How model checking can help

...build trust in the design of distributed protocols like Single Slot Finality

Igor Konnov
Independent security researcher

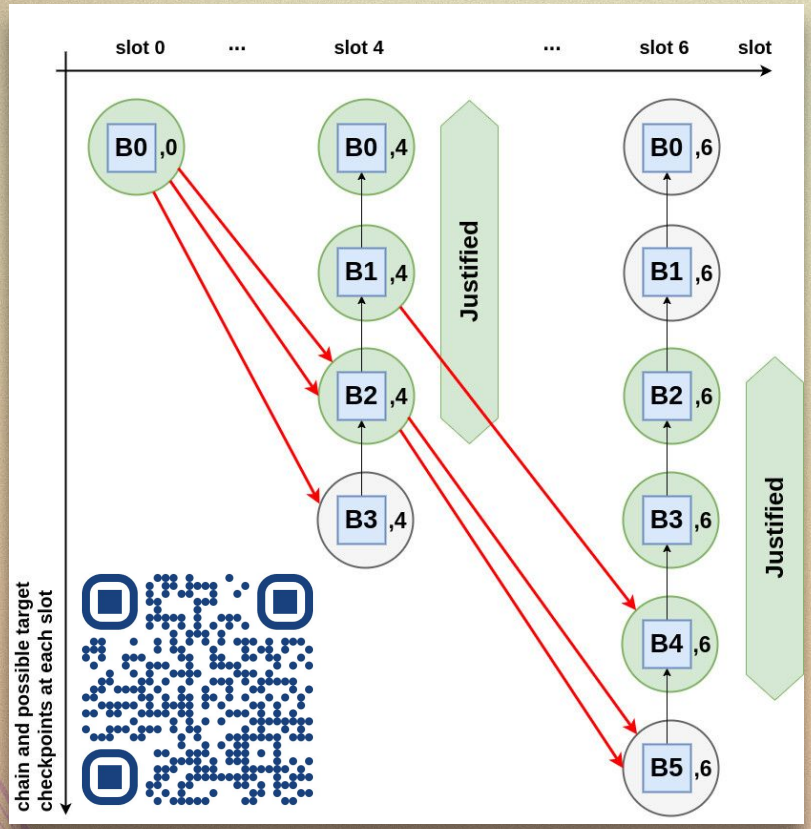
Thanh Hai Tran
Independent researcher

Jure Kukovec
Independent security researcher

Thomas Pani
Independent security researcher

Roberto Saltini
Independent researcher

Casper, Gasper, SSF, 3-Slot Finality



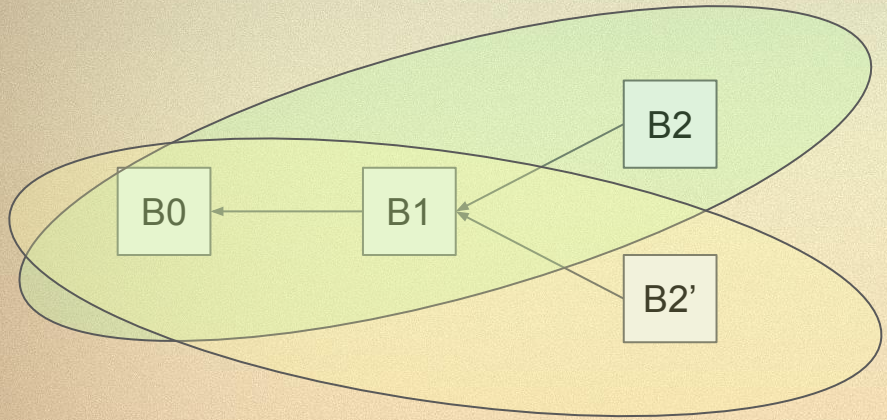
Source: https://arxiv.org/abs/2411.00558

Interplay of multiple concepts:

