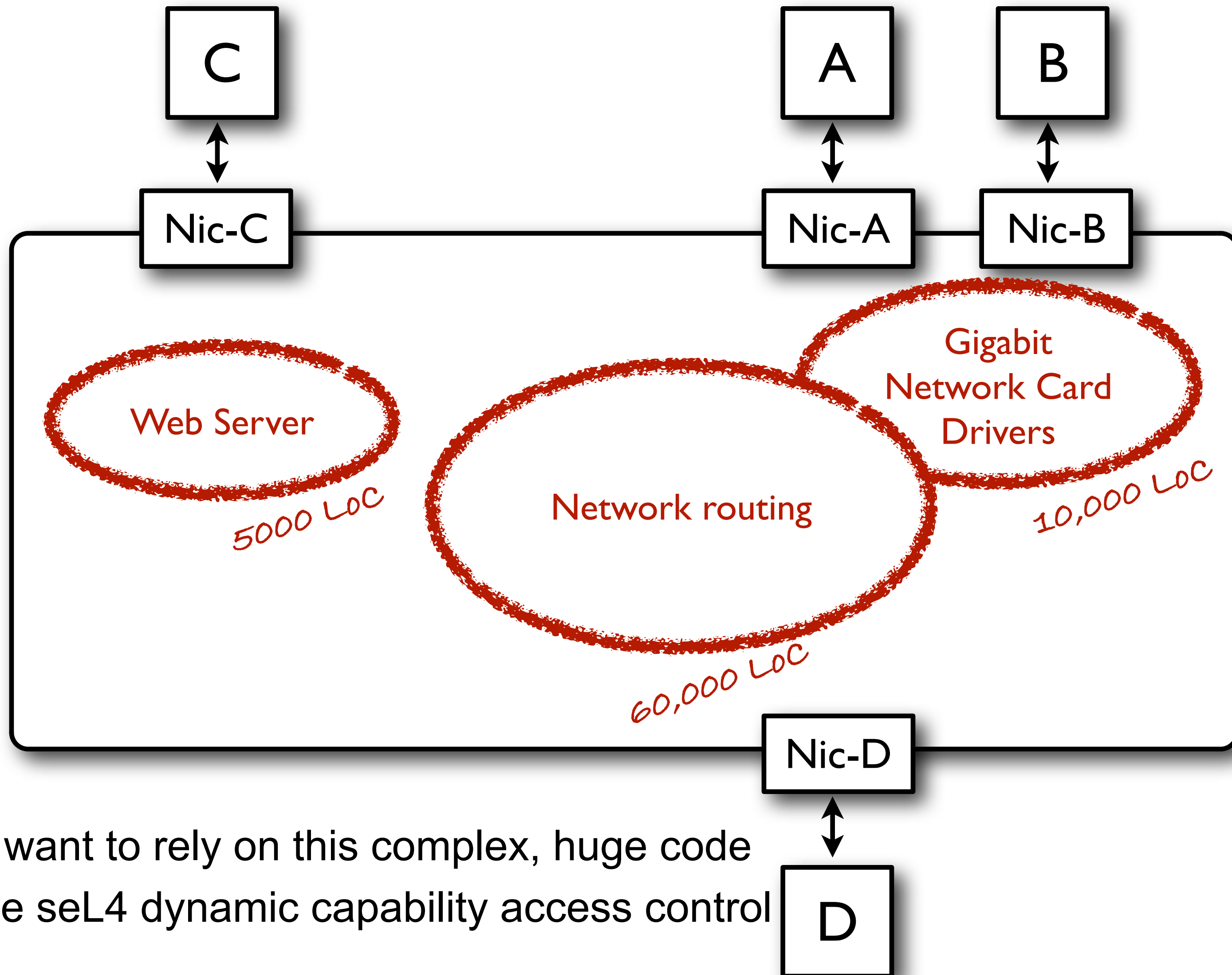
© NICTA 2010

Design



© NICTA 2010

Minimal TCB

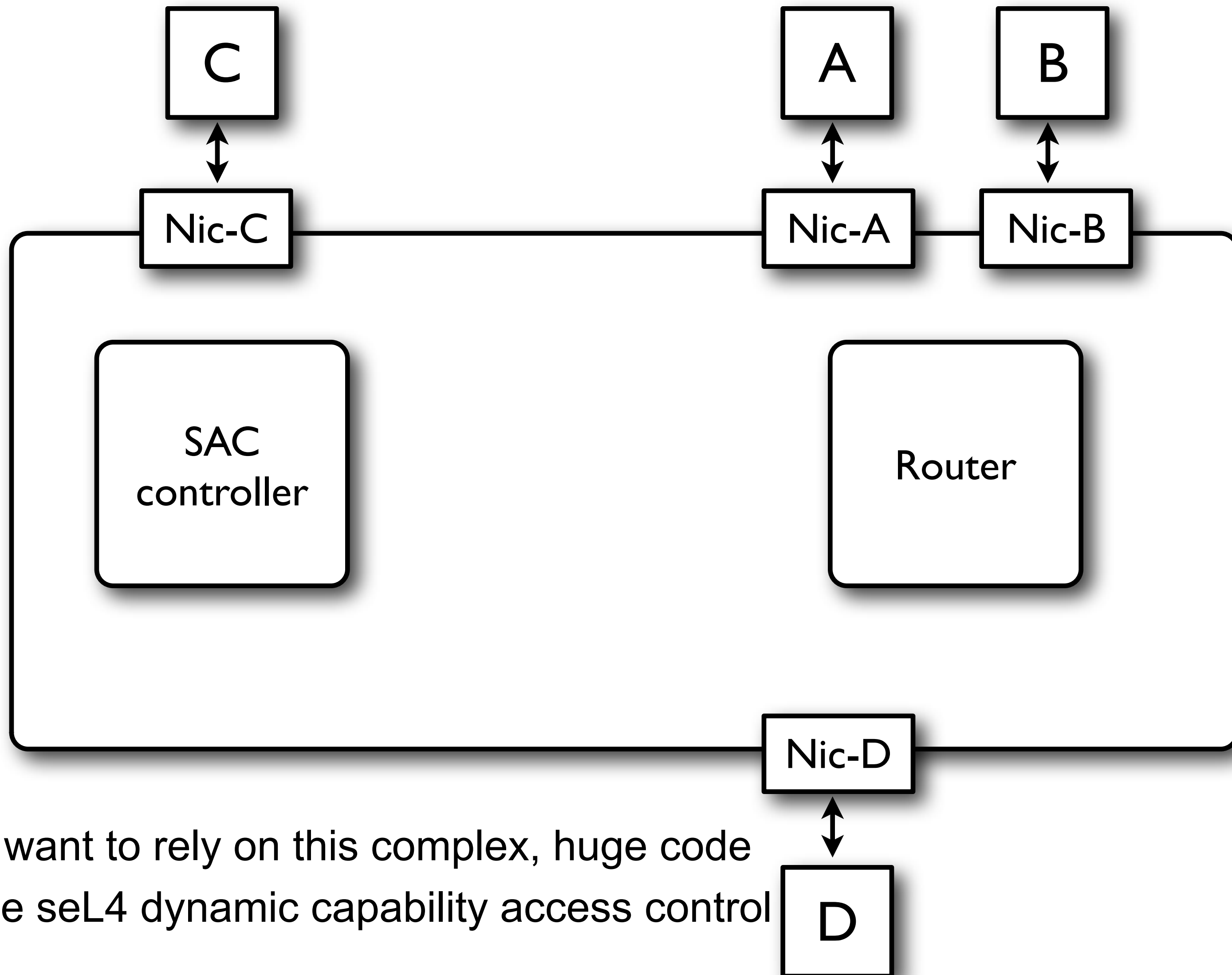


We don't want to rely on this complex, huge code

➜ We use seL4 dynamic capability access control

© NICTA 2010

Minimal TCB

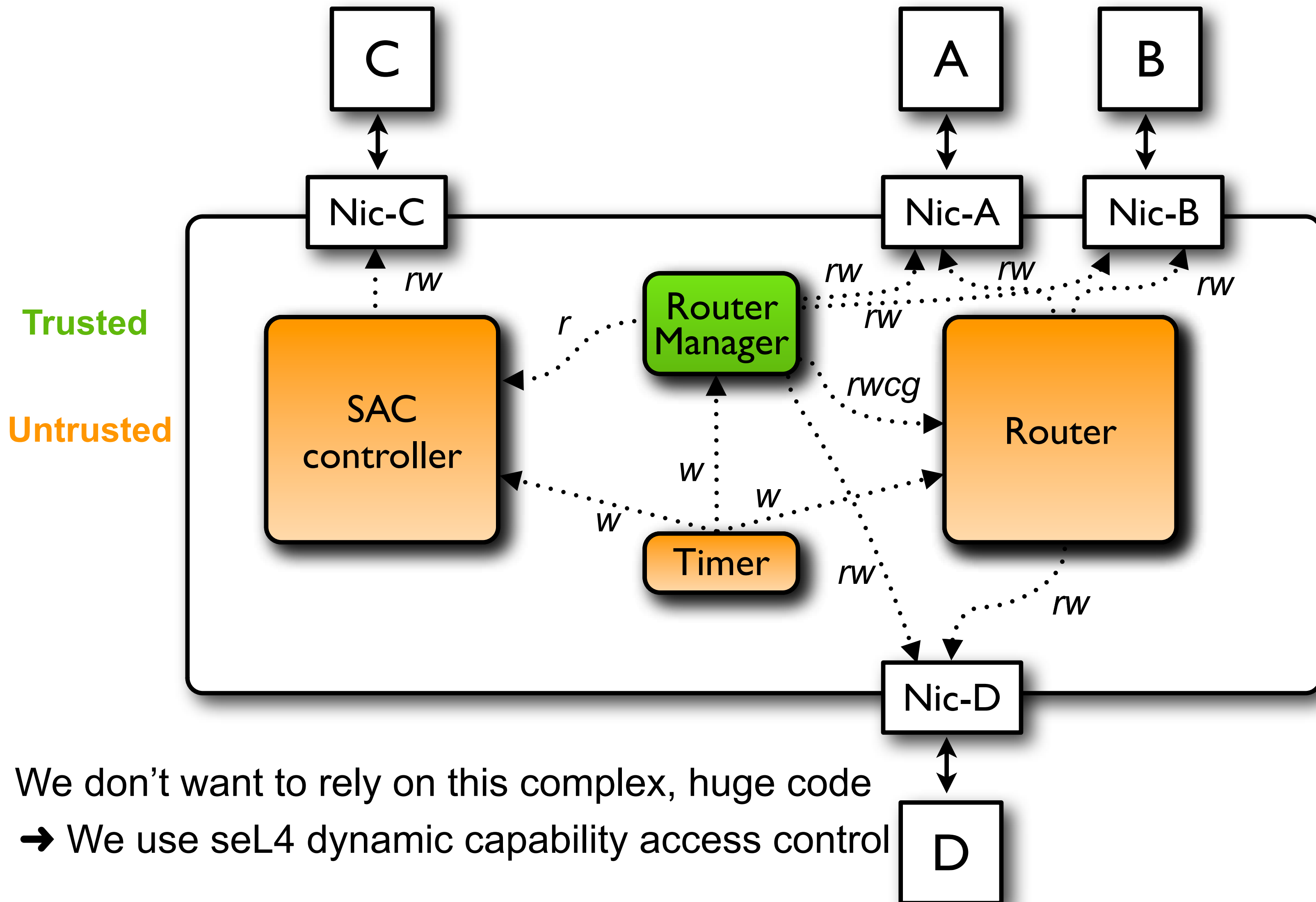


We don't want to rely on this complex, huge code

➜ We use seL4 dynamic capability access control

© NICTA 2010

Minimal TCB



