Symbolic Verification of Distance-bounding Protocols

Application to payments protocols

Alexandre Debant[1], Stéphanie Delaune[2], Cyrille Wiedling[3]

[1]Université de Lorraine, CNRS, Inria, LORIA,
[2]Univ Rennes, CNRS, IRISA,
[3]DGA-MI, Bruz

FM Seminar
April 6th 2021

Introduction

Scenario

Eve, a dishonest agent, wants to buy an item but without paying it!

Introduction

Scenario

Eve, a dishonest agent, wants to buy an item but without paying it!

Existing solutions

- Steal the item…

Introduction

Scenario

Eve, a dishonest agent, wants to buy an item but without paying it!

Existing solutions

- Steal the item…
- Steal a credit card: by-pass the PIN code, pay on the Internet, contactlessly…

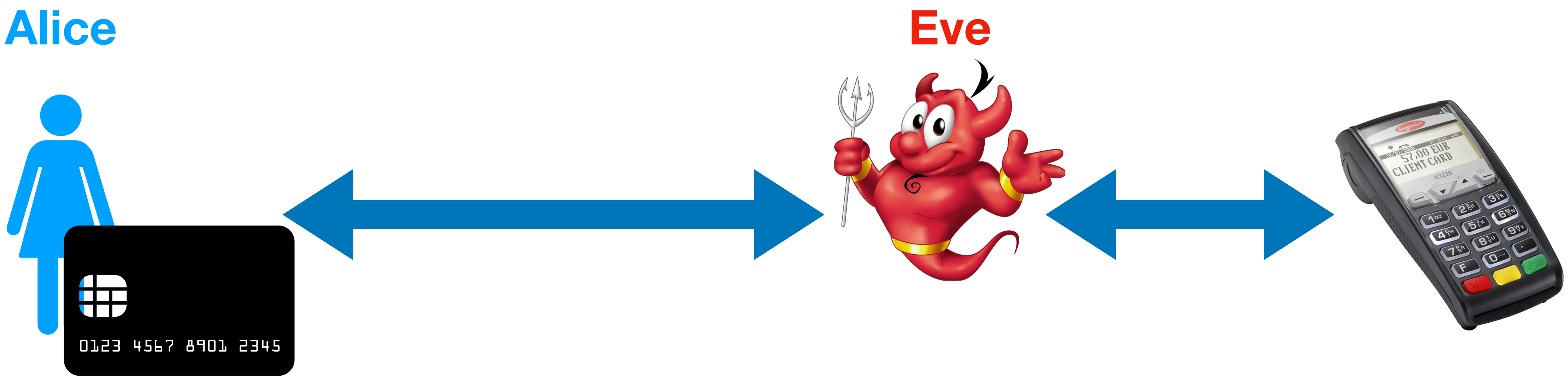
Introduction

Scenario

Eve, a dishonest agent, wants to buy an item but without paying it!

Existing solutions

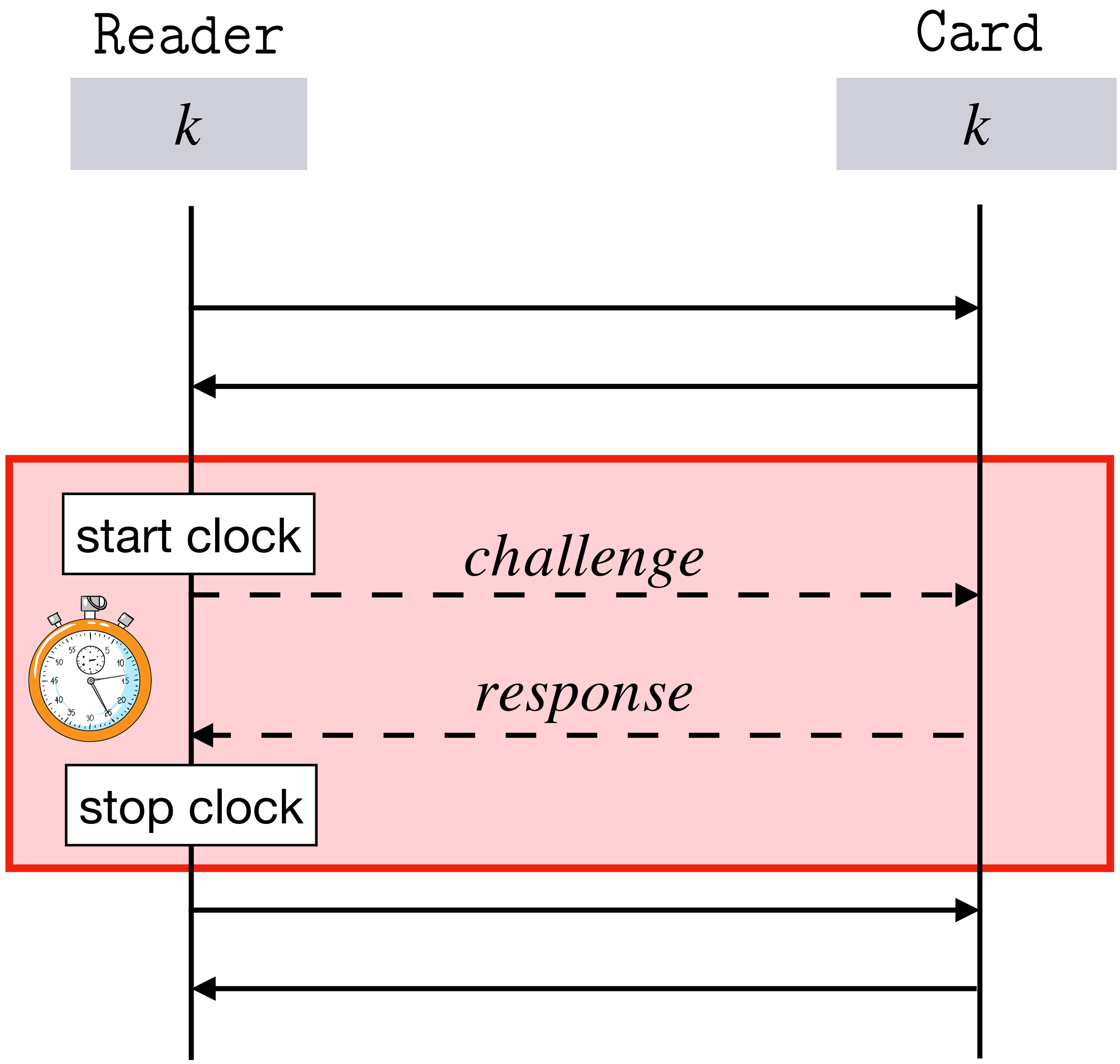
- Steal the item…
- Steal a credit card: by-pass the PIN code, pay on the Internet, contactlessly…
- Abuse a victim by relaying messages using the contactless technology



Proving the physical proximity

History of distance-bounding protocols

- First: Brands and Chaum protocol (1993)
- Today: more than 40 new protocols since 2003
- Application: in EMV's specification since 2016

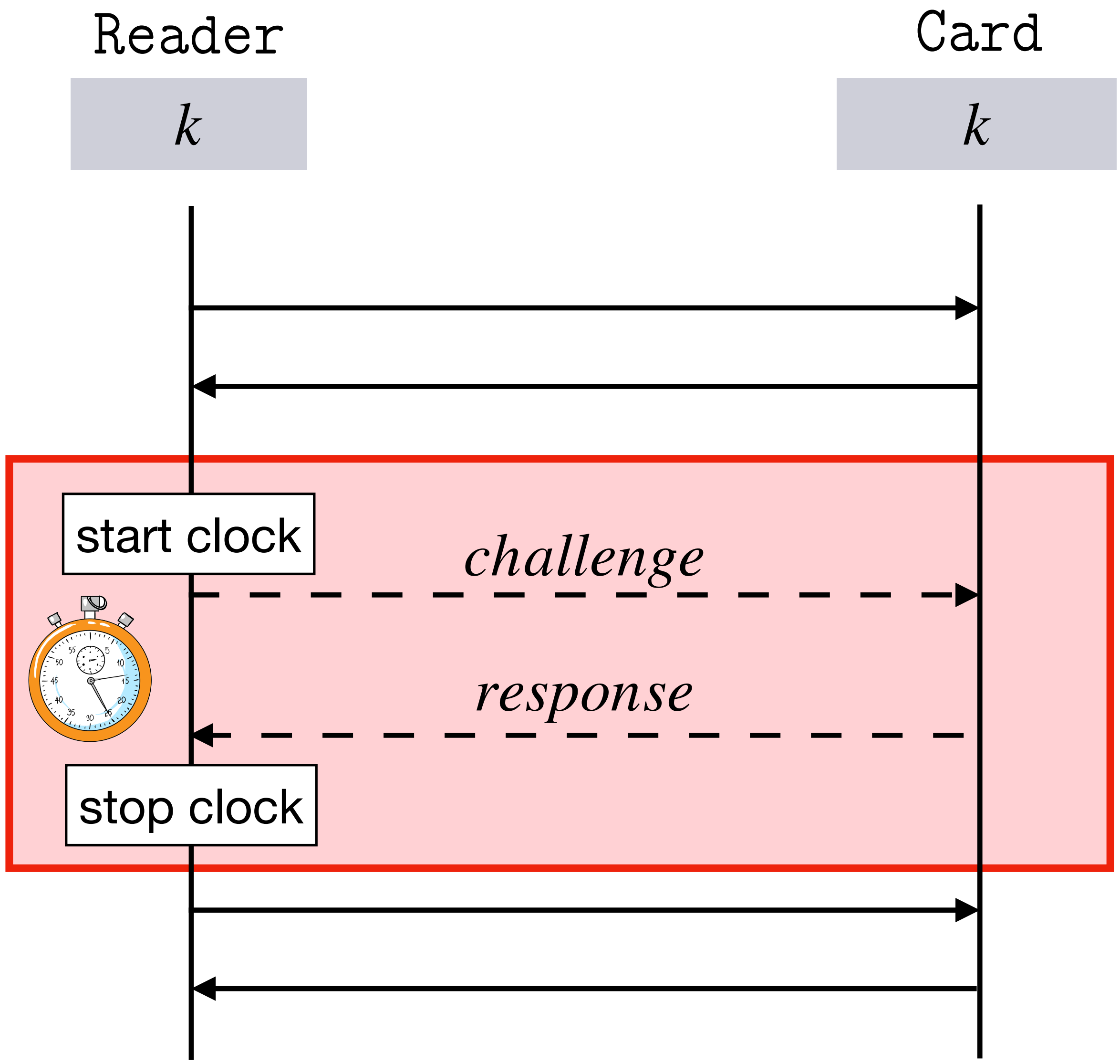


Proving the physical proximity

History of distance-bounding protocols

- First: Brands and Chaum protocol (1993)
- Today: more than 40 new protocols since 2003
- Application: in EMV's specification since 2016

Designing a good protocol is difficult!



Many applications

that are insecure….

[Chothia *et al.* - 2010]

Passport

[Vanhoef *et al.* - 2017]

Wi-Fi

[Nohl *et al.* - 2008]

Transport ticketing

[Murdoch *et al.* - 2010]

Credit card

Two major families of models…

… with some advantages and some drawbacks.