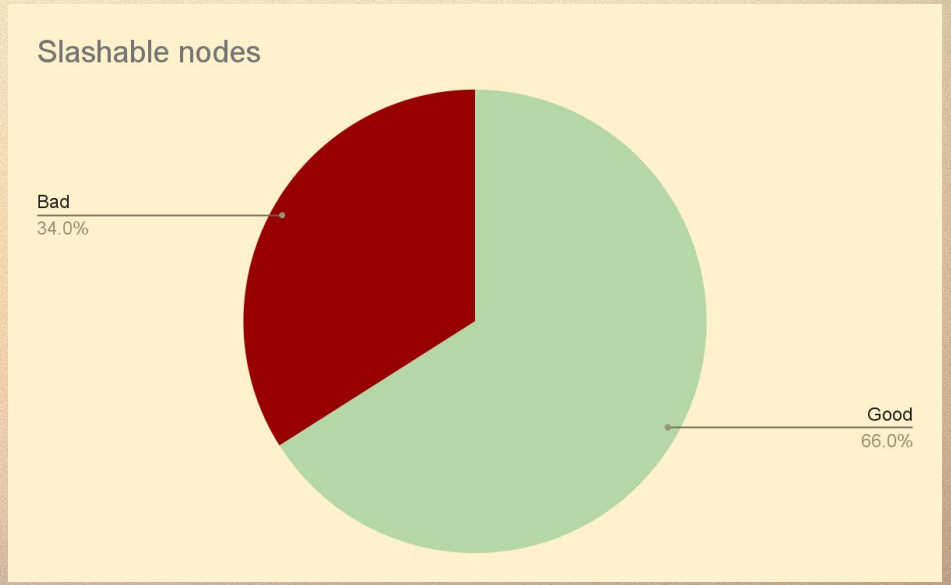
- chained blocks
- slots
- checkpoints: justified and finalized
- votes by validators
- FFG votes connecting checkpoints

Francesco D'Amato, Roberto Saltini, Thanh-Hai Tran, Luca Zanolini.
3-Slot-Finality Protocol for Ethereum

Focus on Accountable Safety



If there is a fork, we should identify N/3 slashable nodes



Ideal solution

Executable spec in Python

```
def get_slashabe_nodes(vote_view: PSet[SignedVoteMessage]) -> PSet[NodeIdentity]:
    return pset_map_to_pset(lambda vote: vote.sender,
                            pset_filter(lambda vote1: not pset_is_empty(pset_filter(lamb
```

Ideal solution

Executable spec in Python

```
def get_slashabe_
    return pset_ma
```

```
ntity]:

t_filter(lamb
```

No off-the-shelf solutions
(not even in sight)