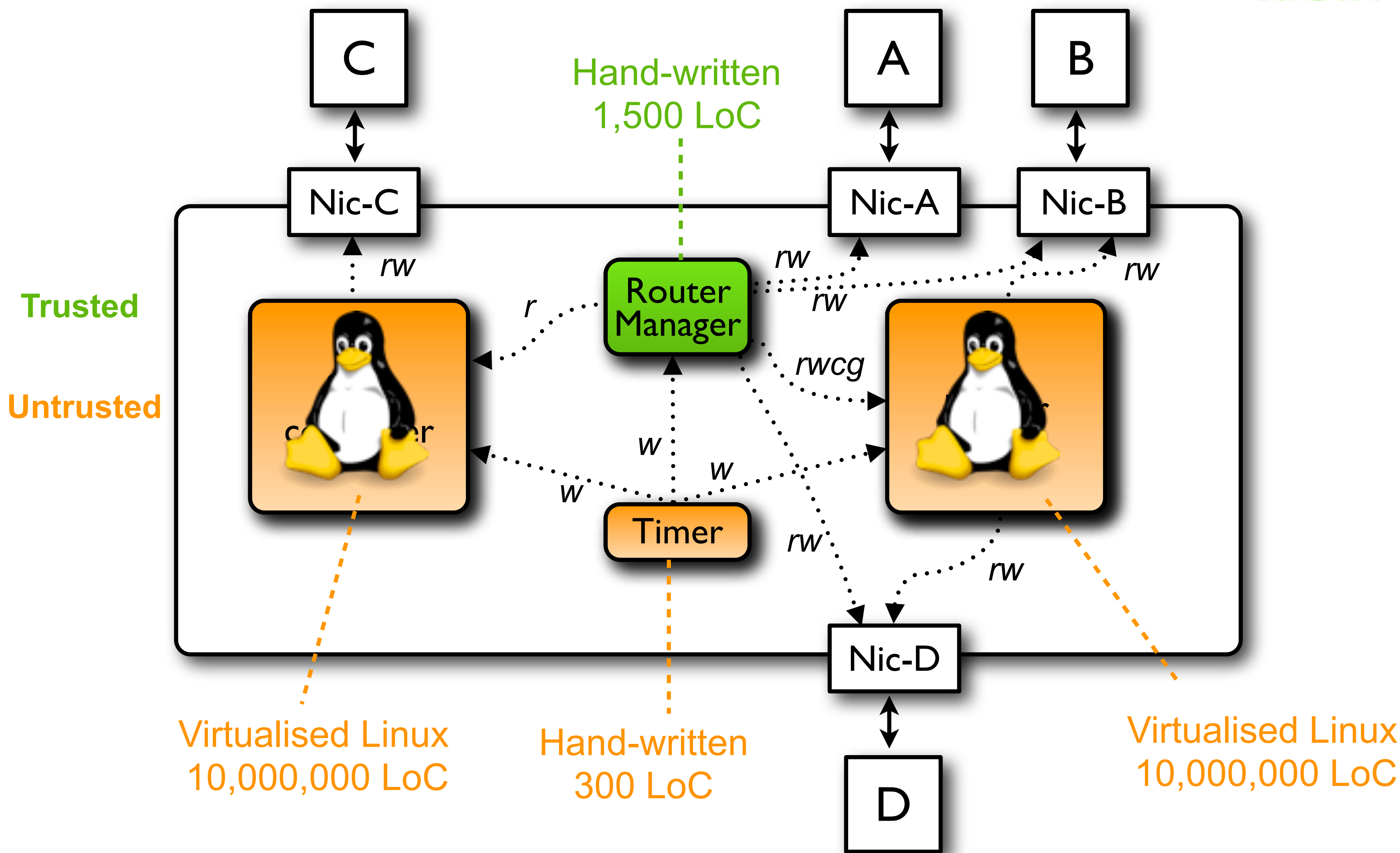
We don't want to rely on this complex, huge code

➜ We use seL4 dynamic capability access control

© NICTA 2010

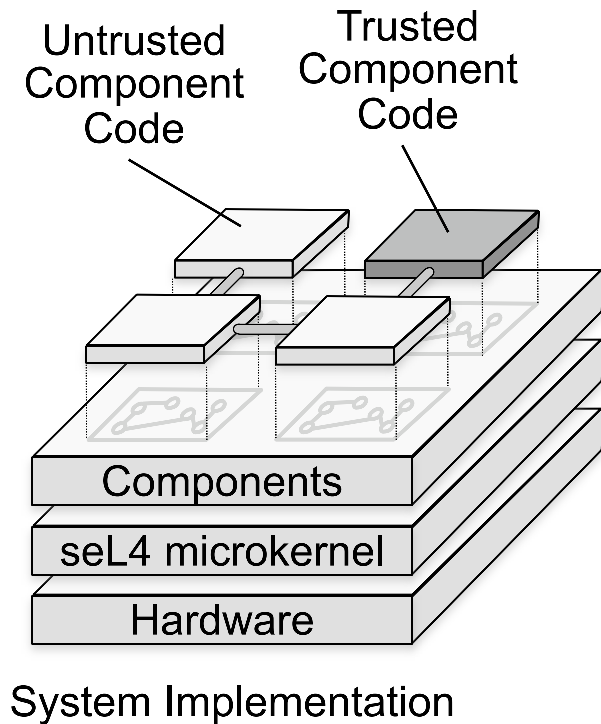
Minimal TCB: Implementation

NICTA

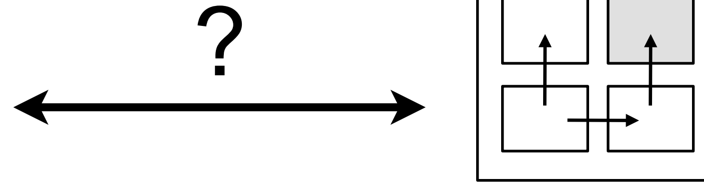


© NICTA 2010

Back to the general picture



Security Architecture



1a. minimal Trusted Computing Base ✓

Now: how to set-up the system with this design?

