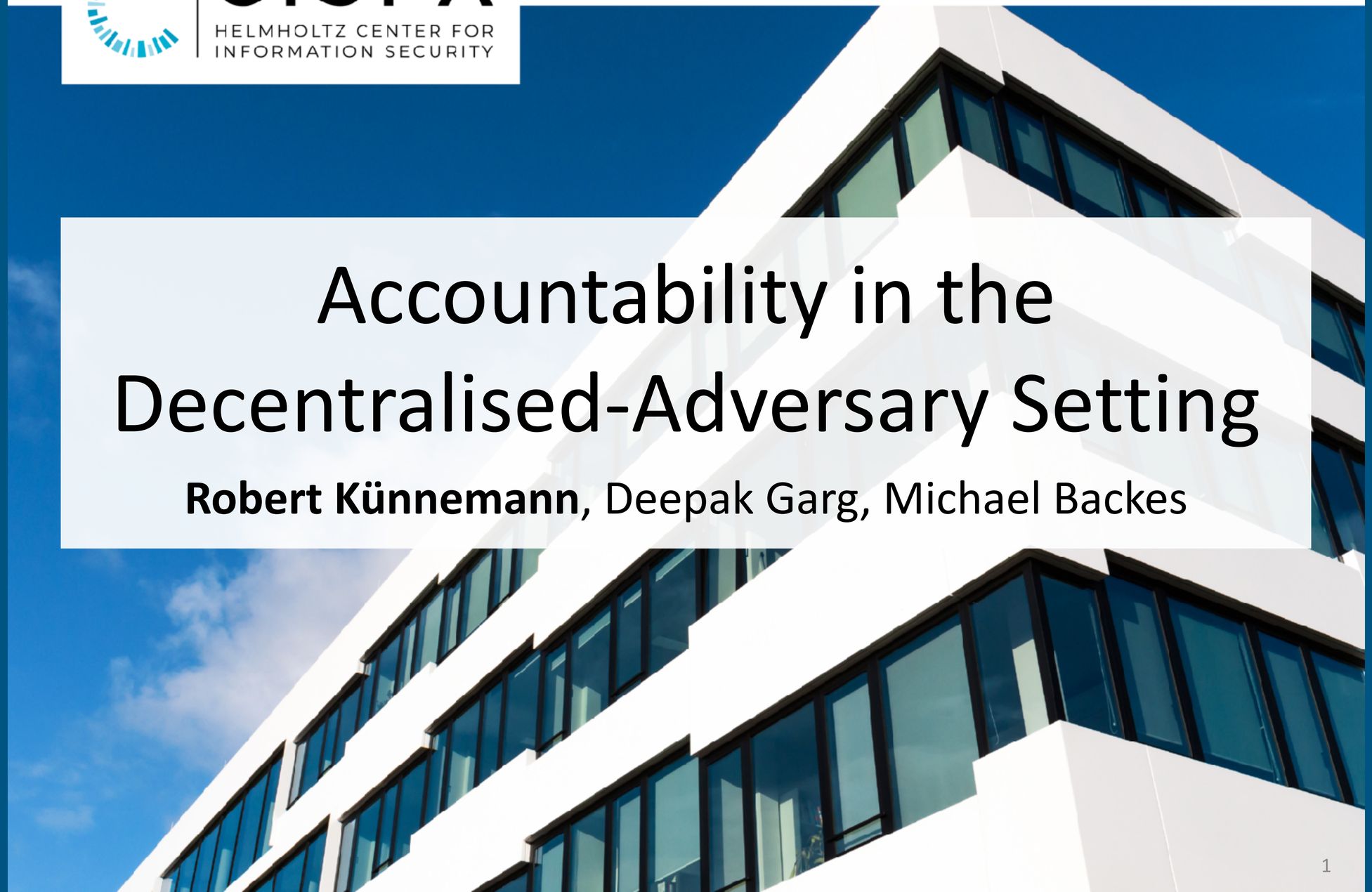
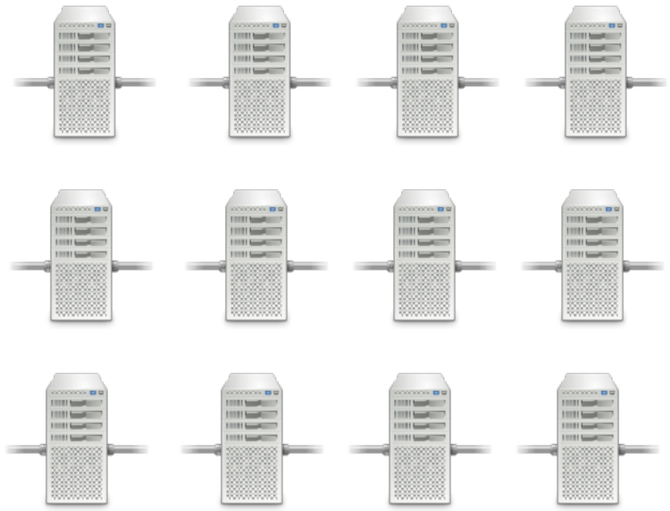
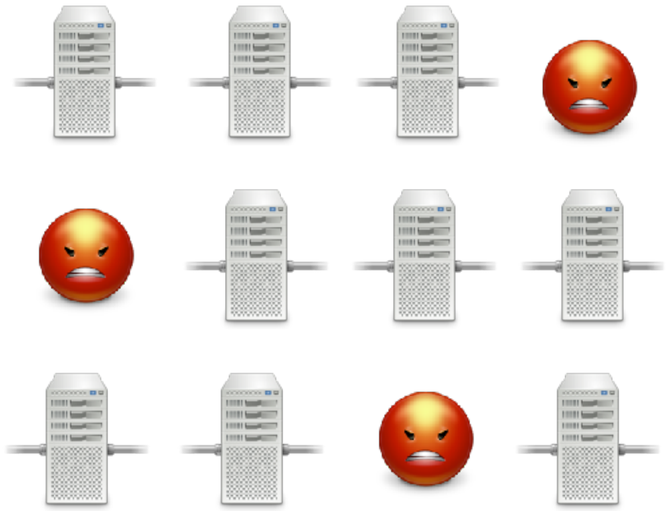