Computational models

+ messages are bitstrings, a general and powerful attacker

– tedious proofs, sometimes mechanized, but often for experts only

Symbolic models

– Some simplifications/abstractions (messages, attacker…)

+ procedures and automated tools

Some results make a link between these two models
[Abadi & Rogaway - 2000]

Symbolic verification in a nutshell

Messages

- Function symbols: $\texttt{enc}(x, k)$, $\texttt{sign}(x, k)$, $\texttt{h}(x)$,…
- Equations: $\texttt{dec}(\texttt{enc}(x, k), k) = x$

Perfect cryptography

Protocols

- Process algebra, multiset rewriting rules, Horn clauses…

The attacker can…

- read / overwrite messages
- intercept / block messages

The attacker cannot…

- break crypto
- use side-channels

Existing verification tools

Bounded number of sessions

- decidable for classes of protocols
- tools implement decision procedures

DEEPSEC AKiSs

Existing verification tools

Bounded number of sessions

- decidable for classes of protocols
- tools implement decision procedures

DEEPSEC

AKiSs

Unbounded number of sessions

- undecidable in general
- efficient tools in practice but:
  - do some approximations
  - may not terminate

ProVerif

TAMARIN
Tamarin prover interactive mode

Existing verification tools

Bounded number of sessions

- decidable for classes of protocols
- tools implement decision procedures

DEEPSEC

AKiSs

Unbounded number of sessions

- undecidable in general
- efficient tools in practice but:
  - do some approximations
  - may not terminate

ProVerif

TAMARIN
Tamarin prover interactive mode

5G-AKA

Belenios e-voting

Proving the physical proximity

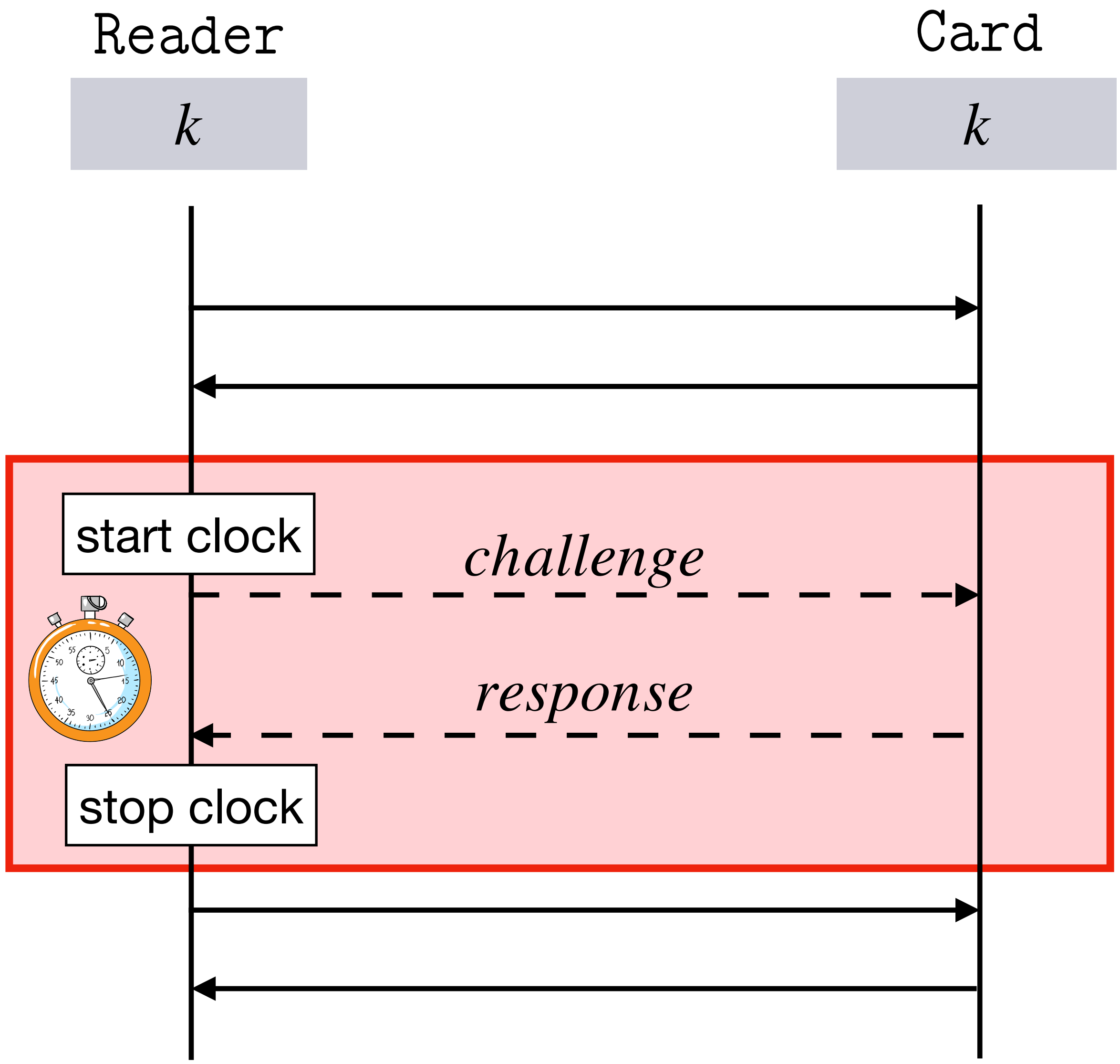
History of distance-bounding protocols

- First: Brands and Chaum protocol (1993)
- Today: more than 40 new protocols since 2003
- Application: in EMV's specification since 2016

Related work in symbolic verification

- Standard models and tools: do not model time and locations!
- Main specific models:
  - Meadows *et al.* (2007),
  - Basin *et al.* (2011)

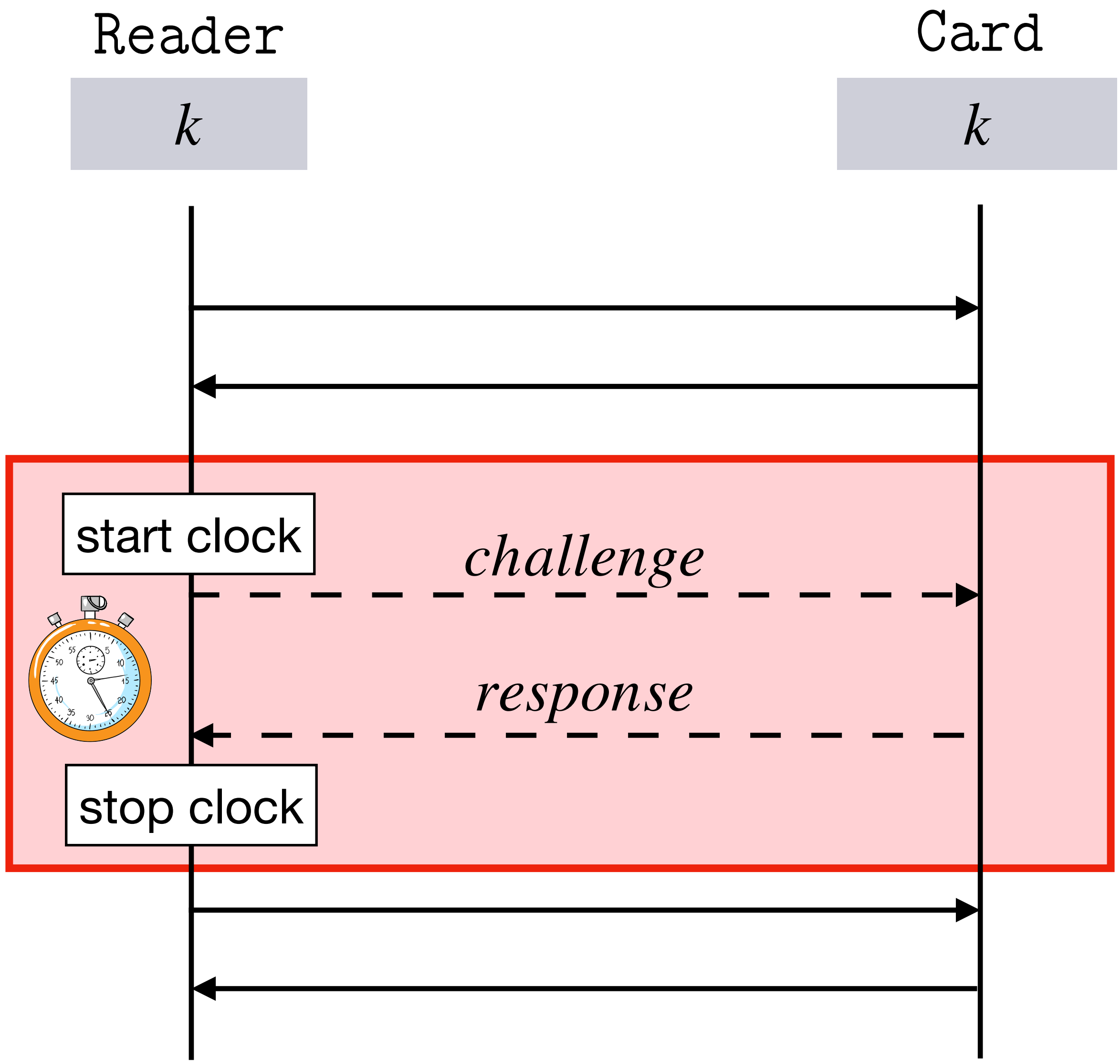
➡ no automated verification procedure…



Proving the physical proximity

History of distance-bounding protocols

- First: Brands and Chaum protocol (1993)
- Today: more than 40 new protocols since 2003
- Application: in EMV's specification since 2016



Related work in symbolic verification

- Standard models and tools: do not model time and locations!
- Main specific models:
  - Meadows *et al.* (2007),
  - Basin *et al.* (2011)

➡ no automated verification procedure…

Can we design a framework that allows for a **fully automated** verification?

Proving the physical proximity

History of distance-bounding protocols

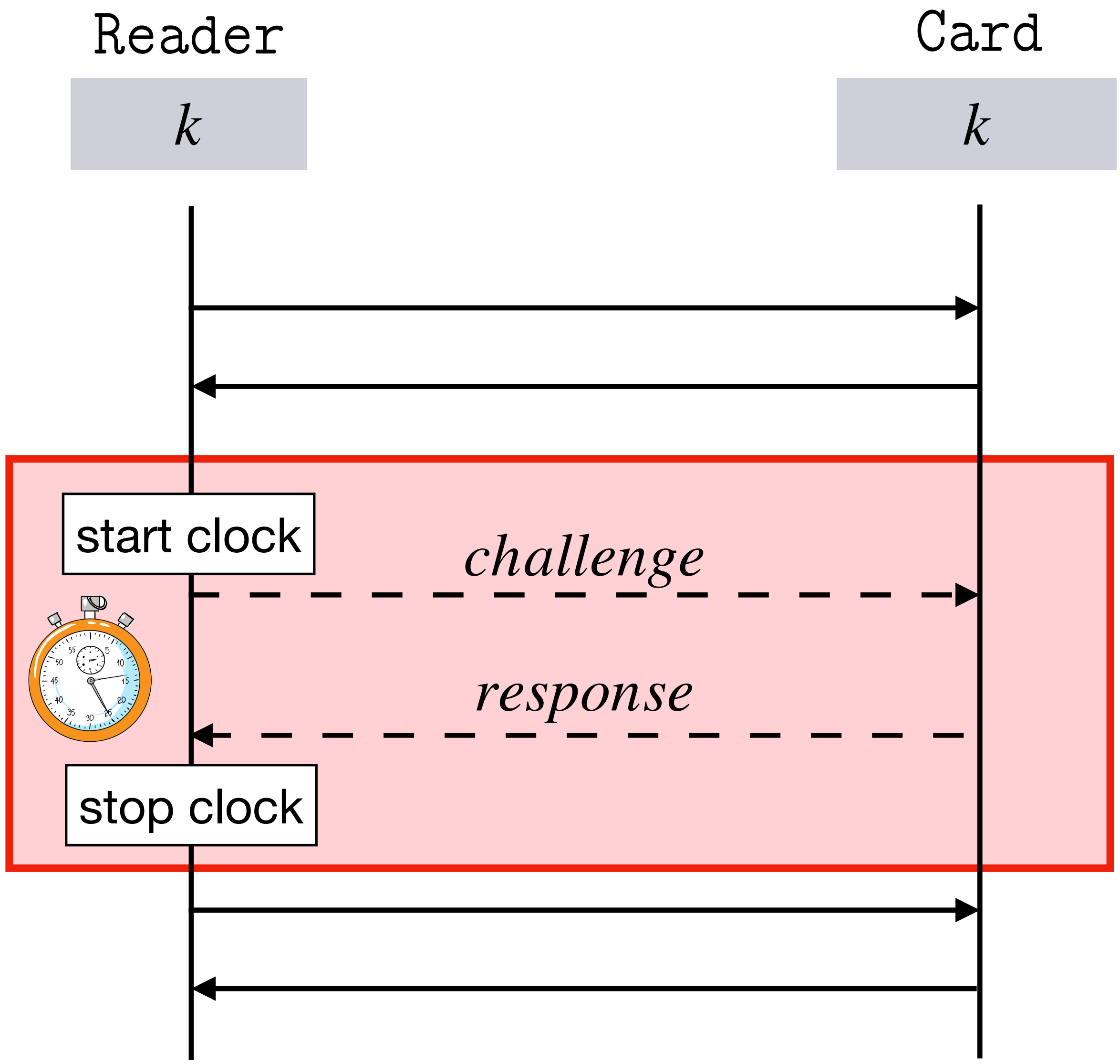
- First: Brands and Chaum protocol (1993)
- Today: more than 40 new protocols since 2003
- Application: in EMV's specification since 2016