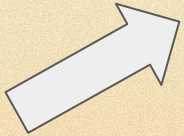
Executic
examples

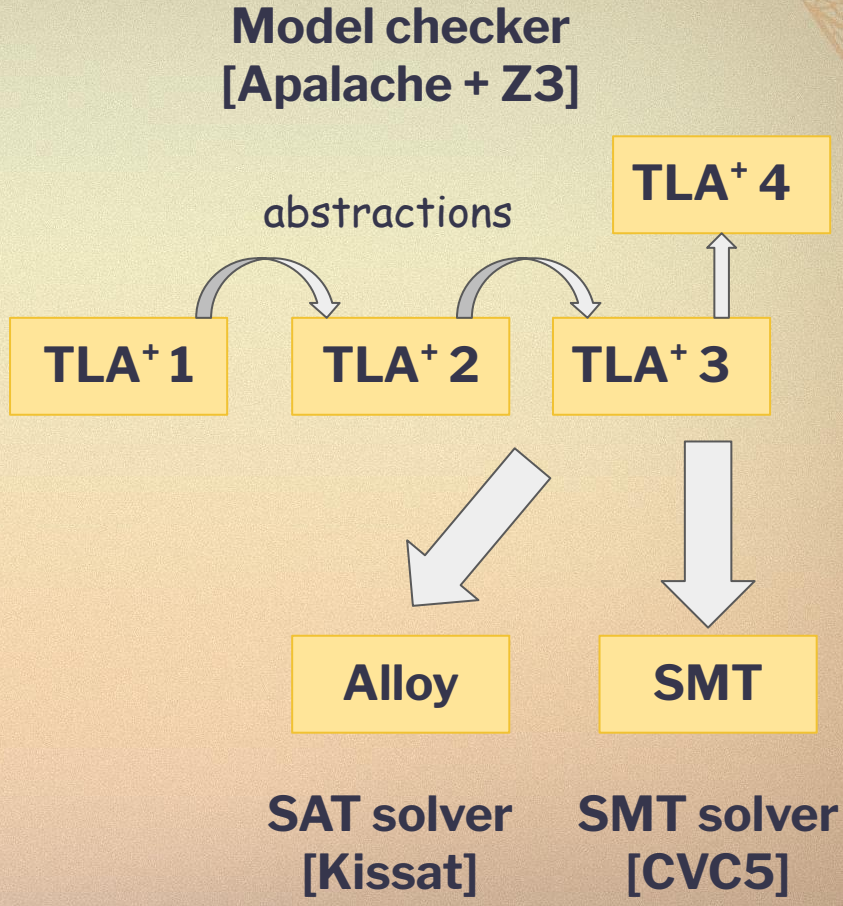
Automatic
proof of
Accountable safety

Our research

Executable spec in Python

```
def get_slashabe_nodes(vote_view: PS
    return pset_map_to_pset(lambda v
                            pset_fil
```





#1 Query for interesting states

Challenge the model checker with a false invariant

```
TwoFinalizedConflictingBlocks ≜
    LET disagreement ≜ ∃ c1, c2 ∈ justified_checkpoints:
        ∧ IsFinalized(c1, votes, justified_checkpoints)
        ∧ IsFinalized(c2, votes, justified_checkpoints)
        ∧ AreConflictingBlocks(c1[1], c2[1])
    IN ¬disagreement
```

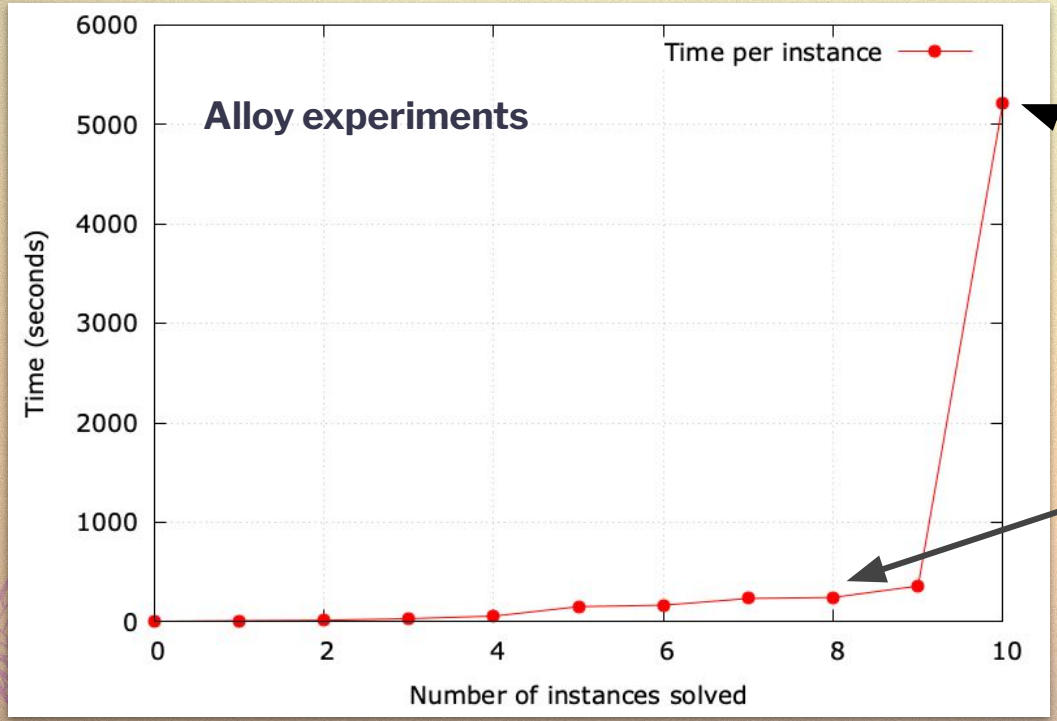
The model checker produces an example in 1.5 min