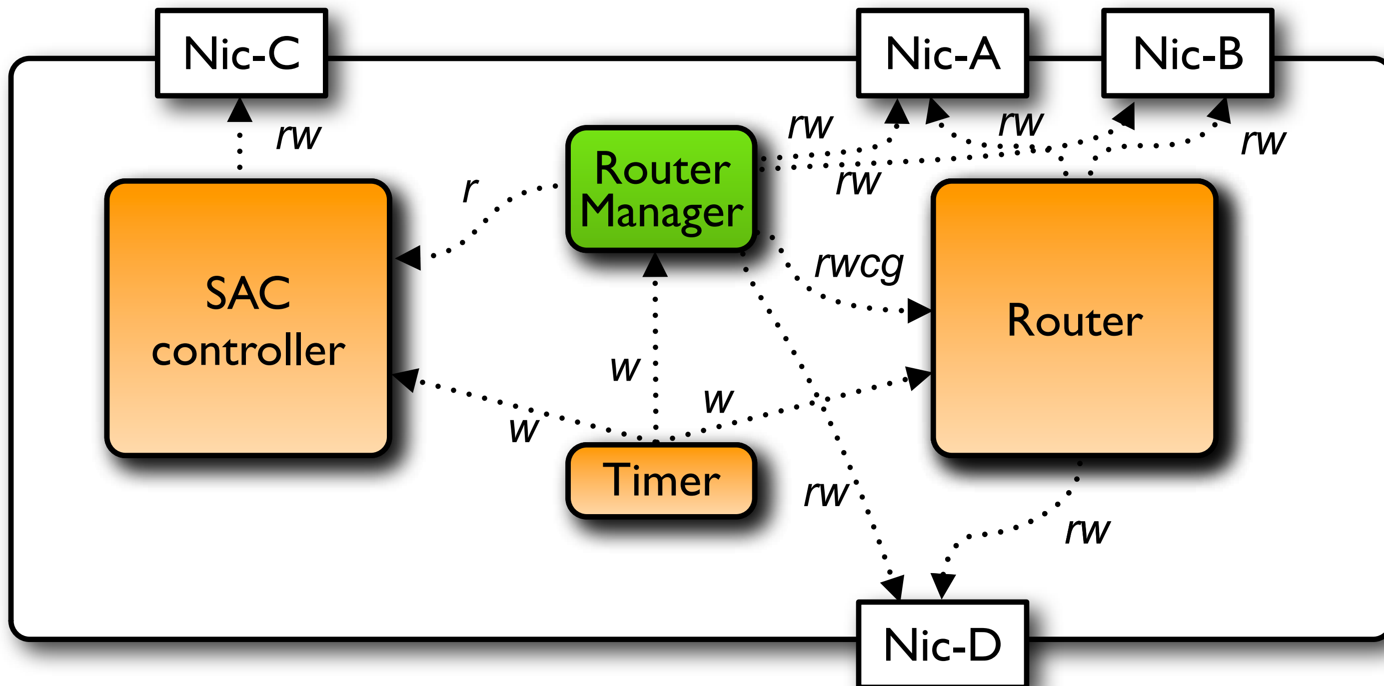
Problem: reality is not that simple

© NICTA 2010

Back to the example

This is what we agree on the whiteboard

Now we need to implement this with actual kernel objects

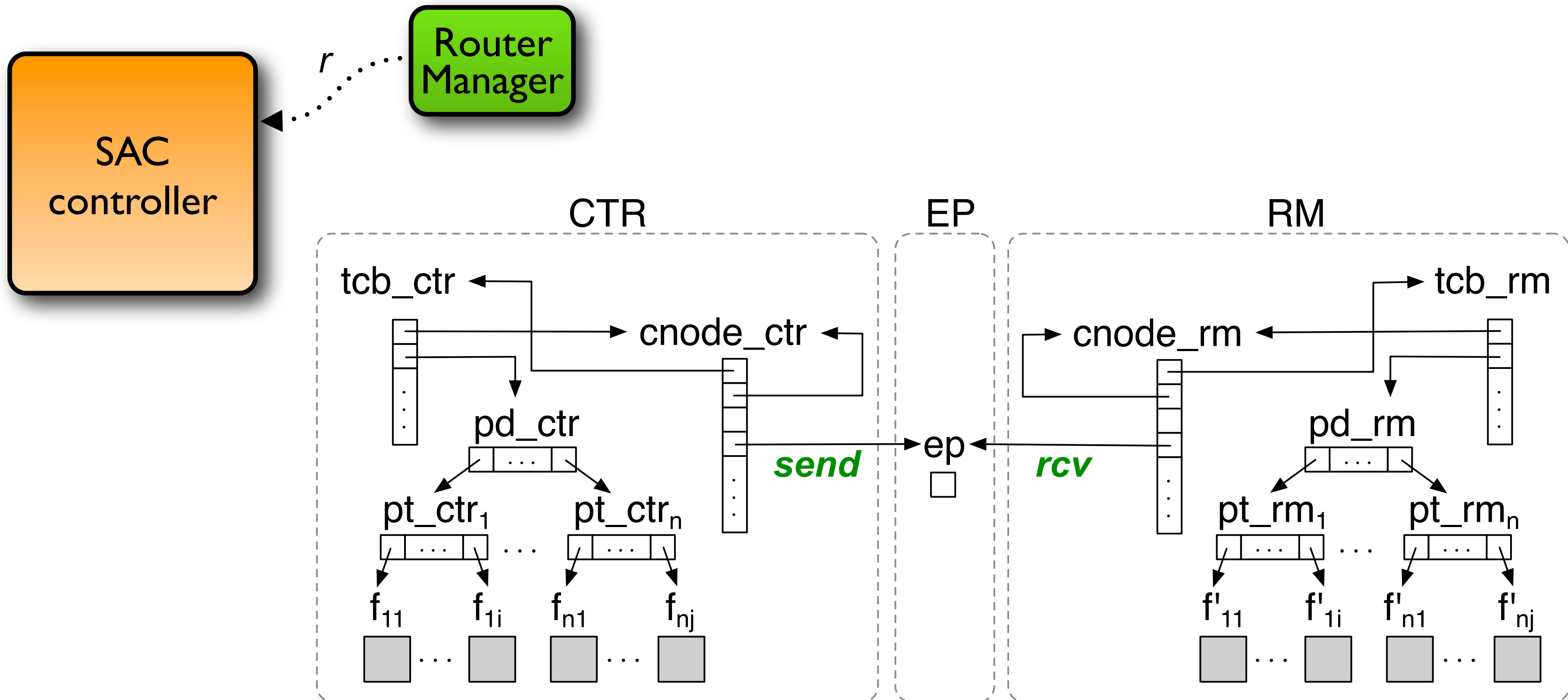


© NICTA 2010

Back to the example

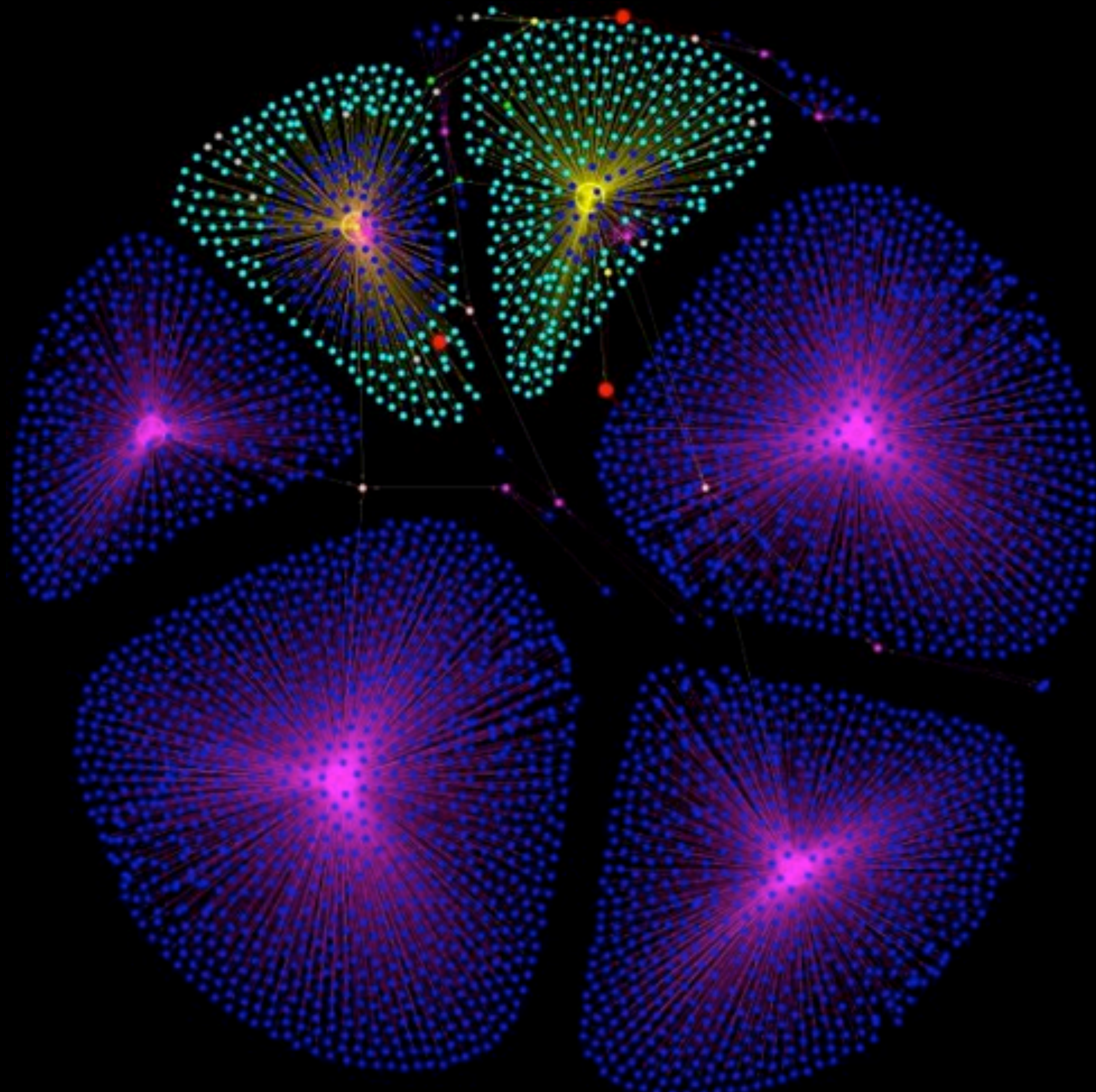
This is what we agree on the whiteboard

Now we need to implement this with actual kernel objects



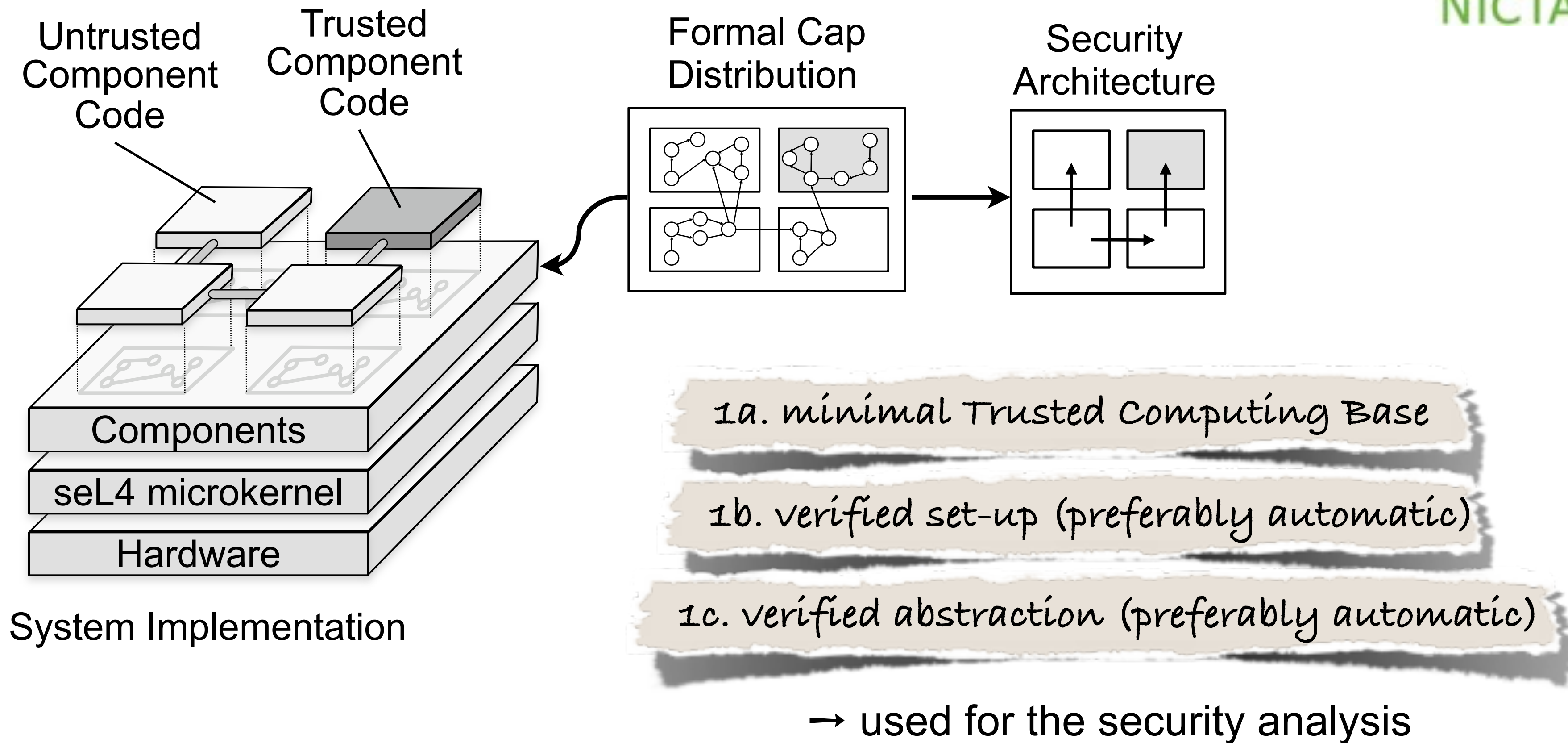
Every arrow is a capability!

© NICTA 2010



capability distribution

Back to the general picture



→ used for the security analysis

capDL: capability distribution language

Example:

$$obj1 \equiv Tcb[0 \mapsto CNodeCap\ 3, ...]$$
$$obj3 \equiv CNode[302 \mapsto CNodeCap\ 9 \quad Read \ \ , ...]$$

© NICTA 2010

Agenda

NICTA

<u>Secure Systems</u> "à la NICTA"

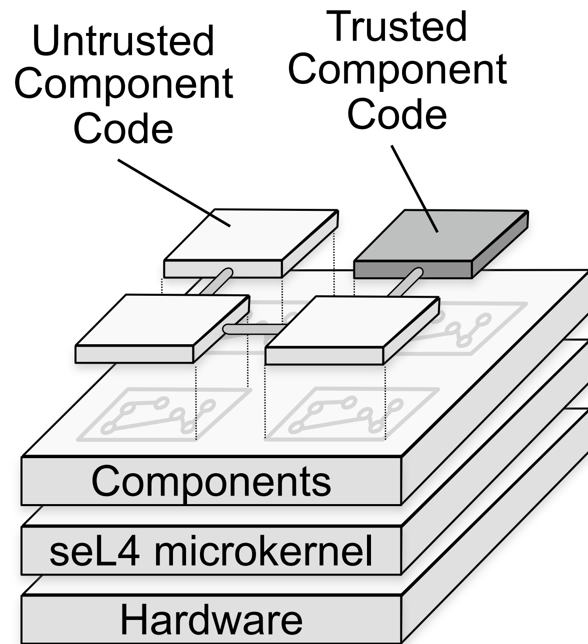
For 1 critical system:

- 1 desired security property P
- an interactive theorem prover
- a bit of patience

1. carefully design your system
2. prove that the design enforces P
3. prove correctness of the TCB
4. prove isolation

© NICTA 2010

Security Proof



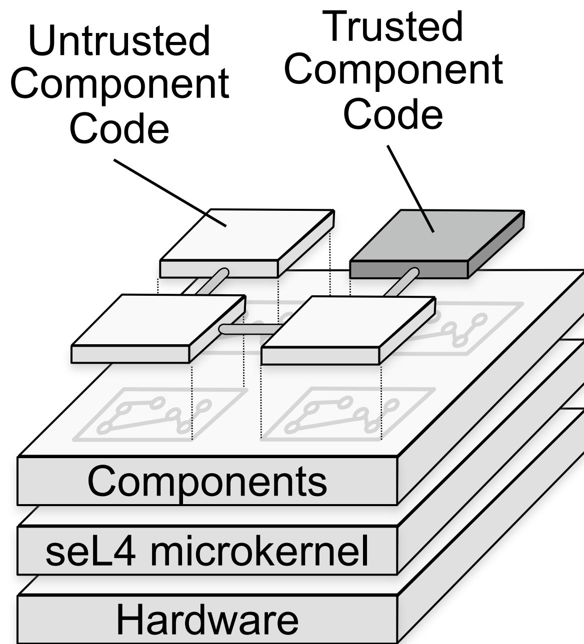
System Implementation

Theorem: $s_0 \xrightarrow{*} s \Rightarrow \mathcal{P}(s)$

```
lemma sacSecurity:
  (SAC-startup →* s) ⇒
   ¬ is_contaminated s NicA
```