CISPA
HELMHOLTZ CENTER FOR INFORMATION SECURITY

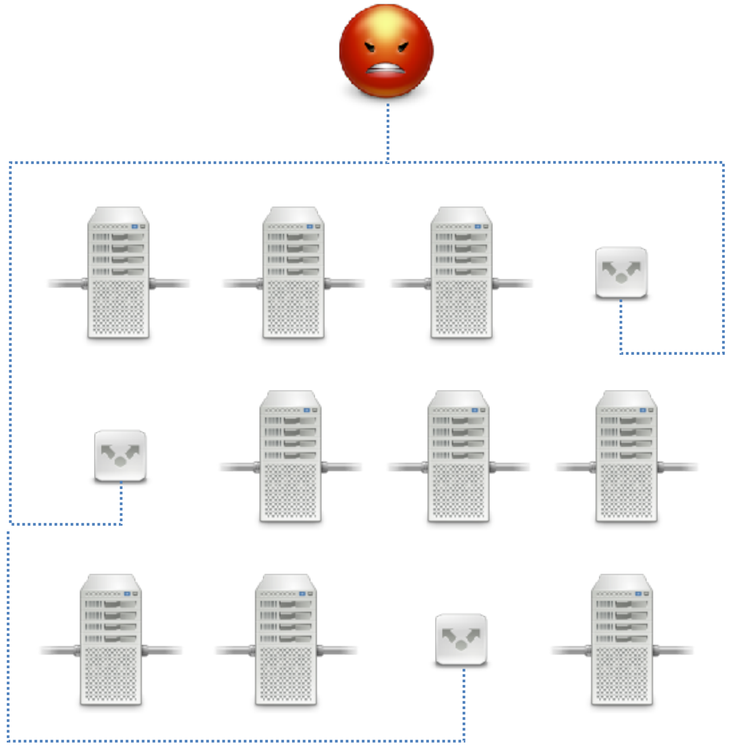
Accountability in the Decentralised-Adversary Setting

