VERIFYING EQUIVALENCES OF FINITE PROCESSES

Vincent Cheval
Steve Kremer
Itsaka Rakotonirina

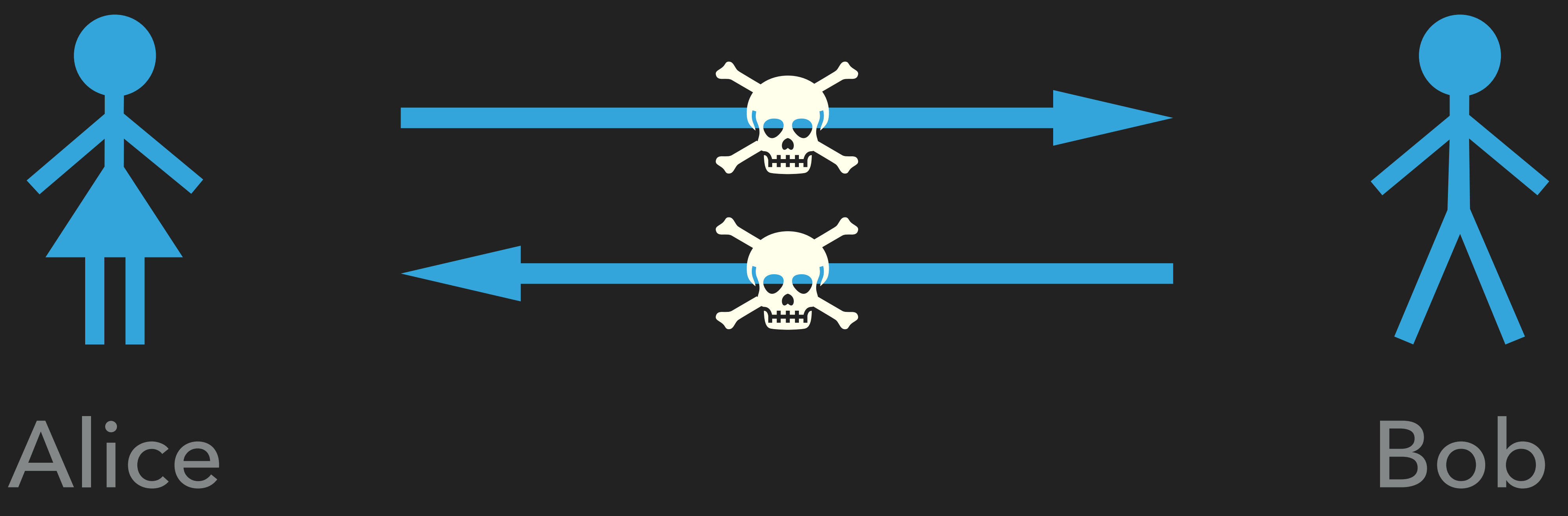
Channels under corruption

Channels under corruption

Security protocols should cope with corrupted channels

Worst case: messages are read by evil entities,
and are replaced by new ones or blocked

Channels under corruption



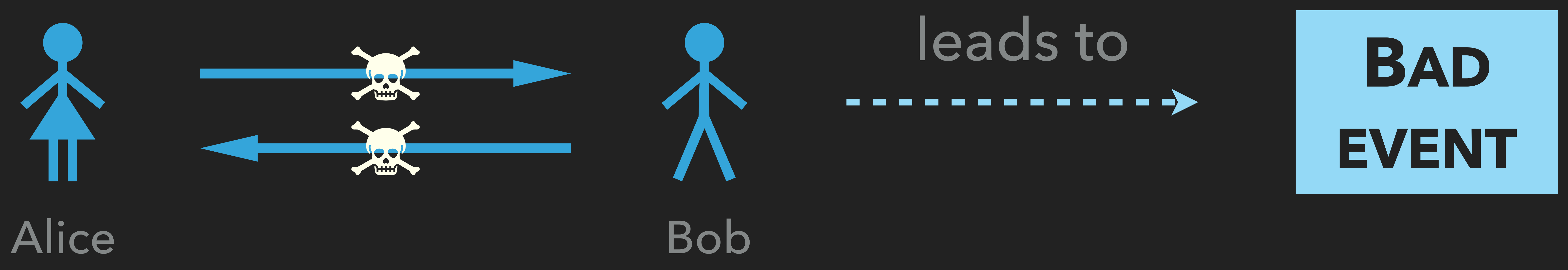
messages are read by evil entities,
and are replaced by new ones or blocked

Security properties shall hold
despite corrupted channels
(assuming perfect cryptography)

Examples

- secrecy (of sensible data)
- authentication (handshake)
- vote privacy (e-voting)