© NICTA 2010

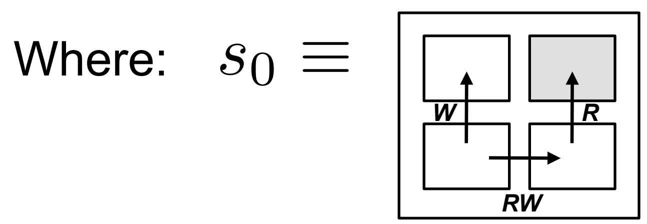
Security Proof



System Implementation

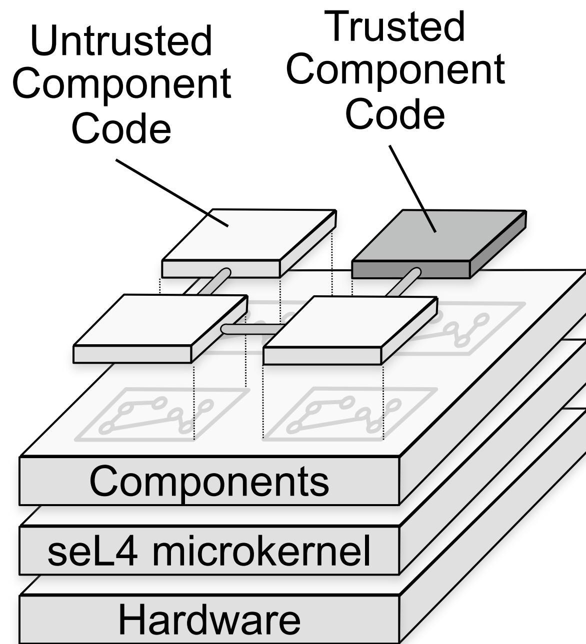
Theorem: $s_0 \xrightarrow{*} s \Rightarrow \mathcal{P}(s)$

Where: $s_0 \equiv$




```
RM_id      -> Some ({cap_r_to_SAC_C, ...}, not_contaminated)
SAC_C_id   -> Some ({cap_rw_to_NIC_C, ...}, not_contaminated)
NIC_A_id   -> Some ({}, not_contaminated)
NIC_B_id   -> Some ({}, contaminated)
```

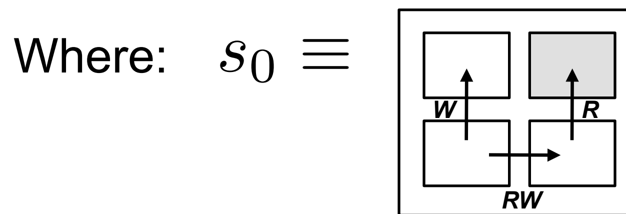
Security Proof



System Implementation

Theorem: $s_0 \xrightarrow{*} s \Rightarrow \mathcal{P}(s)$

Where: $s_0 \equiv$

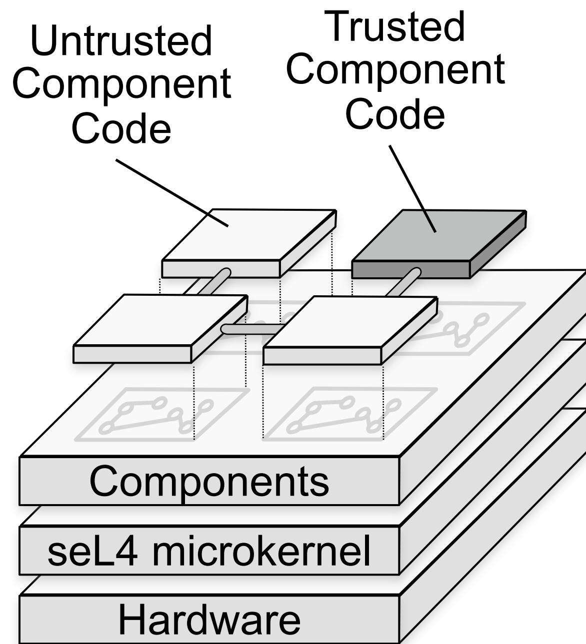


$$s \rightarrow s' \equiv s \xrightarrow{t} s' \vee s \xrightarrow{u} s'$$

$$\begin{aligned} s \xrightarrow{t} s' \equiv\ & let\ tc \in trusted_component(s)\ in \\ & let\ prg = program(tc)\ in \\ & let\ pc = program_counter(c, s)\ in \\ & let\ i = inst(prg, pc)\ in \\ & step(tc, s, i, s') \end{aligned}$$

```
RM_prg ≡
  [ SysOp (SysRead cap_r_to_SAC_C),
    SysOp (SysRemoveAll cap_C_to_R),
    SysOp (SysDelete cap_C_to_R),
    ...  ]
```

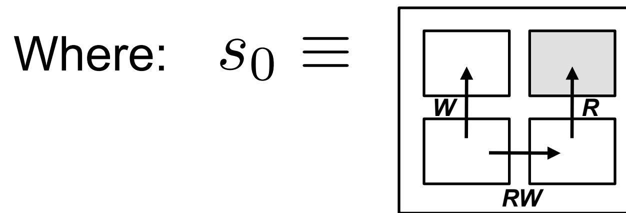
Security Proof



System Implementation

Theorem: $s_0 \xrightarrow{*} s \Rightarrow \mathcal{P}(s)$

Where: $s_0 \equiv$



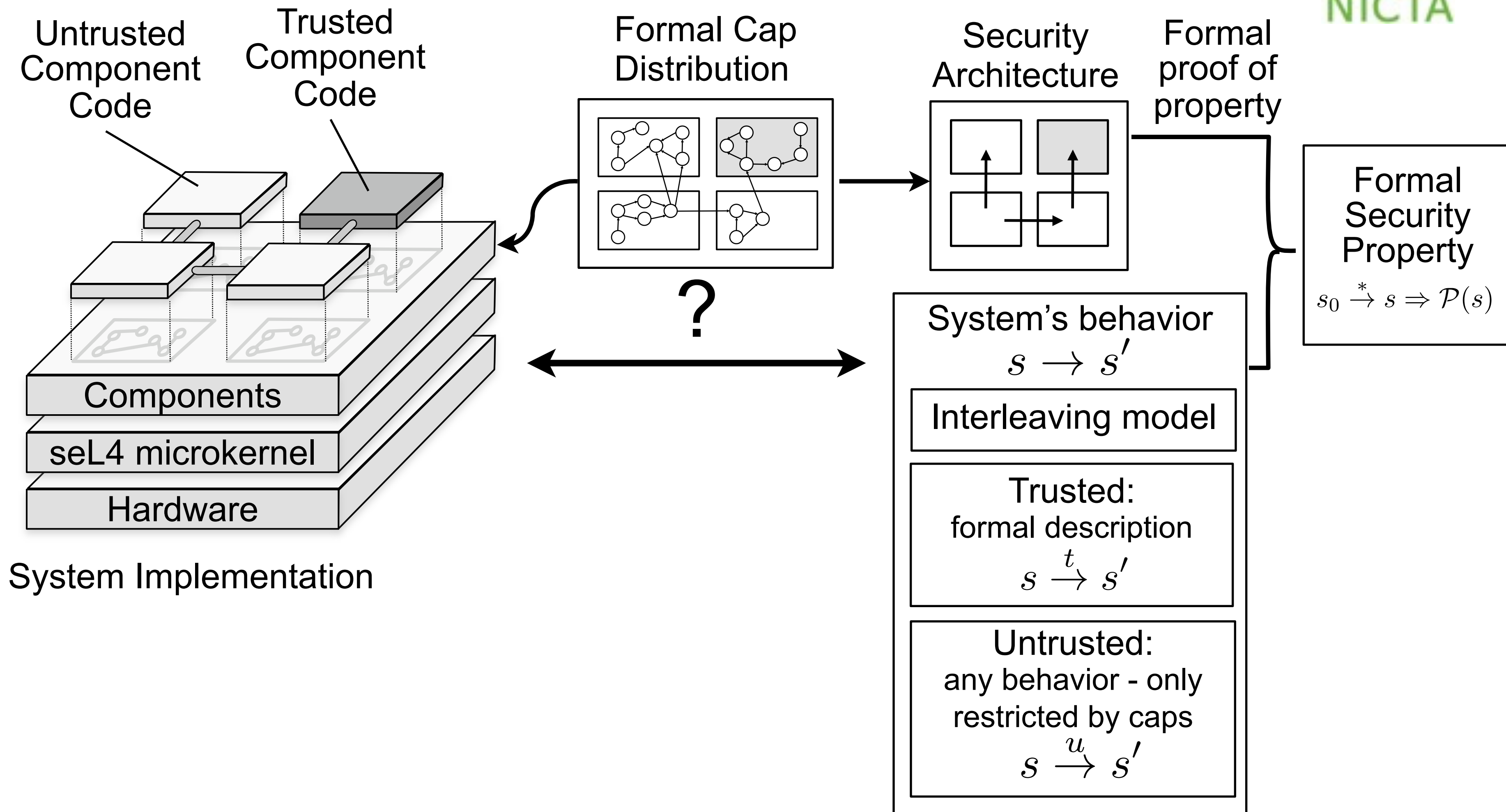
$$s \rightarrow s' \equiv s \xrightarrow{t} s' \vee s \xrightarrow{u} s'$$

$$\begin{aligned} s \xrightarrow{t} s' \equiv\ & let\ tc \in trusted_component(s)\ in \\ & let\ prg = program(tc)\ in \\ & let\ pc = program_counter(c, s)\ in \\ & let\ i = inst(prg, pc)\ in \\ & step(tc, s, i, s') \end{aligned}$$

$$\begin{aligned} s \xrightarrow{u} s' \equiv\ & let\ uc \in untrusted_components(s)\ in \\ & step(uc, s, any_inst, s') \end{aligned}$$