Robert Künnemann, Deepak Garg, Michael Backes





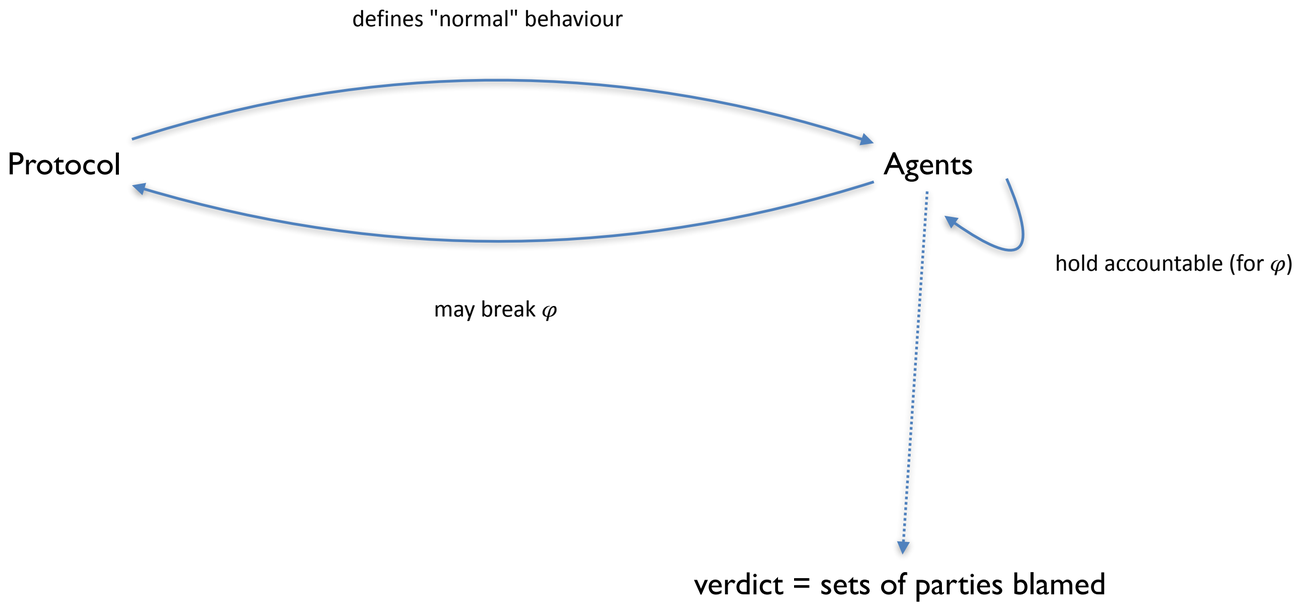
Decentralised Adversary



Centralised Adversary

What we talk about when we talk about accountability

Who Keeps Whom Accountable?



informs

?

Who Keeps Whom Accountable?

defines "normal" behaviour

- everybody who steps out of line?
  - Requires complete communication. It's the Internet, duh!
  - Benign mistakes happen. Moral problem, but also: bad implementations
- all causes (causing parties) of $\neg\varphi$

informs

verdict = sets of parties blamed

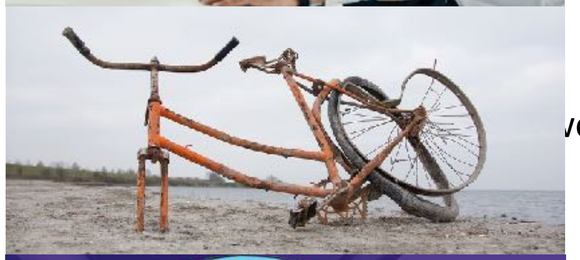
CORRECT!