Related work in symbolic verification

- Standard models and tools: do not model time and locations!
- Main specific models:
  - Meadows *et al.* (2007),
  - Basin *et al.* (2011)

➡ no automated verification procedure…

- Recently: Mauw et. al. (2018, 2019)



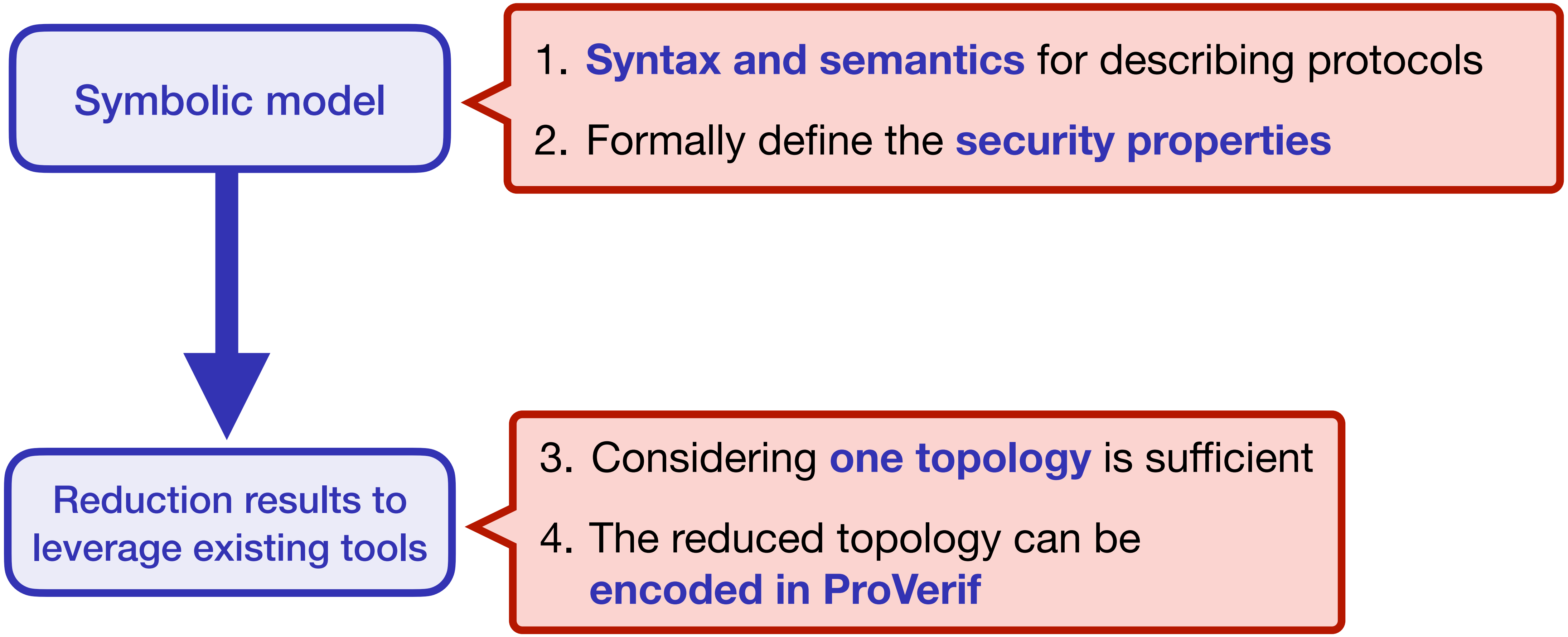
Can we design a framework that allows for a fully automated verification?

The story of verification

Symbolic model

1. **Syntax and semantics** for describing protocols
2. Formally define the **security properties**

The story of verification



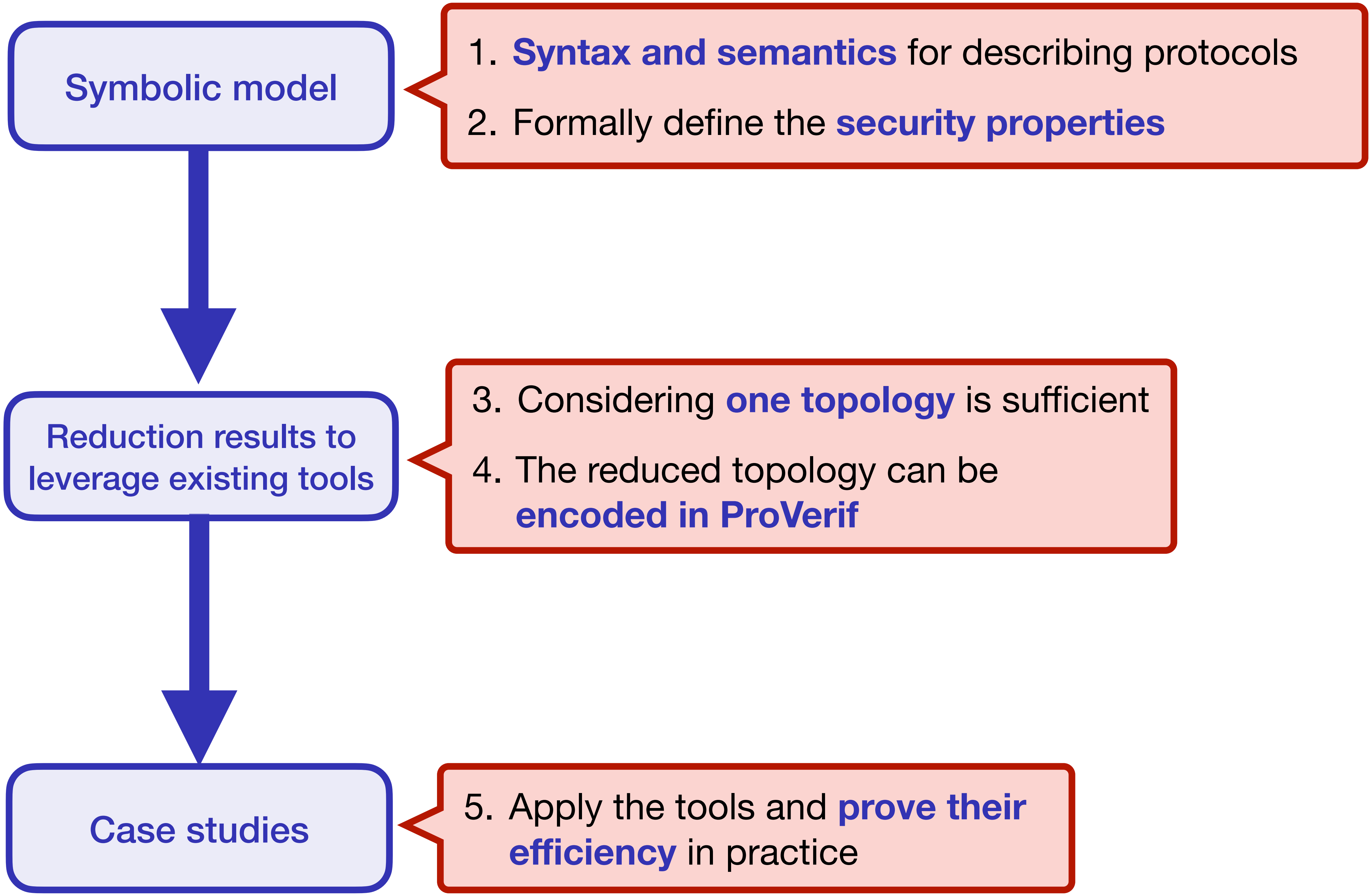
Symbolic model

1. **Syntax and semantics** for describing protocols
2. Formally define the **security properties**

Reduction results to leverage existing tools

3. Considering **one topology** is sufficient
4. The reduced topology can be **encoded in ProVerif**

The story of verification



Symbolic model

1. **Syntax and semantics** for describing protocols
2. Formally define the **security properties**

Reduction results to leverage existing tools

3. Considering **one topology** is sufficient
4. The reduced topology can be **encoded in ProVerif**

Case studies

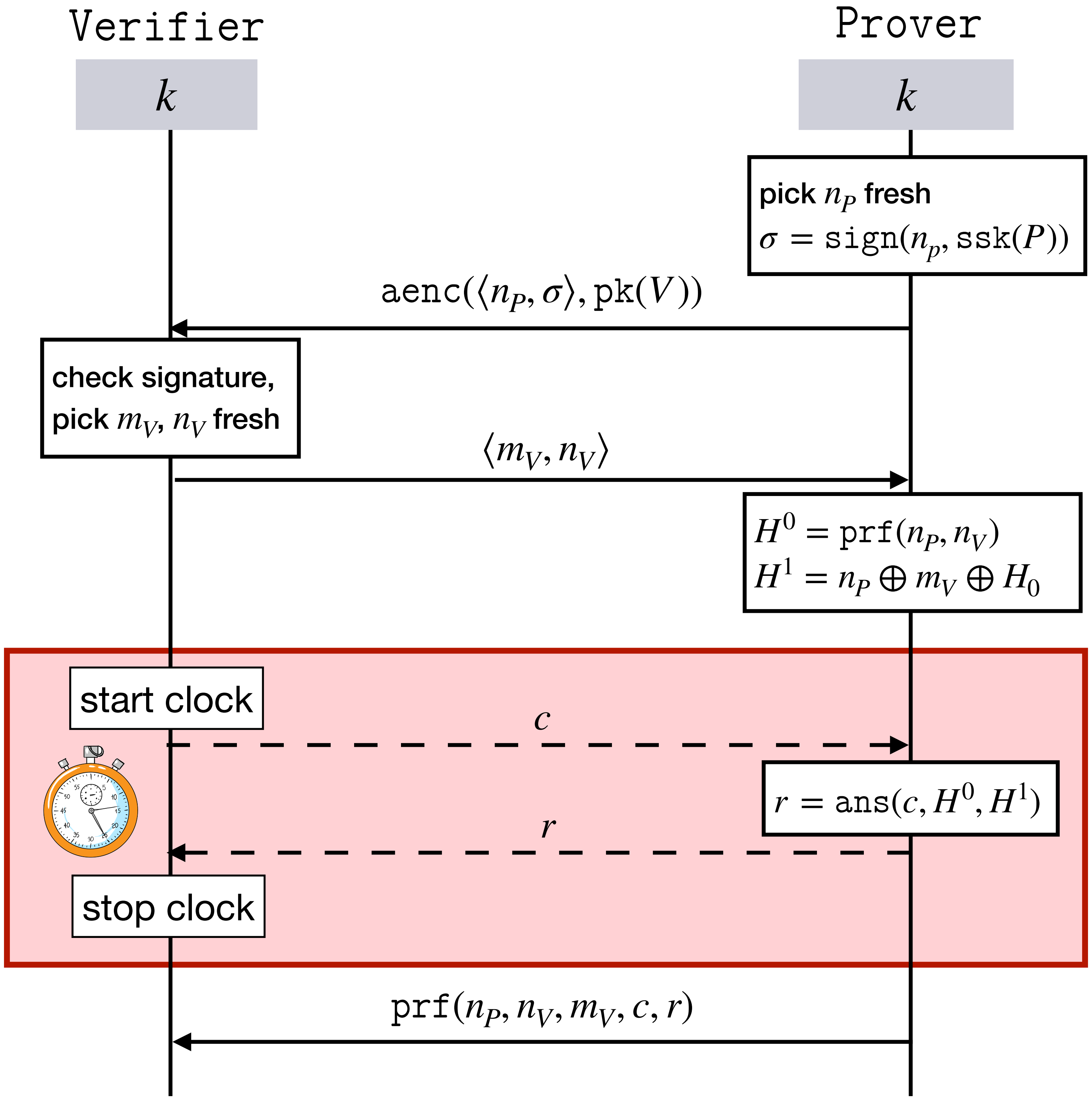
5. Apply the tools and **prove their efficiency** in practice

A symbolic model with time and locations

syntax and semantics

SPADE

[Bultel *et al.* - 2016]



Verifier
Prover
k
k
pick n_P fresh
$\sigma = \mathtt{sign}(n_p, \mathtt{ssk}(P))$
$\mathtt{aenc}(\langle n_P, \sigma \rangle, \mathtt{pk}(V))$
check signature,
pick m_V, n_V fresh
$\langle m_V, n_V \rangle$
$H^0 = \mathtt{prf}(n_P, n_V)$
$H^1 = n_P \oplus m_V \oplus H_0$
start clock
c
$r = \mathtt{ans}(c, H^0, H^1)$
r
stop clock
$\mathtt{prf}(n_P, n_V, m_V, c, r)$

Term algebra

Messages: terms built over a set of names $\mathcal{N}$ and a signature Σ given with either an equational theory E or a rewriting system.