© NICTA 2010

Security Proof



© NICTA 2010

Agenda

NICTA

Secure Systems "à la NICTA"

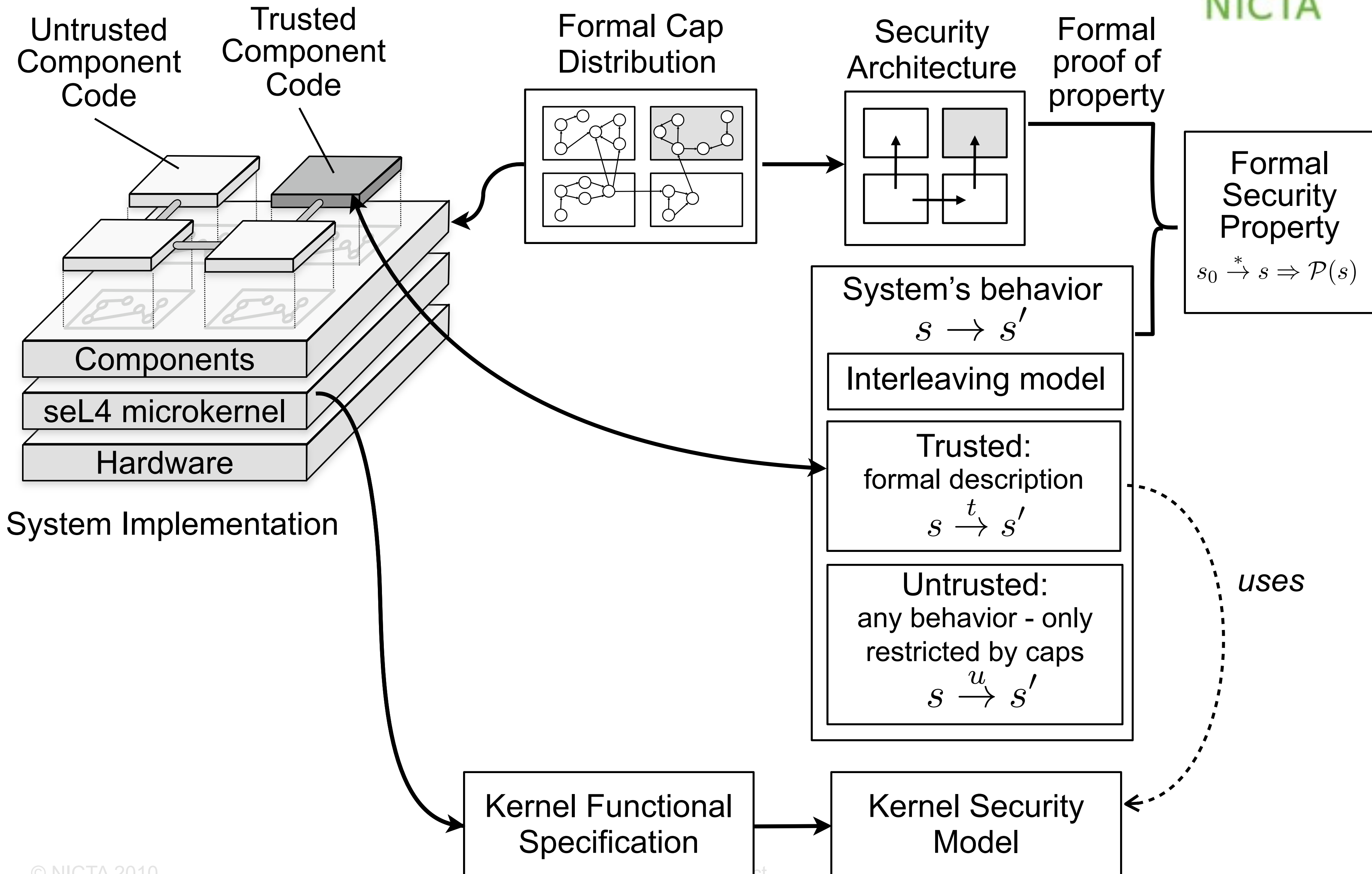
For 1 critical system:

- 1 desired security property P
- an interactive theorem prover
- a bit of patience

1. carefully design your system
2. prove that the design enforces P
3. prove correctness of the TCB
4. prove isolation

© NICTA 2010

Verified TCB



© NICTA 2010
From imagination to impact

Agenda

NICTA

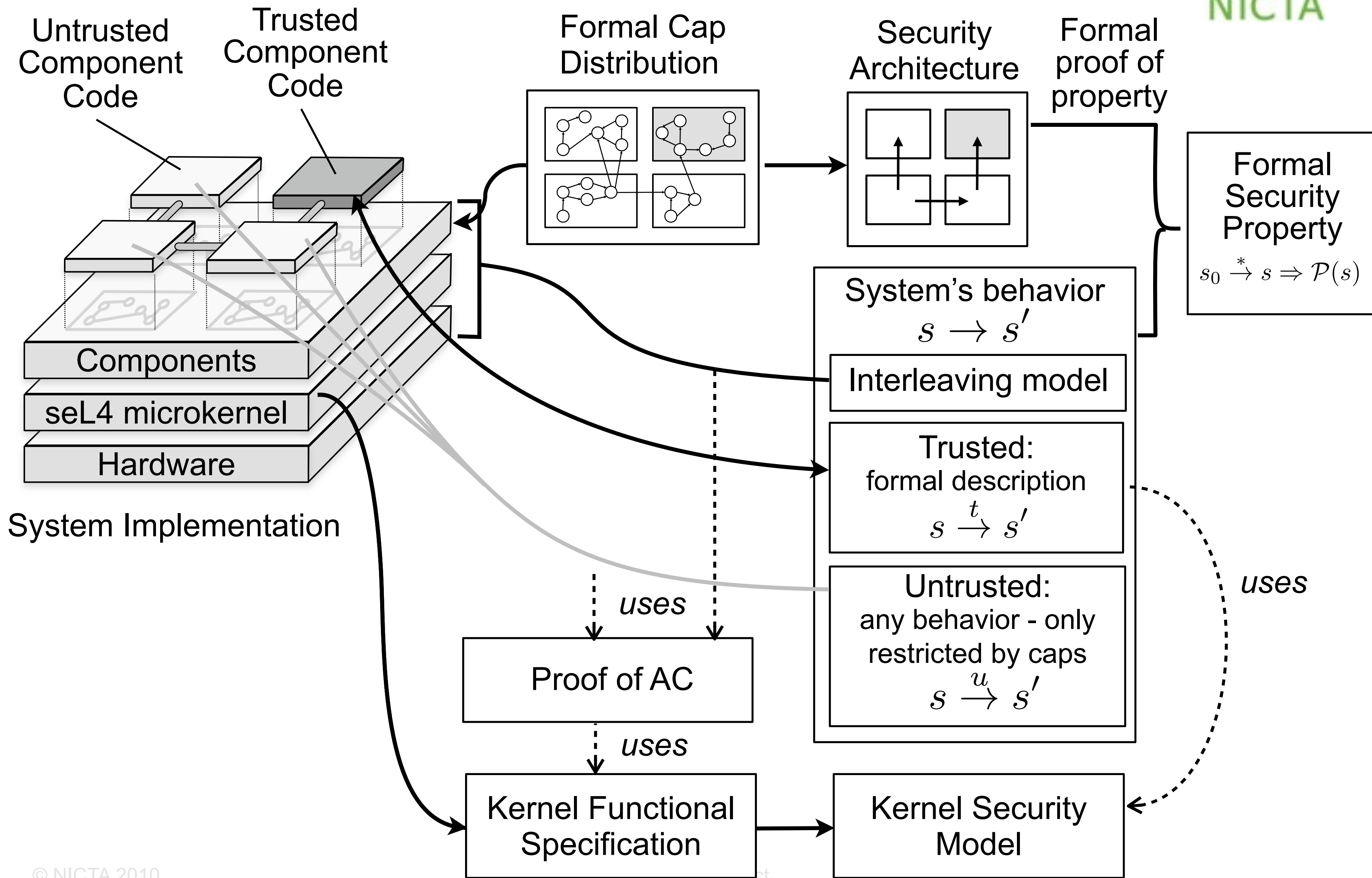
Secure Systems "à la NICTA"

For 1 critical system:

- 1 desired security property P
- an interactive theorem prover
- a bit of patience

1. carefully design your system
2. prove that the design enforces P
3. prove correctness of the TCB
4. prove isolation

Proof of access control

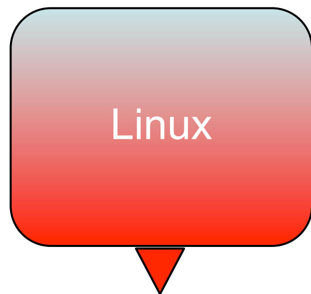


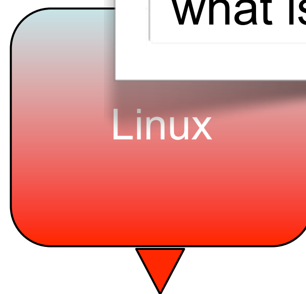
© NICTA 2010

What is AC good for?