From causation to accountability

A colleague asked for help

B ike broke down

C could not take the bus



happened

vork

Image src: shutterstock.com, pxhere.com, PBS wiki

From causation to accountability

A ran deviating program A'

B ran deviating program B'

C ran deviating program C'

L oss of authenticity

It works (in the centralised setting)

We can analyse that stuff (in the centralised model)

Protocol spec
+ accountability lemmas

(internal)

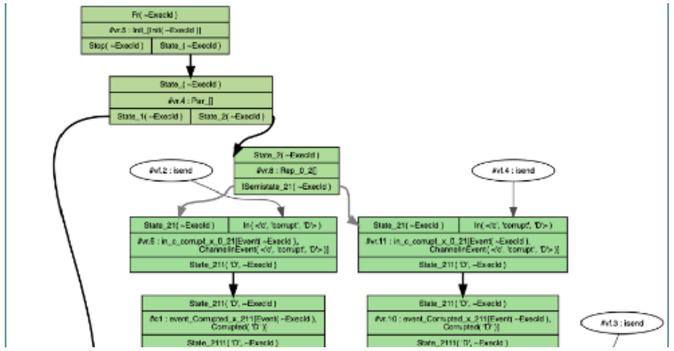
Protocol spec
+ trace property

tamarin-prover

attack / verification
/ timeout

```
/*****************************************
Accountability lemmas
*****************************************/

test mixer_evidence:
    "Ex sid m c0 c1 r1 #i. ma = <sid, m> & BlameMixer(sid, m, c0, c1, r1)@i"

lemma missing:
    mixer_evidence accounts for
    "All sid ms p1 p2 mp1 mp2 #i #j #k. Send(sid, ms)@i
                                & Post(<sid, p1>, '0', mp1)@j
                                & Post(<sid, p2>, '0', mp2)@k
                                & not(#j = #k)
                                ==> mp1 = ms | mp2 = ms"
```

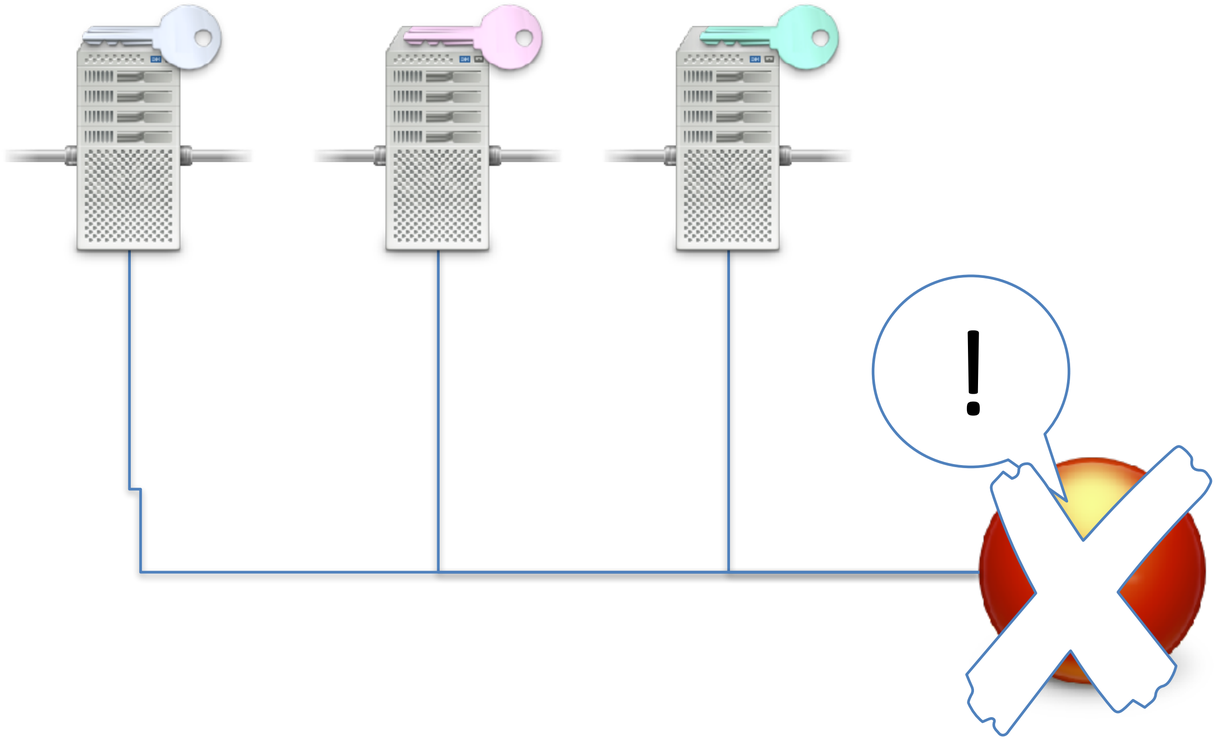
```
SpecialAct))))))))"
by sorry

lemma acc_suf_0:
  exists-trace
  "∀ #i #j ExecId1 ExecId2.
        ((Init( ExecId1 ) @ #i) ∧ (Init( ExecId2 ) @
#j)) ⇒ (#i = #j)) ∧
        ((∃ #i #j a.
          (Execute( a ) @ #i) ∧
          ((LogD( a ) @ #j) ∧ (¬((a = NormalAct) ∨ (a
= SpecialAct))))) ∧
        (((∃ #c1. Corrupted( 'D' ) @ #c1) ∧
          (∀ #c0. (Corrupted( 'C' ) @ #c0) ⇒ (#1)) ∧
(*))) ∧
          (¬(∀ #i a.
            (Execute( a ) @ #i) ⇒
            ((a = SpecialAct) ∨ (a =
NormalAct))))))"
simplify
solve( State_121111( ~ExecId, a, mSign ) ▶₀ #j )
  case event_Control_0_1_12111
  solve( State_211( 'D', ~ExecId ) ▶₀ #c1 )
    case in_c_corrupt_x_0_21
    solve( Execute( a ) @ #i )
```