Example

- Function symbols: $\mathtt{aenc}, \mathtt{adec}, \mathtt{pk}, \mathtt{sk}, \mathtt{sign}, \mathtt{get\_message}, \mathtt{spk}, \mathtt{ssk}, \langle \cdot , \cdot \rangle, \mathtt{proj}_1, \mathtt{proj}_2$
- Rules:

$$\mathtt{adec}(\mathtt{aenc}(x, \mathtt{pk}(y)), \mathtt{sk}(y)) \rightarrow x \qquad \mathtt{proj}_1(\langle x, y \rangle) \rightarrow x$$

$$\mathtt{get\_message}(\mathtt{sign}(x, \mathtt{ssk}(y)), \mathtt{spk}(y)) \rightarrow x \qquad \mathtt{proj}_2(\langle x, y \rangle) \rightarrow y$$

$$\mathtt{eq}(x, x) \rightarrow ok$$

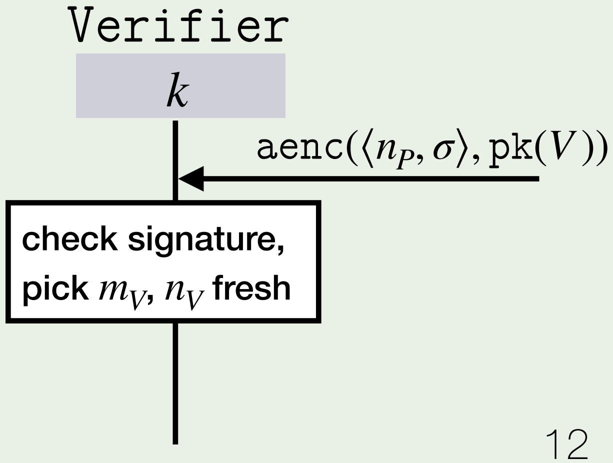
Running example

$V(v, p) = \mathtt{in}(x)\,.$

$\quad \mathtt{let}\ u = \mathtt{adec}(x, \mathtt{sk}(v))\ \mathtt{in}$

$\quad \mathtt{let}\ x_{ok} = \mathtt{eq}(\mathtt{proj}_1(u), \mathtt{get\_message}(\mathtt{proj}_2(u), \mathtt{spk}(P))\ \mathtt{in}$

$\quad \ldots$


Verifier
k
$\mathtt{aenc}(\langle n_P, \sigma \rangle, \mathtt{pk}(V))$
check signature, pick m_V, n_V fresh

Process algebra

The role of each agent is described by a process following the grammar:

$$
\begin{array}{lcll}
P & := & 0 & \text{null process} \\
  & | & \mathtt{new}\ n\,.\,P & \text{name restriction} \\
  & | & \mathtt{let}\ x = u\ \mathtt{in}\ P & \text{conditional declaration} \\
  & | & \mathtt{out}(u)\,.\,P & \text{output} \\
  & | & \mathtt{in}(x)\,.\,P & \text{input}
\end{array}
$$

Process algebra

The role of each agent is described by a process following the grammar:

$P \; := \; 0$	null process
$\mid \; \mathtt{new}\ n \,.\, P$	name restriction
$\mid \; \mathtt{let}\ x = u\ \mathtt{in}\ P$	conditional declaration
$\mid \; \mathtt{out}(u) \,.\, P$	output
$\mid \; \mathtt{in}(x) \,.\, P$	input
$\mid \; \mathtt{in}^{<t}(x) \,.\, P$	guarded input
$\mid \; \mathtt{reset} \,.\, P$	personal clock reset

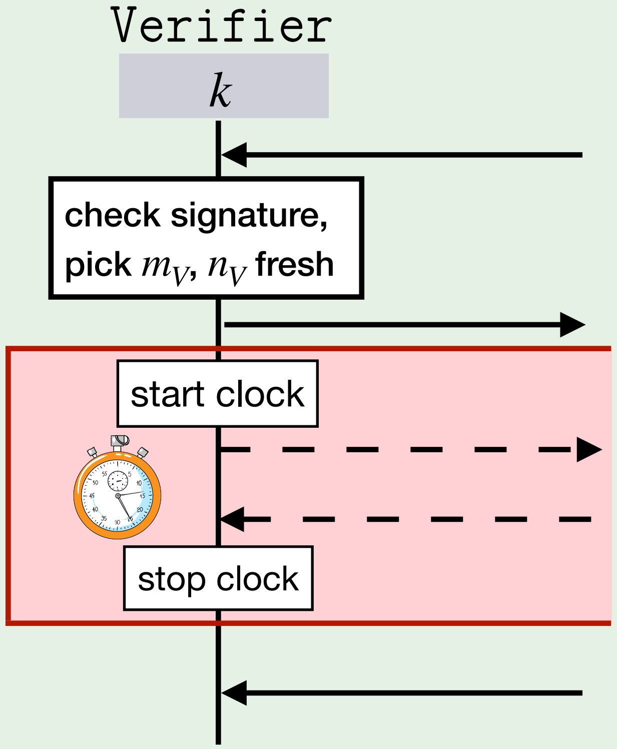
Process algebra

The role of each agent is described by a process following the grammar:

$P \; :=$	0	null process
$\mid$	$\mathtt{new}\ n\,.\,P$	name restriction
$\mid$	$\mathtt{let}\ x = u\ \mathtt{in}\ P$	conditional declaration
$\mid$	$\mathtt{out}(u)\,.\,P$	output
$\mid$	$\mathtt{in}(x)\,.\,P$	input
$\mid$	$\mathtt{in}^{<t}(x)\,.\,P$	guarded input
$\mid$	$\mathtt{reset}\,.\,P$	personal clock reset

Running example

$$
\begin{aligned}
V(v,p) = {} & \mathtt{in}(x)\,.\\
& \mathtt{let}\ u = \mathtt{adec}(x, \mathtt{sk}(v))\ \mathtt{in}\\
& \mathtt{let}\ x_{ok} = \mathtt{eq}(\mathtt{proj}_1(u), \mathtt{get\_message}(\mathtt{proj}_2(u), \mathtt{spk}(P))\ \mathtt{in}\\
& \mathtt{new}\ m_V\,.\,\mathtt{new}\ n_V\,.\\
& \mathtt{out}(\langle m_V, n_V\rangle)\,.\\
& \mathtt{reset}\,.\,\mathtt{new}\ c\,.\,\mathtt{out}(c)\,.\,\mathtt{in}^{<t}(y)\,.\\
& \mathtt{in}(z)\,.\ldots
\end{aligned}
$$



Verifier

k

check signature, pick m_V, n_V fresh

start clock

stop clock

Semantics

Physical restrictions

- locations: elements in $\mathbb{R}^3$, i.e. $\texttt{Loc} : \mathscr{A} \to \mathbb{R}^3$
- distance: Euclidean norm between locations, i.e. $\texttt{Dist}(a, b) = \dfrac{\|\texttt{Loc}(a) - \texttt{Loc}(b)\|}{c}$
- message transmission: a message takes time to reach its destination

Semantics

Physical restrictions

- locations: elements in $\mathbb{R}^3$, i.e. $\mathtt{Loc} : \mathscr{A} \rightarrow \mathbb{R}^3$
- distance: Euclidean norm between locations, i.e. $\mathtt{Dist}(a, b) = \dfrac{\|\mathtt{Loc}(a) - \mathtt{Loc}(b)\|}{c}$
- message transmission: a message takes time to reach its destination

System configuration $(\mathscr{P}, \Phi, t)$

- $\mathscr{P}$: multiset of processes which remain to execute, i.e.
- Φ: frame made of the output messages so far, i.e. $w \xrightarrow{a, t_a} u$
- t: current global time

Semantics

Physical restrictions

- locations: elements in $\mathbb{R}^3$, i.e. $\mathtt{Loc} : \mathscr{A} \to \mathbb{R}^3$
- distance: Euclidean norm between locations, i.e. $\mathtt{Dist}(a,b) = \dfrac{\|\mathtt{Loc}(a) - \mathtt{Loc}(b)\|}{c}$
- message transmission: a message takes time to reach its destination

System configuration $(\mathscr{P}, \Phi, t)$

- $\mathscr{P}$: multiset of processes which remain to execute, i.e.
- Φ: frame made of the output messages so far, i.e. $w \xrightarrow{a,t_a} u$
- t: current global time

Execution rules

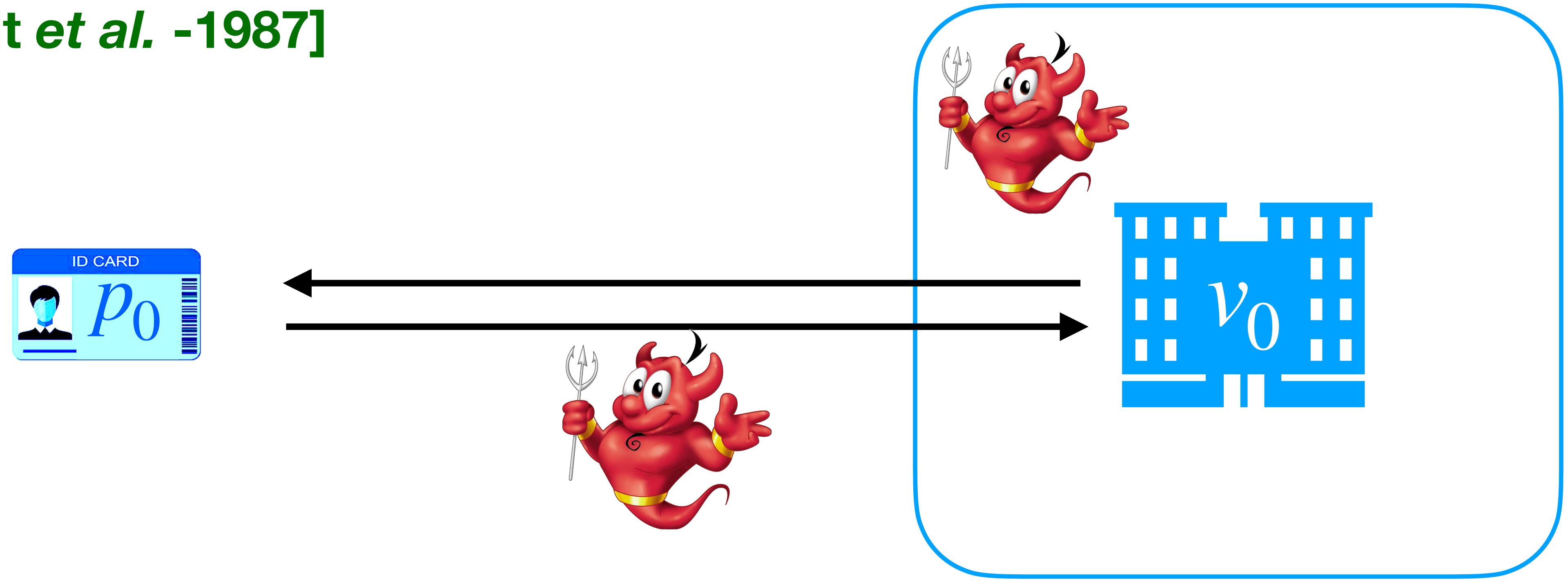
- *TIM*: $(\mathscr{P}, \Phi, t) \longrightarrow (\mathtt{Shift}(\mathscr{P}, \delta), \Phi, t + \delta)$ with $\delta > 0$
- *OUT*: $(\lfloor \mathtt{out}(u) \,.\, P \rfloor_a^{t_a} \uplus \mathscr{P}, \Phi, t) \xrightarrow{a,\mathtt{out}(u)} (\lfloor P \rfloor_a^{t_a} \uplus \mathscr{P}, \Phi \cup \{w \xrightarrow{a,t} u\}, t)$
- *IN*: $(\lfloor \mathtt{in}(x) \,.\, P \rfloor_a^{t_a} \uplus \mathscr{P}, \Phi, t) \xrightarrow{a,\mathtt{in}(u)} (\lfloor P\{x \mapsto u\} \rfloor_a^{t_a} \uplus \mathscr{P}, \Phi, t)$