Communication tool with the protocol designers !

```
State6 ==
  all_blocks
      = { [body |-> -1, slot |-> 1],
          [body |-> 0, slot |-> 0],
          [body |-> 1, slot |-> 2] }
    /\ chain1 = { [body |-> 0, slot |-> 0], [body |-> 1, slot |-> 2] }
    /\ chain1_tip = [body |-> 1, slot |-> 2]
    /\ chain2 = { [body |-> -1, slot |-> 1], [body |-> 0, slot |-> 0] }
    /\ chain2_fork_block_number = -1
    /\ chain2_tip = [body |-> -1, slot |-> 1]
    /\ ffg_votes
      = { [source |-> <<[body |-> -1, slot |-> 1], 3>>,
           target |-> <<[body |-> -1, slot |-> 1], 4>>],
          [source |-> <<[body |-> 0, slot |-> 0], 0>>,
           target |-> <<[body |-> -1, slot |-> 1], 3>>],
          [source |-> <<[body |-> 0, slot |-> 0], 3>>,
           target |-> <<[body |-> 1, slot |-> 2], 4>>],
          [source |-> <<[body |-> 1, slot |-> 2], 4>>,
           target |-> <<[body |-> 1, slot |-> 2], 5>>] }
    /\ justified_checkpoints
      = { <<[body |-> -1, slot |-> 1], 3>>,
        <<[body |-> -1, slot |-> 1], 4>>,
        <<[body |-> 0, slot |-> 0], 0>>,
        <<[body |-> 0, slot |-> 0], 3>>,
        <<[body |-> 0, slot |-> 0], 4>>,
        <<[body |-> 1, slot |-> 2], 4>>,
        <<[body |-> 1, slot |-> 2], 5>> }
    /\ votes
      = { [ffg_vote |->
             [source |-> <<[body |-> -1, slot |-> 1], 3>>,
              target |-> <<[body |-> -1, slot |-> 1], 4>>],
           validator |-> "V1"],
          [ffg_vote |->
             [source |-> <<[body |-> -1, slot |-> 1], 3>>,
              target |-> <<[body |-> -1, slot |-> 1], 4>>],
```

#2 Show Acc. Safety

```
AccountableSafety ≜
    LET disagreement ≜ ∃ c1, c2 ∈ justified_checkpoints:
            ∧ IsFinalized(c1, votes, justified_checkpoints)
            ∧ IsFinalized(c2, votes, justified_checkpoints)
            ∧ AreConflictingBlocks(c1[1], c2[1])
    IN ¬disagreement ∨ Cardinality(SlashableNodes) * 3 ≥ N
```



5 blocks
7 checkpoints
24 votes

5 blocks
6 checkpoints
15 votes

Summary

Model checking helps ✅

Human ingenuity to tackle verification complexity (towers of NP?)

Tune in for the upcoming full technical report

Igor Konnov
igor@konnov.phd
x/twitter: @k0nn0v

Thanh Hai Tran
thanhhai1302@gmail.com

Thank you!

Igor Konnov
igor@konnov.phd
x/twitter: @k0nn0v

Thanh Hai Tran
thanhhai1302@gmail.com