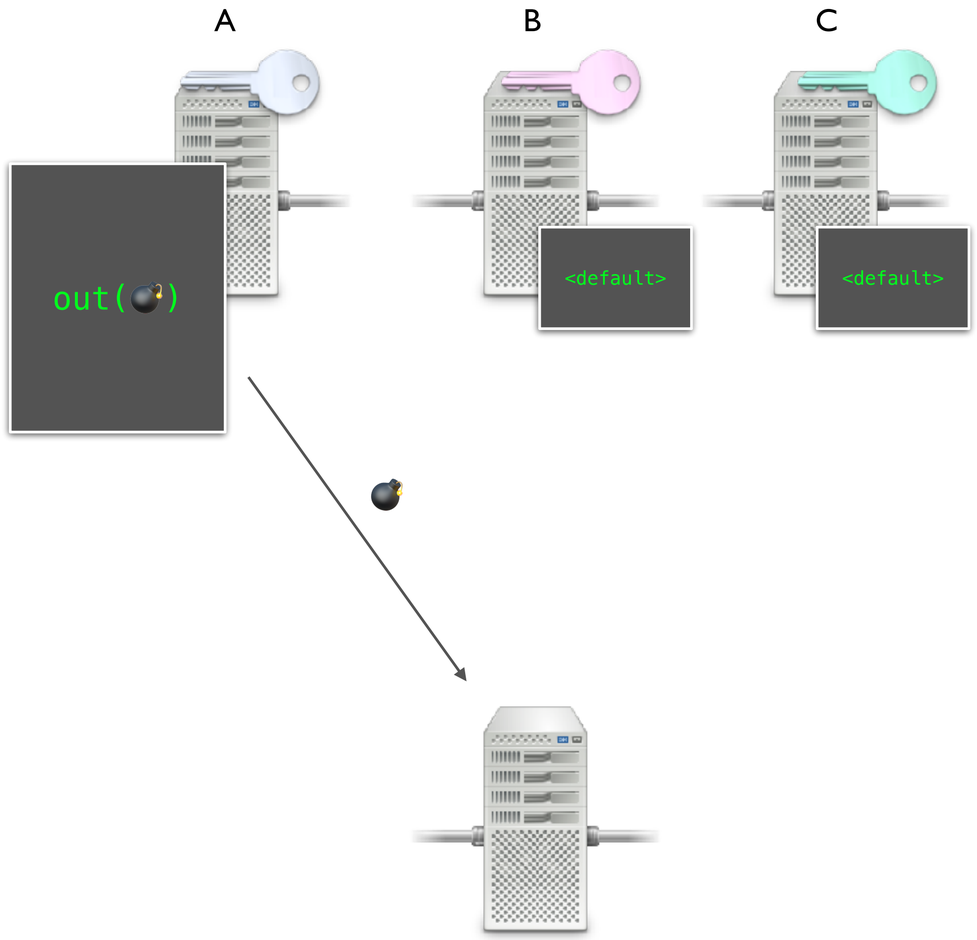
- Certificate Transparency
- OCSP Stapling
- MixNets
- Alethea/MixVote
- Accountable Algorithms