NICTA Copyright 2010

What is AC good for?





Examples

- R does not write to NicB if it does not have a write capability to it
- R does not change RM's program counter

Question: for all operation *op* s.t.

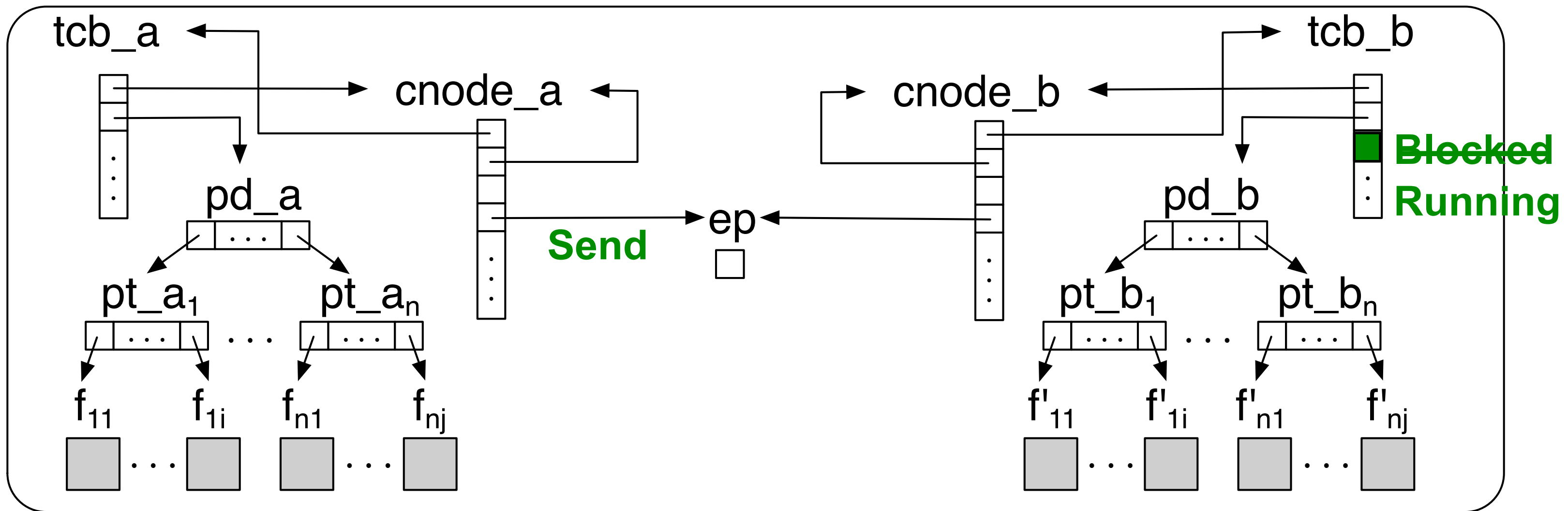
$$s \xrightarrow{op} s'$$

what is allowed to change in s' ?

NICTA Copyright 2010

Example

If ***op*** is ***set_thread_state tcb_b v***
If **tcb_a** is running in state **s** where **s** is:



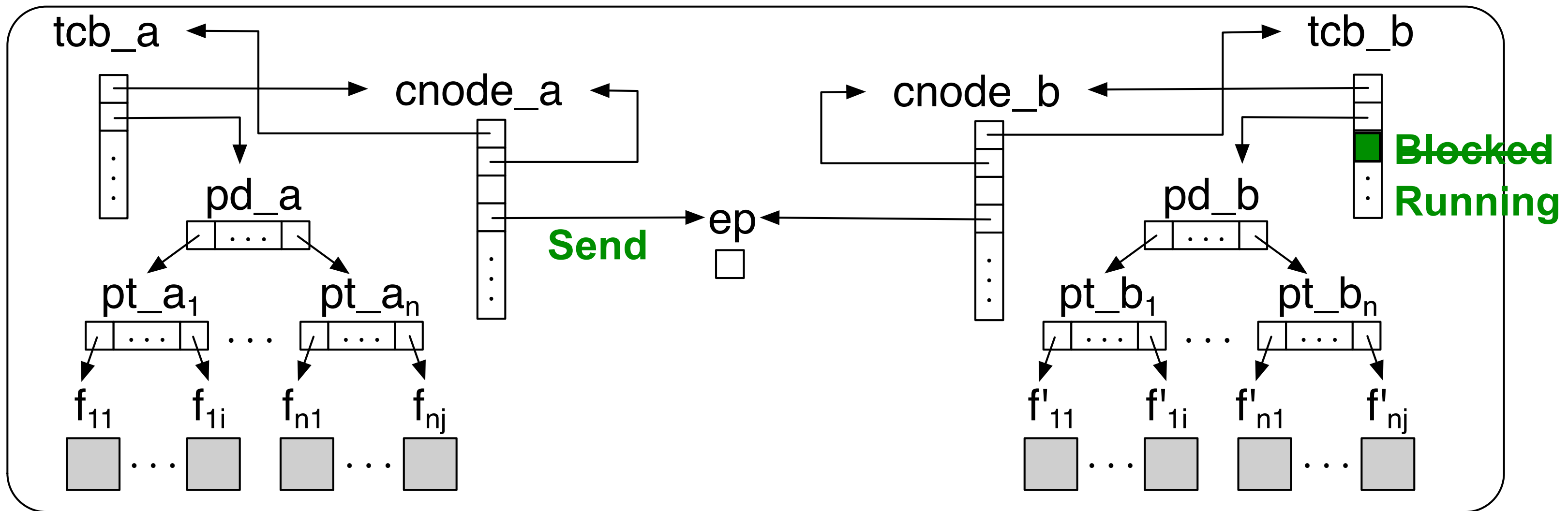
Then in which condition may **tcb_b** change

and what is allowed to change?

NICTA Copyright 2010

Example

If ***op*** is ***set_thread_state tcb_b v***
If **tcb_a** is running in state **s** where **s** is:



<u>Obvious (but wrong) solution</u>:

only the thread-state field of **tcb_b** is allowed tc c ange
and only under the following conditi

- **tcb_a** has a cap t
- or **tcb** ... **tcb_b** is waiting on in state **s**
- or **tcb** ... he untyped region containing **tcb_b**, in state **s**
- or ...

Policy closely depends on state

Solution: Labelling

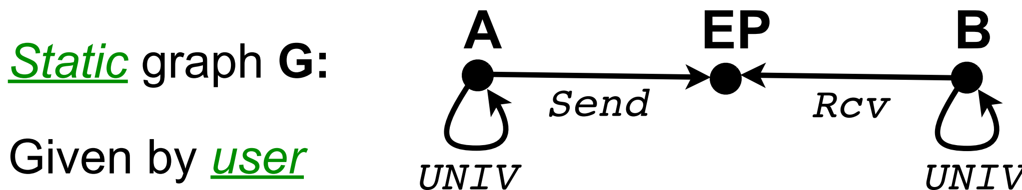
A EP B

tcb_a cnode_a pd_a pt_a_1 pt_a_n f_{11} f_{1i} f_{n1} f_{nj}

ep

cnode_b tcb_b pd_b pt_b_1 pt_b_n f'_{11} f'_{1i} f'_{n1} f'_{nj}

Static graph **G:**



Given by *user*

State must be *subset* a of **G**

G is *subjective*: current label contains the (untrusted) running thread

We prove:

1. **Graph preservation (authority confinement)**
2. **Access control at the label level**

NICTA Copyright 2010

Solution: Labelling

If **A** is the running label in **G**

then for any operation ***op*** that changes **s** to **s'**,

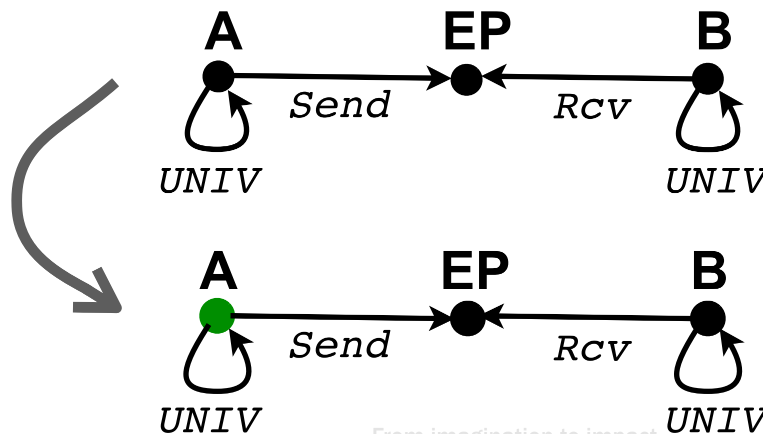
for any object **obj** of label **B**,

obj can only be changed if **A=B** or in 4 small precise cases, as:

"***obj*** *is a TCB blocked on an endpoint of label* ***EP****,*

*and (**A**,Send,**EP**)* ⊆ ***G***

and only the thread-state of ***obj*** *can be changed, to Running*"



We prove:

1. **Graph preservation (authority confinement)**
2. **Access control at the label level**

NICTA Copyright 2010 From imagination to impact

Agenda

NICTA

<u>Secure Systems</u> "à la NICTA"

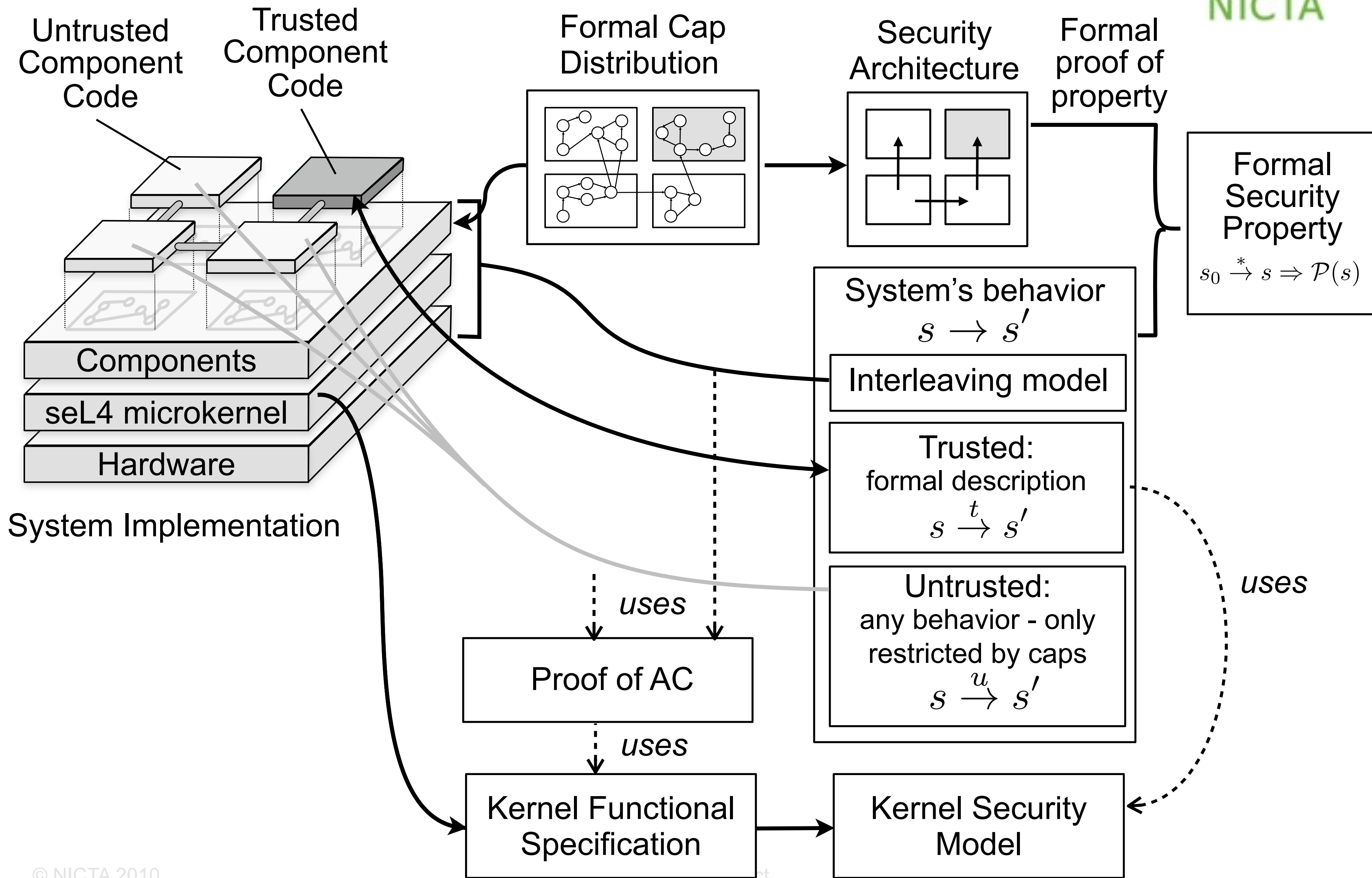
For 1 critical system:

- 1 desired security property P
- an interactive theorem prover
- a bit of patience

1. carefully design your system
2. prove that the design enforces P
3. prove correctness of the TCB
4. prove isolation

© NICTA 2010

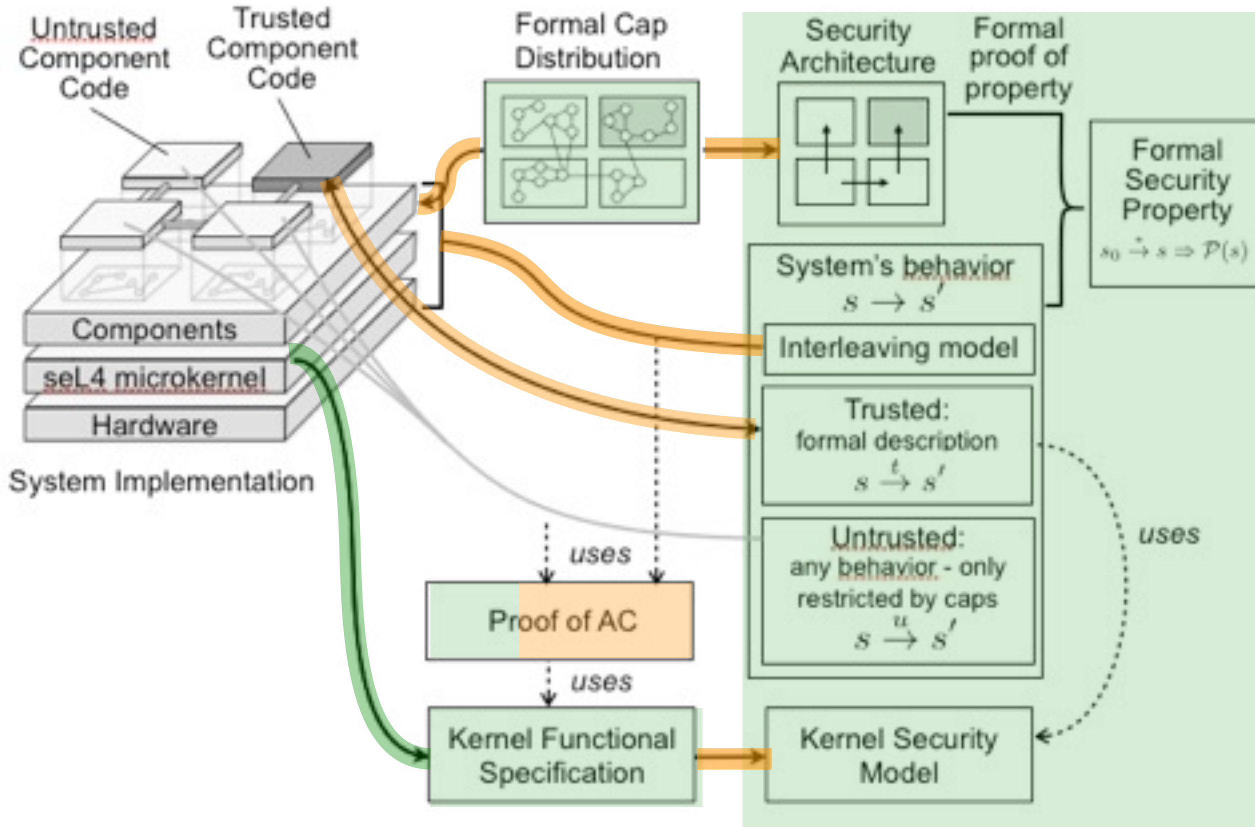
Conclusion



© NICTA 2010

Conclusion

NICTA



Results so far:

- Case study sec. proof
- seL4 enforces integrity
- capDL
- seL4 correctness proof
- certification: ST
- seL4 binary and formal spec released

Challenges:

Automation
verified code generator from high level code

Integration
system trace reasoning, concurrency

Confidentiality
preserved by refinement

Proof engineering
- refactoring
- efficient proof rerun

© NICTA 2010

Questions?