Security as reachability



By now well understood ✓

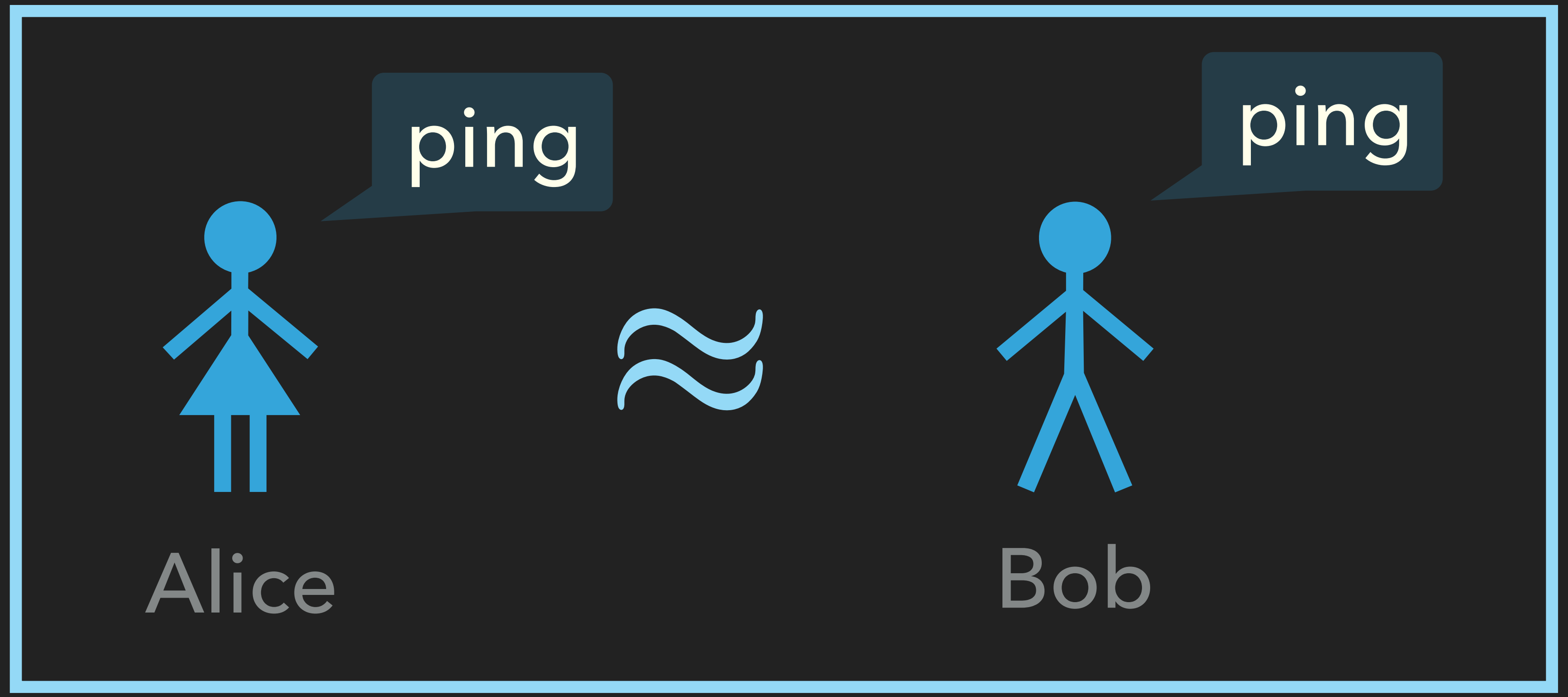
theoretical understanding of the problem (complexity results) and mature automated analysers