Limits of the centralised-adversary setting

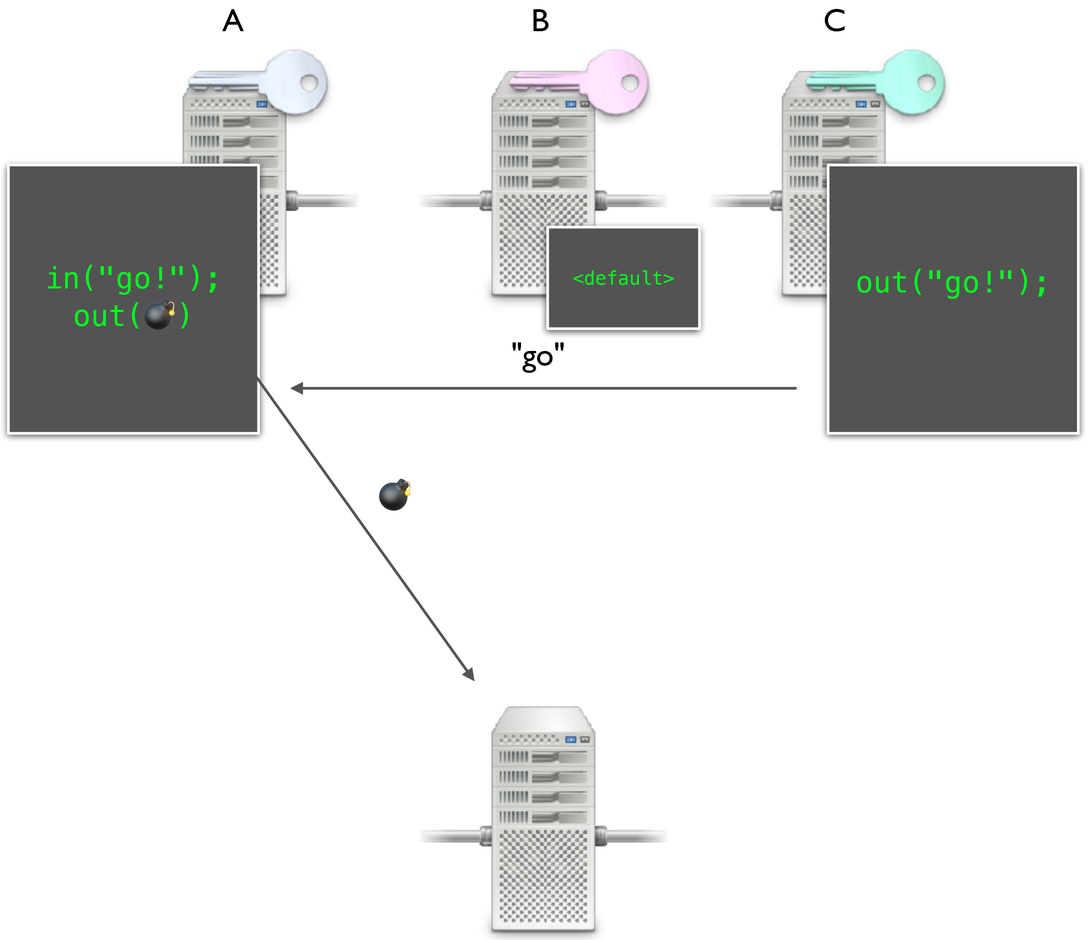
The centralised adversary



Provocation - scenario 1

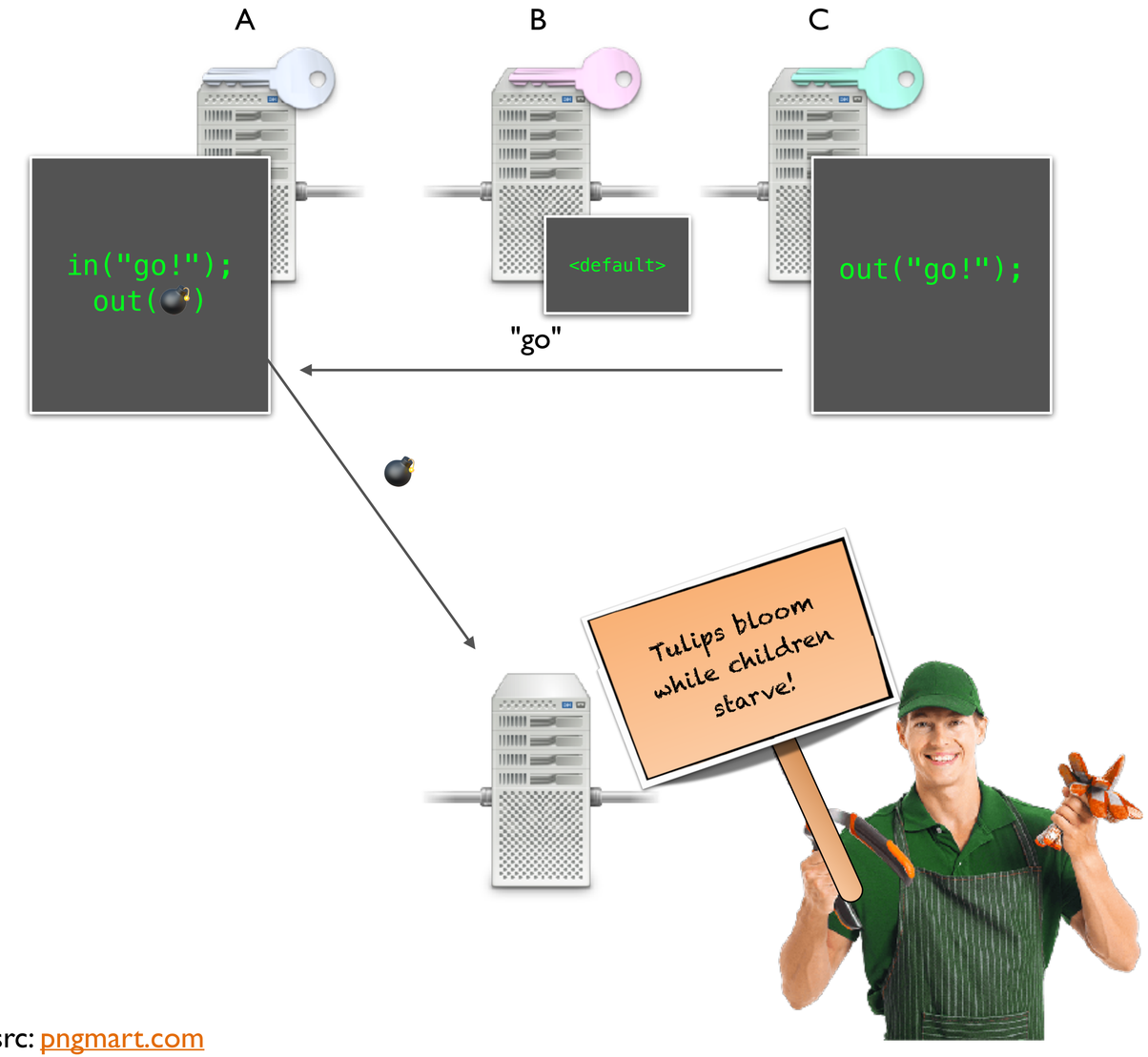


Provocation - scenario 2



- There is one cause, {A, C}.
- Anyone can derive "go!"
- Indistinguishable from A if
  - private communication possible
  - or code of A not known
- Not a modeling artefact