  if u is deducible from Φ at time t
- …

Mafia fraud (MiM attacks)

An honest verifier shall not authenticate an honest and distant prover even in presence of an attacker in his vicinity.

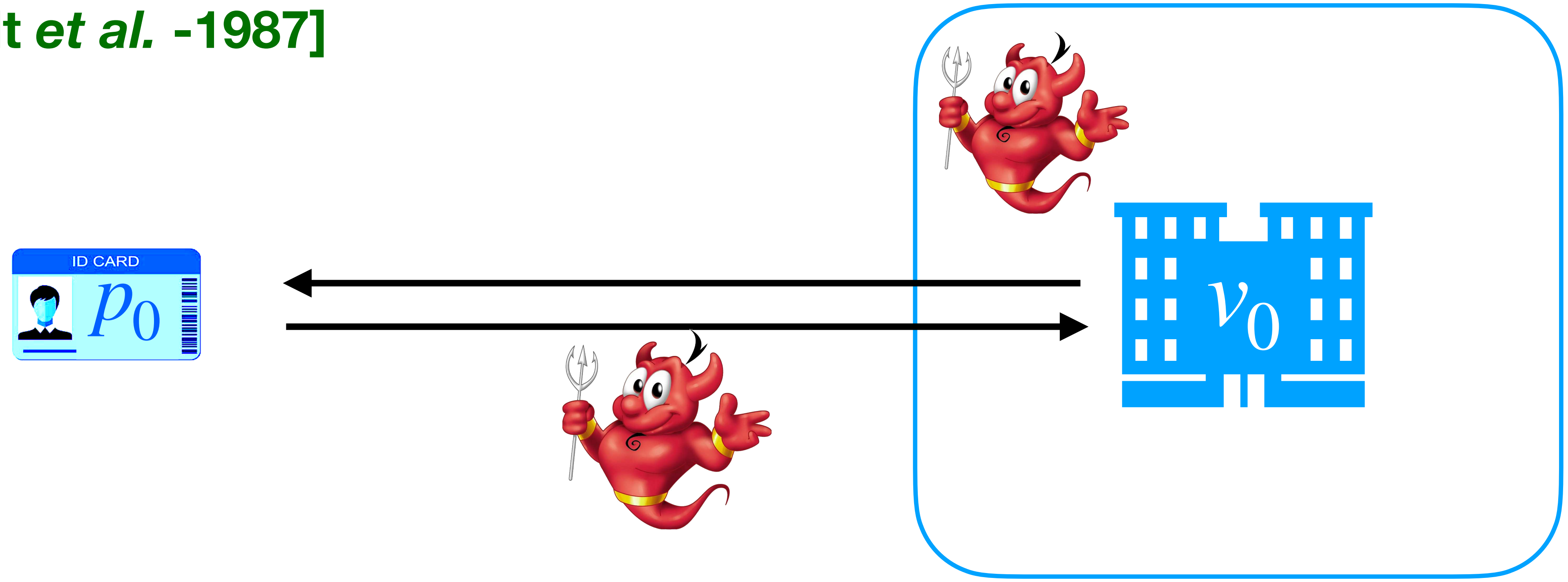
[Desmedt *et al.* -1987]



Mafia fraud (MiM attacks)

An honest verifier shall not authenticate an honest and distant prover even in presence of an attacker in his vicinity.

[Desmedt *et al.* -1987]



Definition

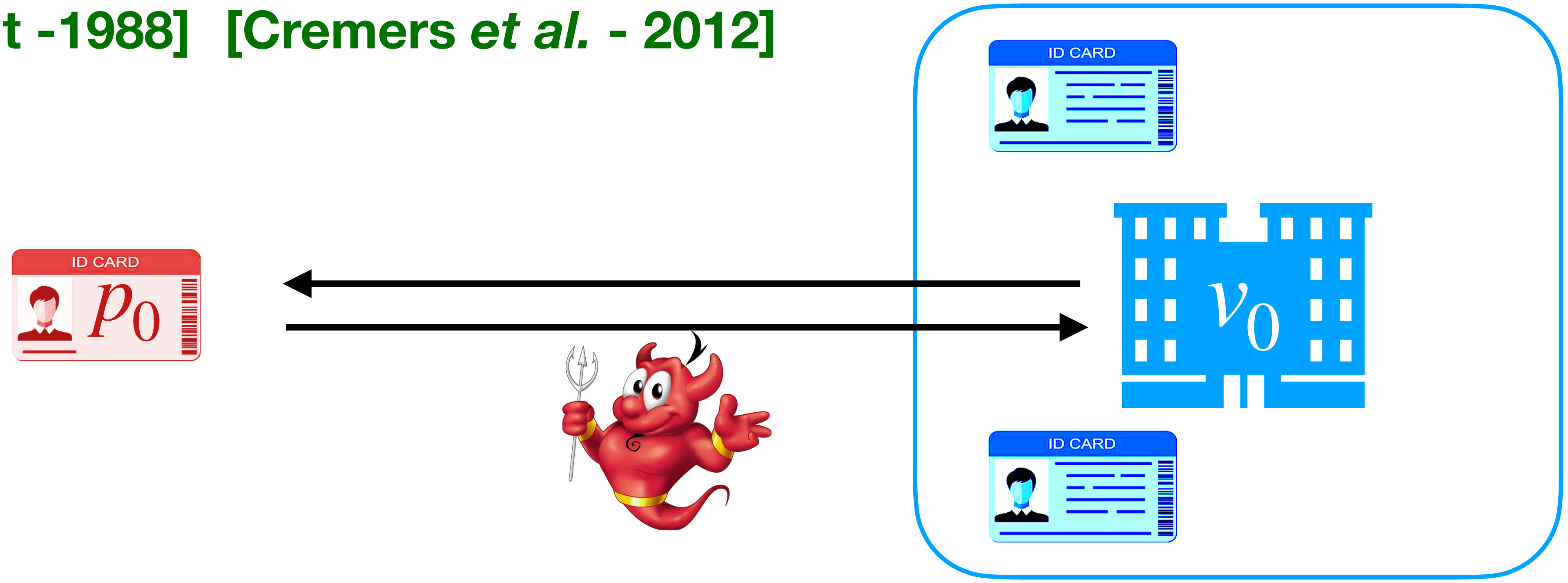
A protocol admits a mafia fraud if there exists a topology $\mathscr{T} \in \mathscr{C}_{\mathtt{MF}}$ and an initial configuration K such that:

$$K \longrightarrow (\lfloor \mathtt{end}(v_0, p_0) \rfloor_{v_0}^{t_{v_0}} \; ; \; \Phi \; ; \; t)$$

Distance hijacking attack

An honest verifier shall not authenticate a malicious and distant prover even in the presence of honest participants in his vicinity.

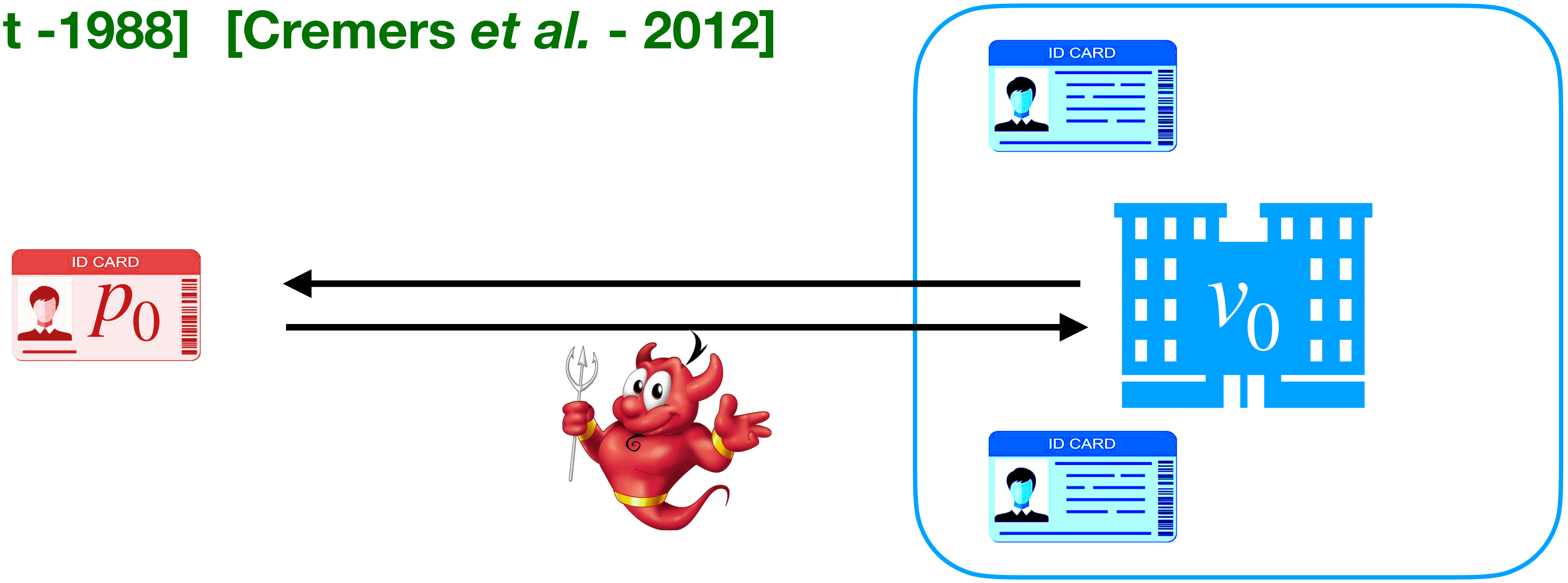
[Desmedt -1988] [Cremers *et al.* - 2012]



Distance hijacking attack

An honest verifier shall not authenticate a malicious and distant prover even in the presence of honest participants in his vicinity.