Security as equivalence

Anonymity

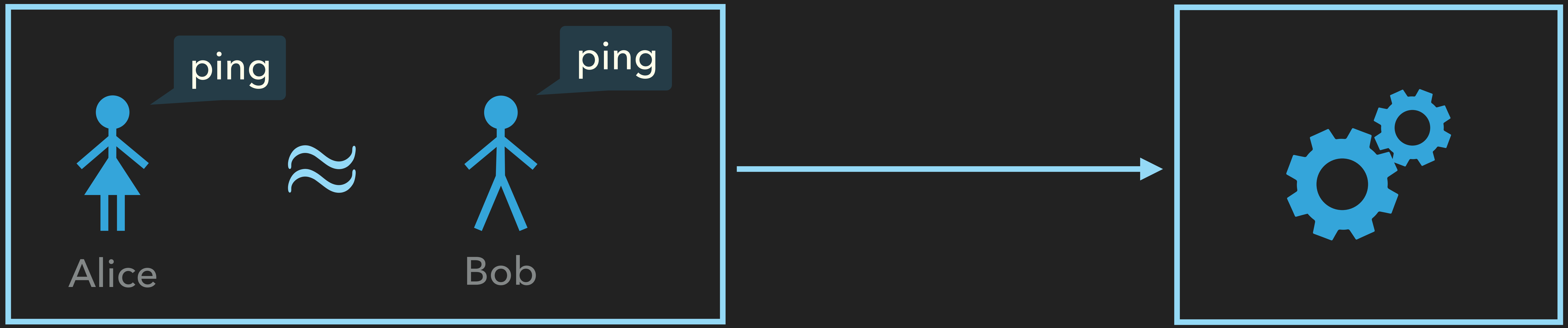
Unlinkability

Verifying equivalences: DEEPSEC



Description of the protocol

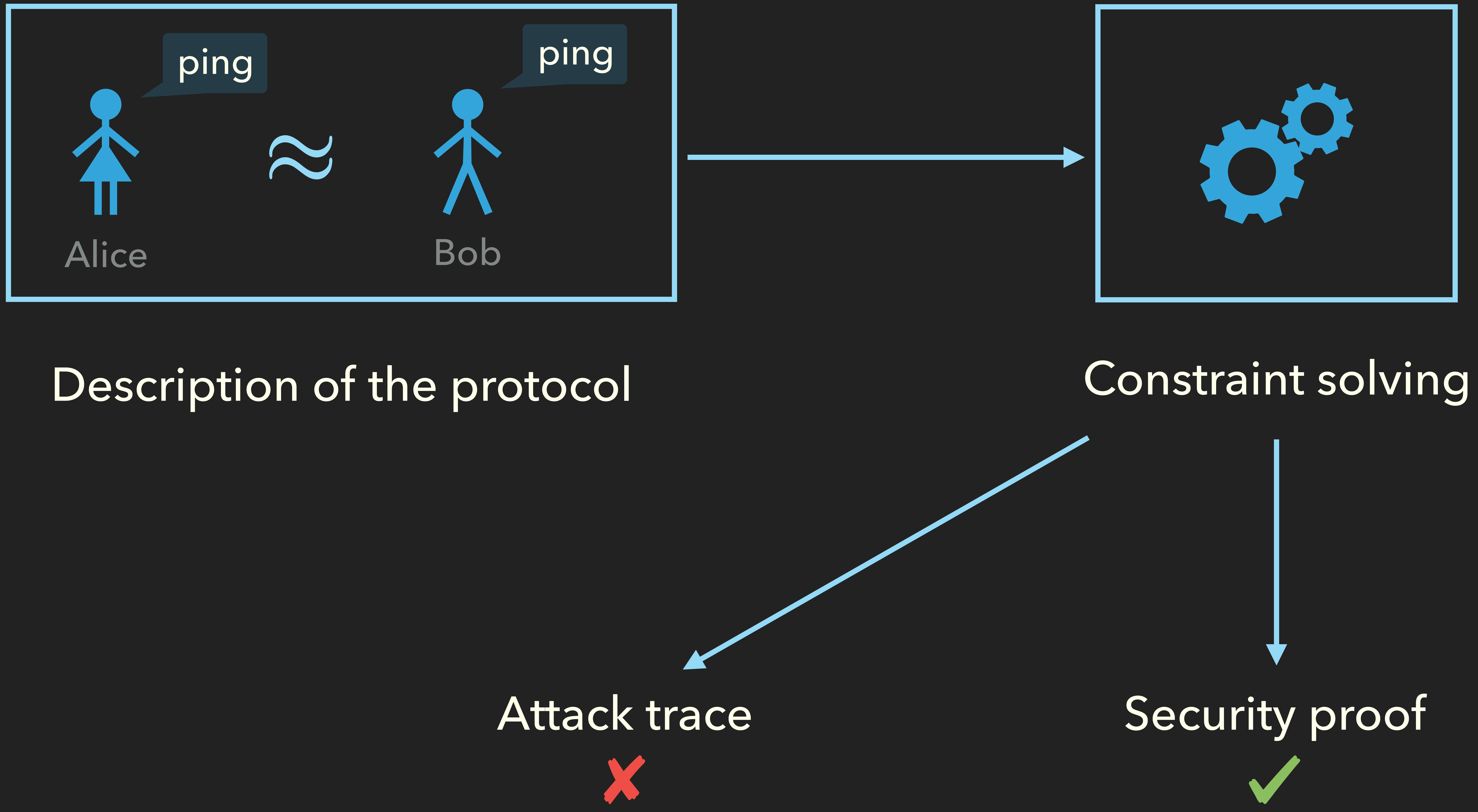
Verifying equivalences: DEEPSEC



Description of the protocol

Constraint solving

Verifying equivalences: DEEPSEC



A hard problem

Verification is **very** hard

Complexity results

(subterm convergent cryptographic primitives)

coNP-complete

with a passive attacker

coNEXP-complete

with an active attacker

Solutions ?

Restrictions

restrict the fragment, make sound approximations

Efficiency "in practice"

optimisations for realistic protocols

A hard problem

Verification is **very** hard

Solutions ?

Restrictions

restrict the fragment, make sound approximations

Efficiency "in practice"

optimisations for realistic protocols

Current work

A subequivalence harnessing symmetries between processes to speed-up security proofs