Provocation - scenario 2



- similar problems with causation in general (The Gardener & the Queen of England)
- causation considers different "worlds" and some are more plausible
- ordering of worlds
- "under constrained" (e.g. radical Gardener could despise all inedible flora)

img src: pngmart.com

Optimality

- **pick smallest possible verdict**:
  - logical entailment when verdict interpreted as DNF
    $\{\{A, B\}, \{C\}\} = A \wedge B \vee C$
  - {{A}} < {{A,C}} because $A \wedge C \implies A$

- **pick knowledge-optimal *explanations***, i.e., code for deviating parties
  - if A has knowledge to produce 💣, scenario 1 is knowledge-optimal

- **pick simple explanations**
  - includes knowledge-optimal
  - code cannot have conditionals (because we cannot see their effect)

fairness: all blamed parties cause violation

completeness: all parties causing violation are blamed

"the real deal"

full accountability

all communication must be visible

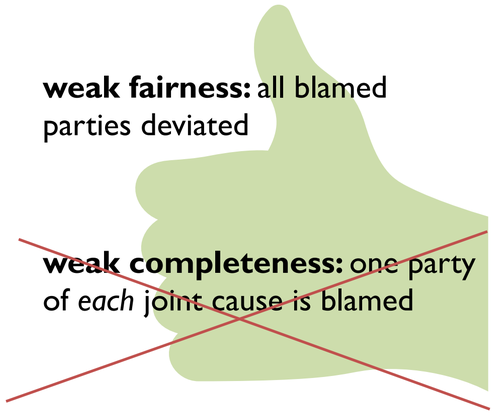
weak fairness: all blamed parties deviated

weak completeness: one party of *each* joint cause is blamed

"try to be specific!"

verdict-optimal accountability

verdicts with intersections (e.g., {A,B}, {B,C})
impossible



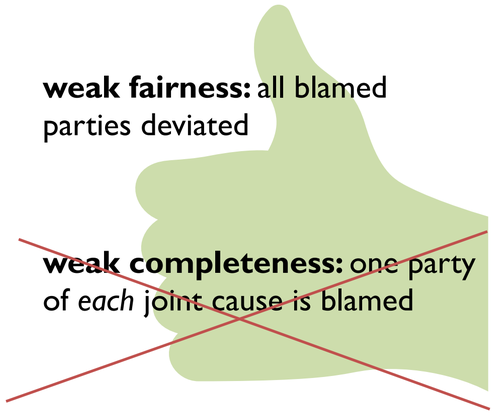
weak fairness: all blamed parties deviated

weak completeness: one party of *each* joint cause is blamed

"assume minimal information sharing"

knowledge-optimal accountability

either **verdicts always non-intersecting** or **no indirect communication**



weak fairness: all blamed parties deviated

~~**weak completeness:** one party of *each* joint cause is blamed~~

"minimal information sharing + no conditionals"

simple accountability

=

accountability in the centralised-adversary setting

Conclusion

Conclusion

- Accountability is identifying misbehaving parties
- "misbehaving party" = "party whose deviation caused $\neg\varphi$"
- the centralised setting is not w.l.o.g.:
  - silent assumptions: optimal information sharing and linear programs
  - guaranteed: weak fairness (party that is blamed deviated)
  - not guaranteed: weak completeness (catch member of each cause)
- verdict-optimality:
  - provides weak completeness
  - applicable for tasks like access control, randomness generation or holding a third party accountable
- all separating examples rely on signalling behavior unrelated to protocol
  - maybe optimality principle is adequate (Occam's razor, optimality & defaults in causation)
  - at least we know what we are doing now


Thank you!