[Desmedt -1988] [Cremers *et al.* - 2012]



Definition

A protocol admits a distance hijacking attack if there exists a topology $\mathcal{T} \in \mathcal{C}_{\mathtt{DH}}$ and an initial configuration K such that:

$$K \longrightarrow (\lfloor \mathtt{end}(v_0, p_0) \rfloor_{v_0}^{t_{v_0}} \; ; \; \Phi \; ; \; t)$$

Some reduction results

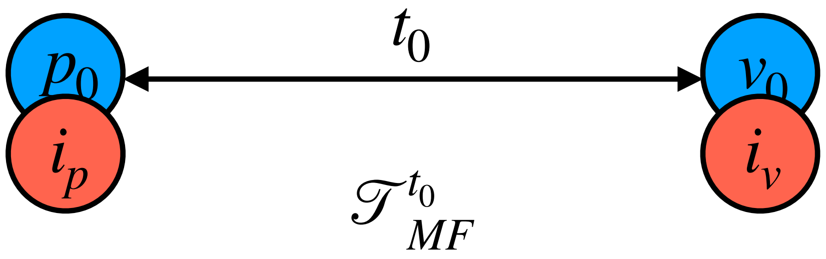
Topologies and time

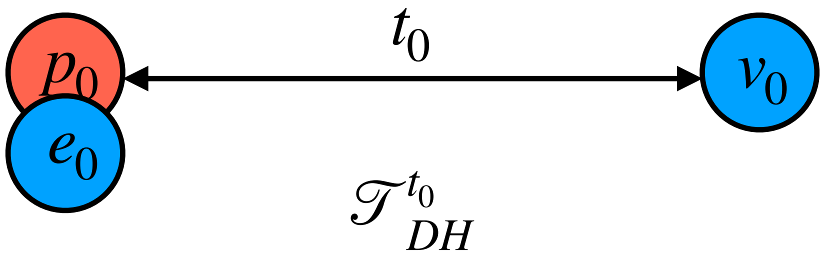
Main difficulties

1. An infinite number of topologies must be considered for each class of attacks

Main difficulties

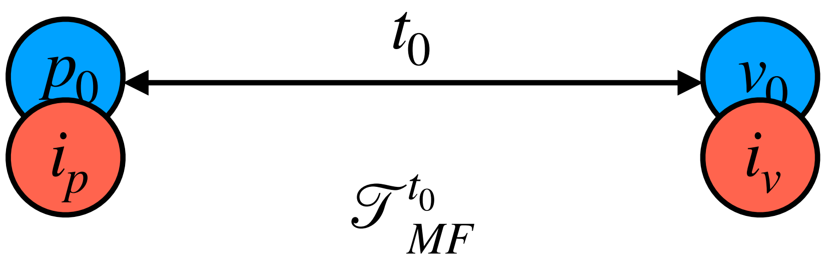
1. An infinite number of topologies must be considered for each class of attacks
   —> it is sufficient to focus on a unique topology for each class!

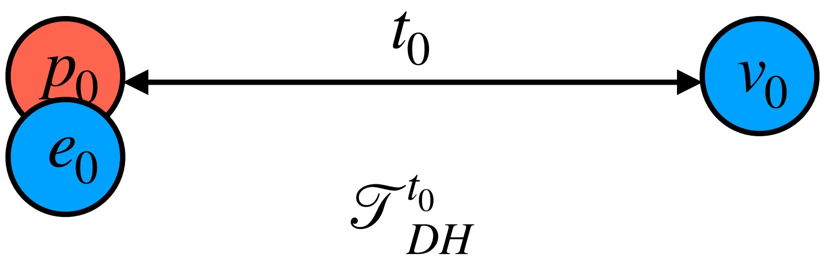




Main difficulties

1. An infinite number of topologies must be considered for each class of attacks
   —> it is sufficient to focus on a unique topology for each class!

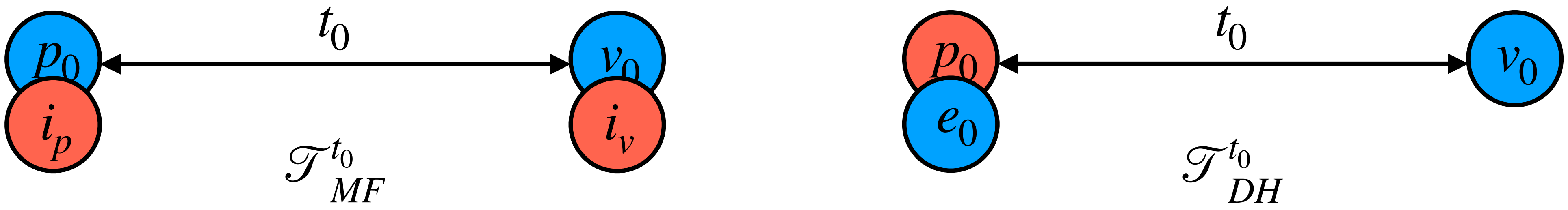




2. We must deal with time when conducting our analyses

Main difficulties

1. An infinite number of topologies must be considered for each class of attacks
   —> it is sufficient to focus on a unique topology for each class!

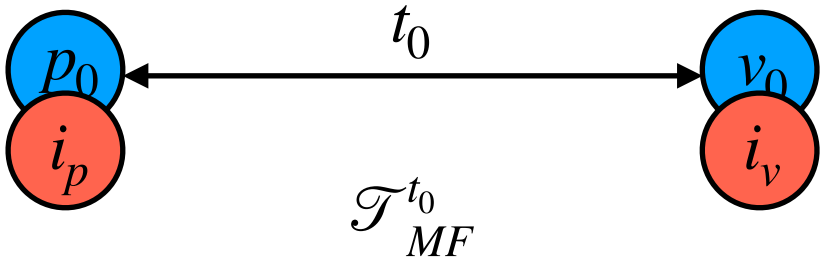


2. We must deal with time when conducting our analyses
   —> we can use ProVerif’s phases to encode the topologies!

Mafia frauds

Theorem

A protocol admits a mafia fraud, if and only if, there is an attack in $\mathscr{T}^{t_0}_{MF}$.

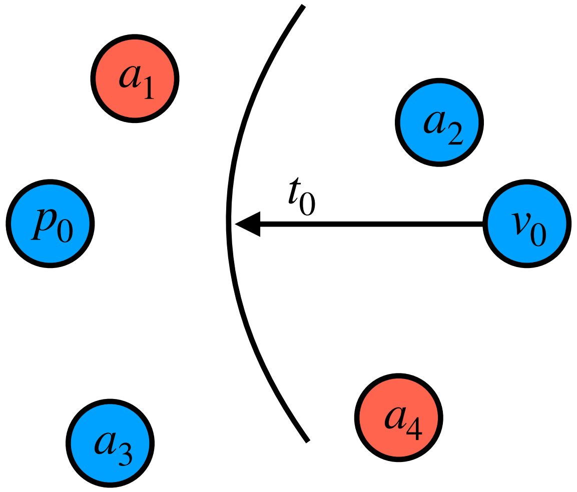


Mafia frauds

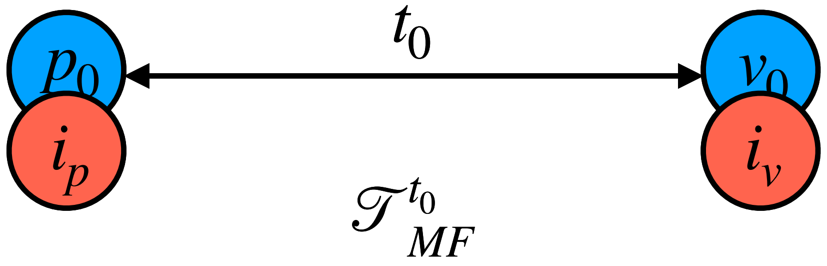
Theorem

A protocol admits a mafia fraud, if and only if, there is an attack in $\mathscr{T}^{t_0}_{MF}$.

Sketch of proof:



An attack trace in an **arbitrary** topology

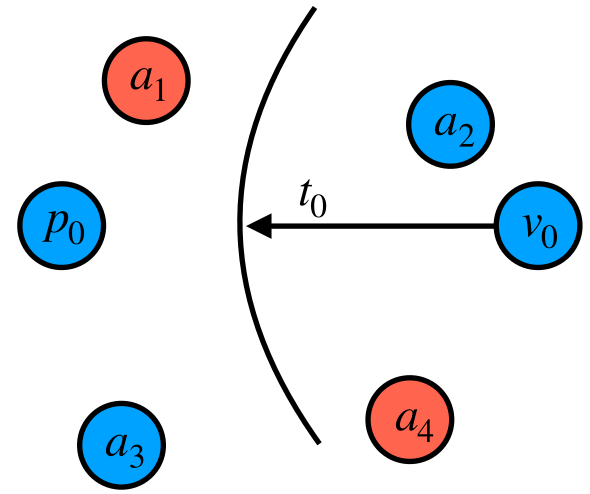


Mafia frauds

Theorem

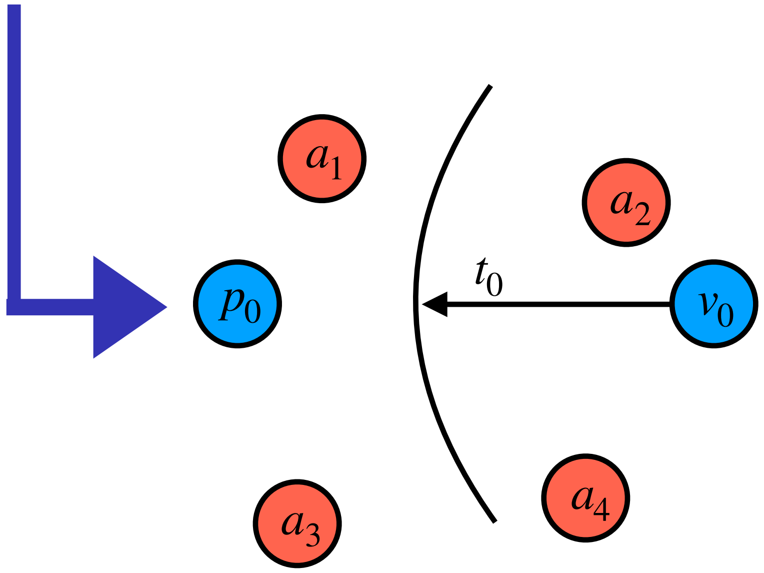
A protocol admits a mafia fraud, if and only if, there is an attack in $\mathcal{T}^{t_0}_{MF}$.

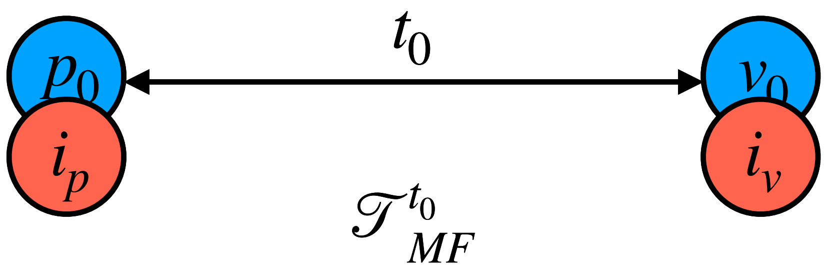
Sketch of proof:



An attack trace in an **arbitrary** topology

Assume everyone **malicious**



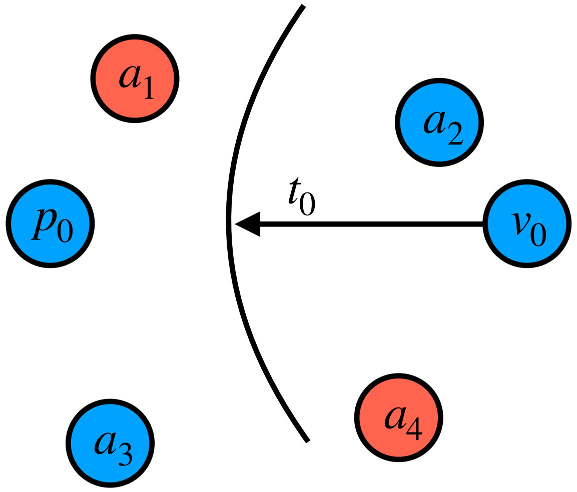


Mafia frauds

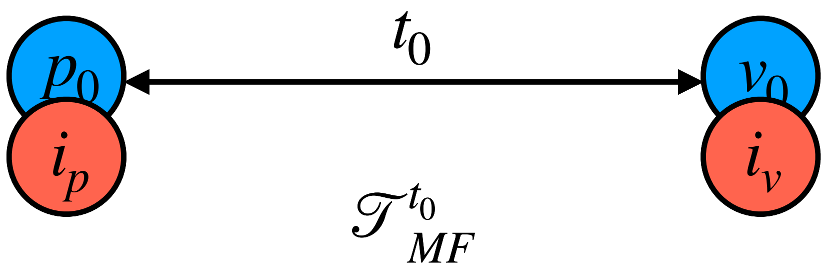
Theorem

A protocol admits a mafia fraud, if and only if, there is an attack in $\mathscr{T}^{t_0}_{MF}$.

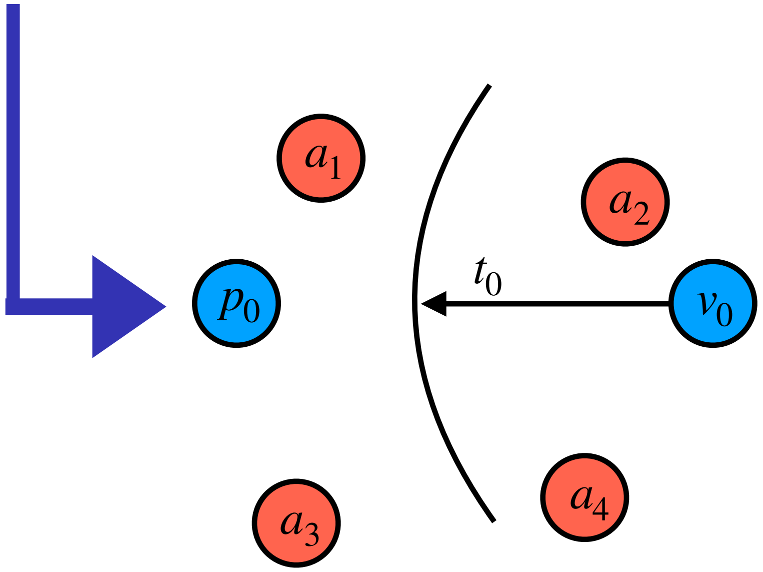
Sketch of proof:



An attack trace in an **arbitrary** topology

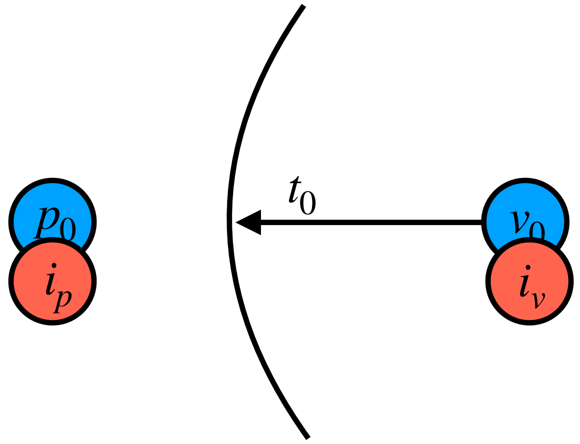


Assume everyone **malicious**



[Nigam *et al.* - 2016]

Place malicious agents **ideally**

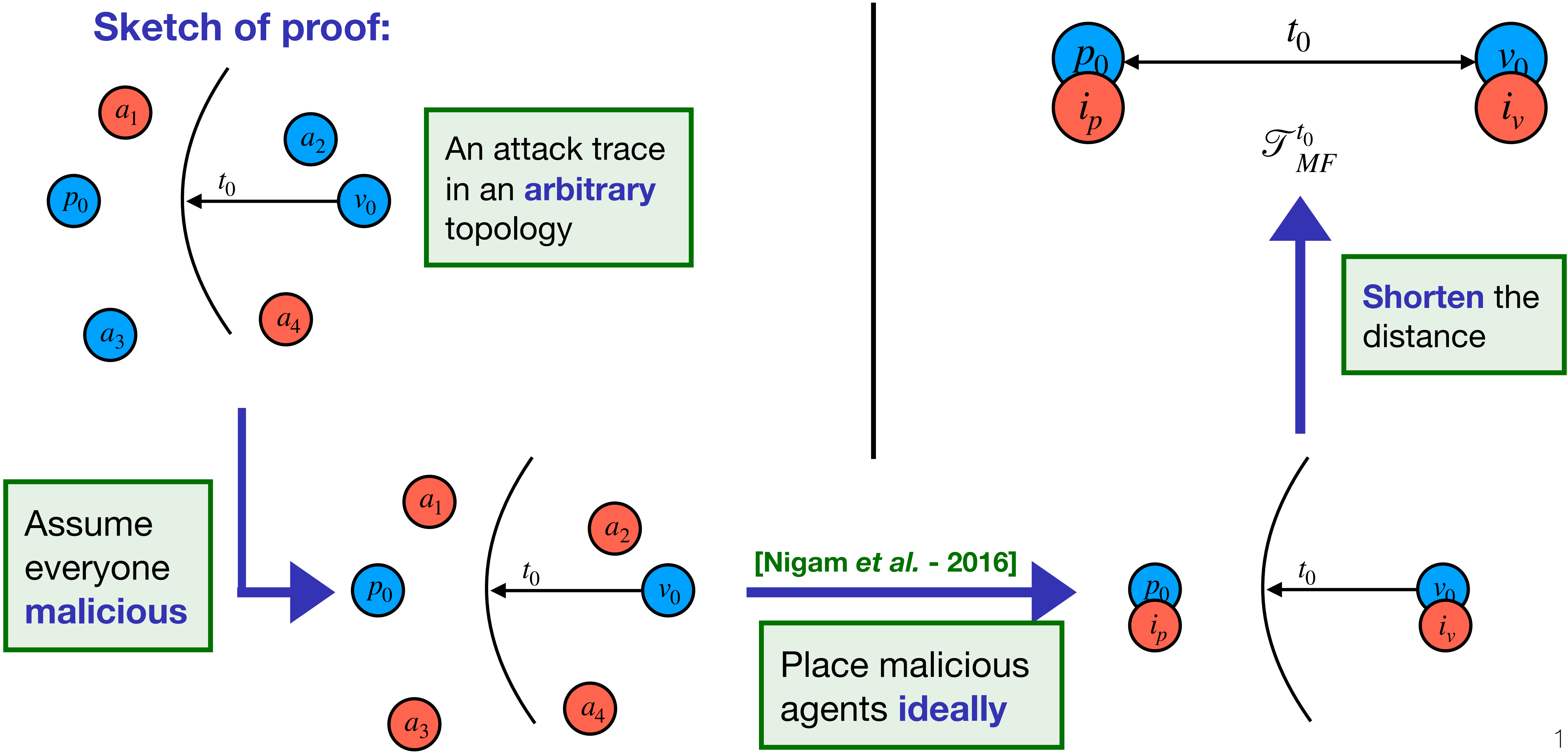


Mafia frauds

Theorem

A protocol admits a mafia fraud, if and only if, there is an attack in $\mathscr{T}^{t_0}_{MF}$.

Sketch of proof:



An attack trace in an **arbitrary** topology

Assume everyone **malicious**

[Nigam *et al.* - 2016]

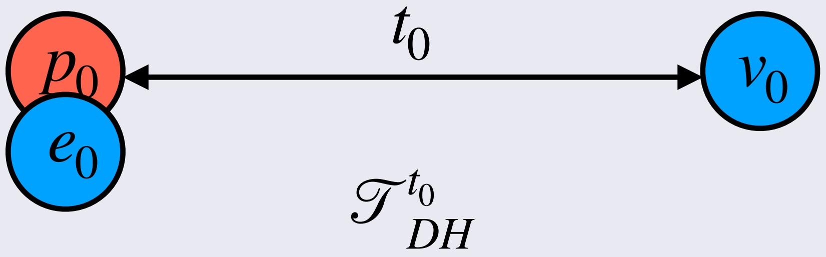
Place malicious agents **ideally**

Shorten the distance

Distance hijacking attacks

Theorem

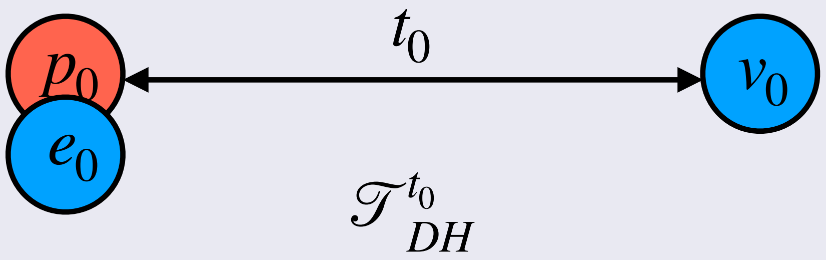
If $\mathscr{P}_{\mathtt{db}}$ admits a distance hijacking attack, then $\overline{\mathscr{P}_{\mathtt{db}}}$ admits an attack in $\mathscr{T}^{t_0}_{DH}$.



Distance hijacking attacks

Theorem

If $\mathscr{P}_{\texttt{db}}$ admits a distance hijacking attack, then $\overline{\mathscr{P}_{\texttt{db}}}$ admits an attack in $\mathscr{T}^{t_0}_{DH}$.

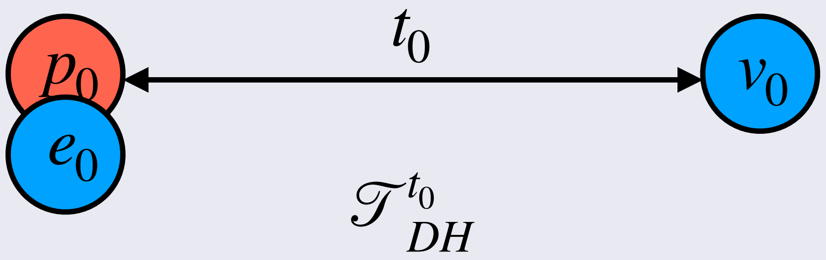


Remark: the previous proof does not apply!

Distance hijacking attacks

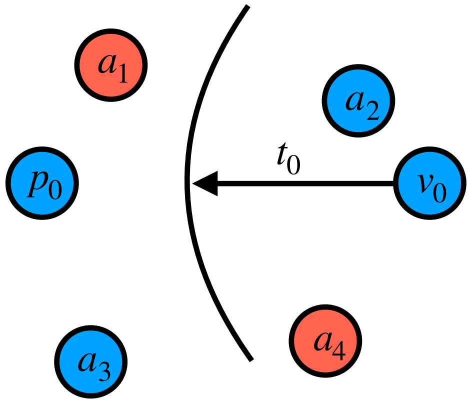
Theorem

If $\mathscr{P}_{\mathtt{db}}$ admits a distance hijacking attack, then $\overline{\mathscr{P}_{\mathtt{db}}}$ admits an attack in $\mathscr{T}^{t_0}_{DH}$.



Remark: the previous proof does not apply!

Sketch of proof:

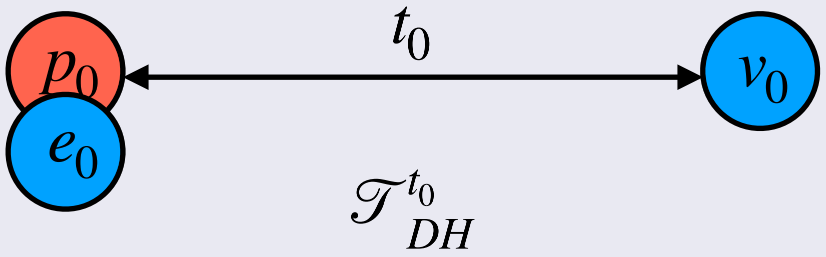


An attack trace in an **arbitrary** topology

Distance hijacking attacks

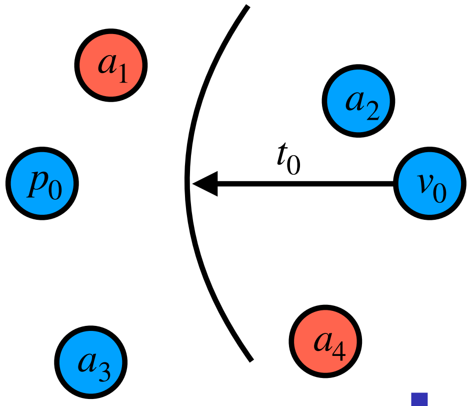
Theorem

If $\mathscr{P}_{\mathtt{db}}$ admits a distance hijacking attack, then $\overline{\mathscr{P}_{\mathtt{db}}}$ admits an attack in $\mathscr{T}^{t_0}_{DH}$.



Remark: the previous proof does not apply!

Sketch of proof:



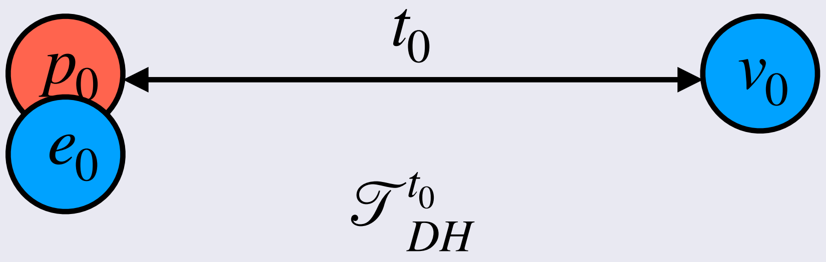
An attack trace in an **arbitrary** topology

Untimed witness of attack

Distance hijacking attacks

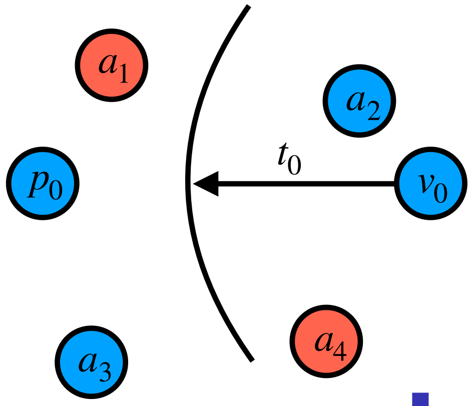
Theorem

If $\mathscr{P}_{\mathtt{db}}$ admits a distance hijacking attack, then $\overline{\mathscr{P}_{\mathtt{db}}}$ admits an attack in $\mathscr{T}^{t_0}_{DH}$.



Remark: the previous proof does not apply!

Sketch of proof:



An attack trace in an **arbitrary** topology

Untimed **witness of attack**

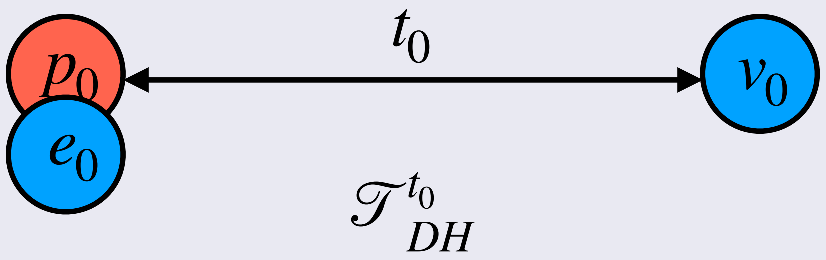
Action **re-ordering**

Untimed **witness of attack**

Distance hijacking attacks

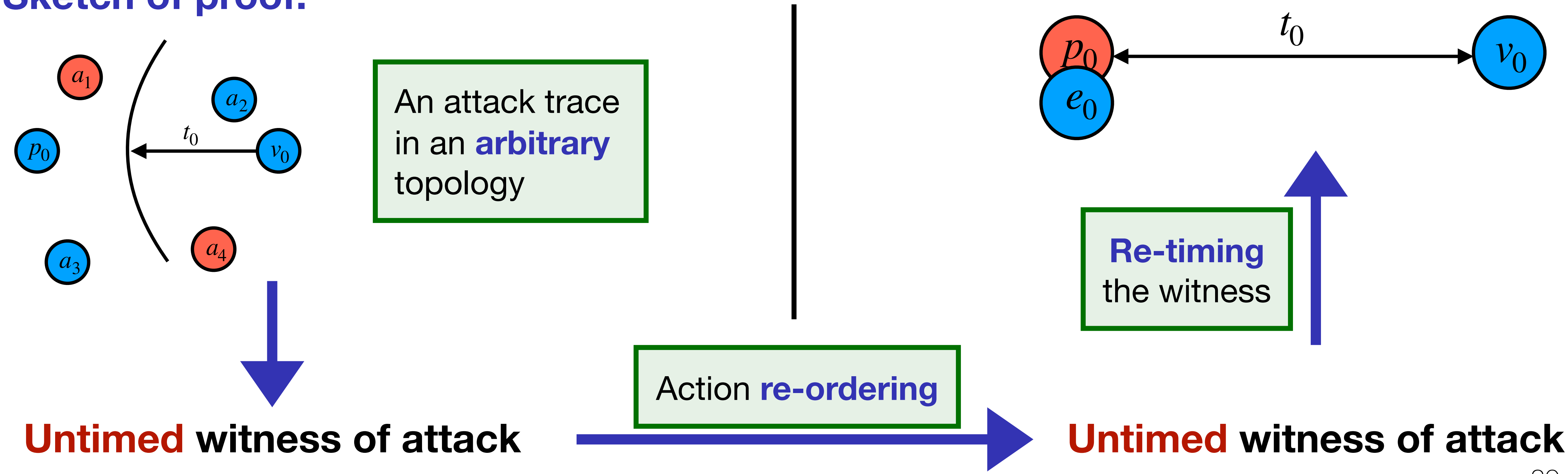
Theorem

If $\mathscr{P}_{\mathtt{db}}$ admits a distance hijacking attack, then $\overline{\mathscr{P}_{\mathtt{db}}}$ admits an attack in $\mathscr{T}^{t_0}_{DH}$.



Remark: the previous proof does not apply!

Sketch of proof:



An attack trace in an **arbitrary** topology

Untimed **witness of attack**

Action **re-ordering**

Untimed **witness of attack**

Re-timing the witness

Getting rid of time

Even a single topology cannot be modeled into existing tools

Getting rid of time

Even a single topology cannot be modeled into existing tools

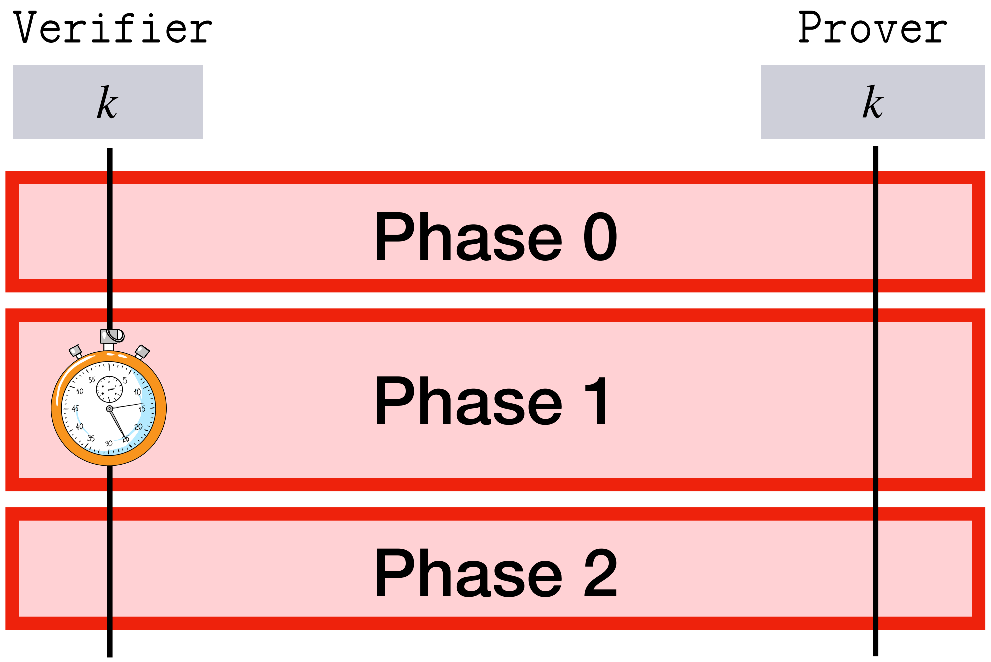
Encoding the two topologies with phases

[Chothia *et al.* - 2015]

➡ it relies on the phases of ProVerif

- *Phase 0 ⟶ slow initialization phase*
- *Phase 1 ⟶ rapid phase*
- *Phase 2 ⟶ slow verification phase*

➡ *Remote agents do not act in phase 1!*



Getting rid of time

Even a single topology cannot be modeled into existing tools

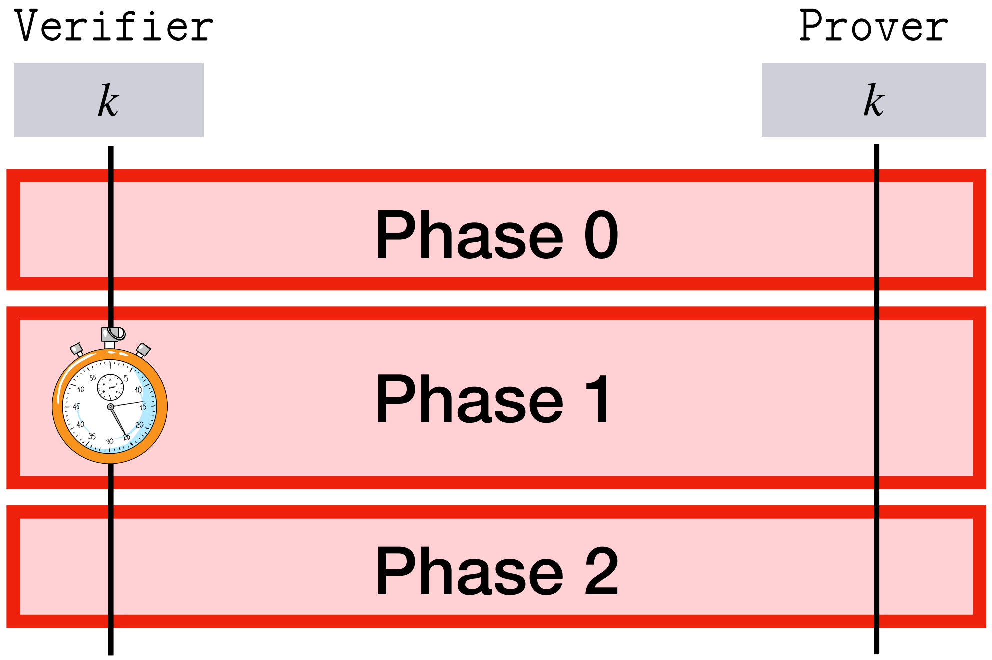
Encoding the two topologies with phases

[Chothia *et al.* - 2015]

➡ it relies on the phases of ProVerif

- *Phase 0 ⟶ slow initialization phase*
- *Phase 1 ⟶ rapid phase*
- *Phase 2 ⟶ slow verification phase*

➡ *Remote agents do not act in phase 1!*



Verifier

Prover

k

k

Phase 0

Phase 1

Phase 2

Proposition

If a protocol $\mathscr{P}_{\mathrm{db}}$ admits a mafia fraud (resp. distance hijacking, terrorist fraud) then $\mathtt{end}(v_0, p_0)$ is reachable in $\mathscr{F}(\mathscr{P}_{\mathrm{db}})$.

A comprehensive case studies analysis

Application to distance-bounding protocols

Case studies analyses

Corpus +25 protocols

Tool ProVerif (slightly modified for distance hijacking attacks)

Abstractions

- rapid phase collapsed in a single round-trip

model limitation

- weak exclusive-OR

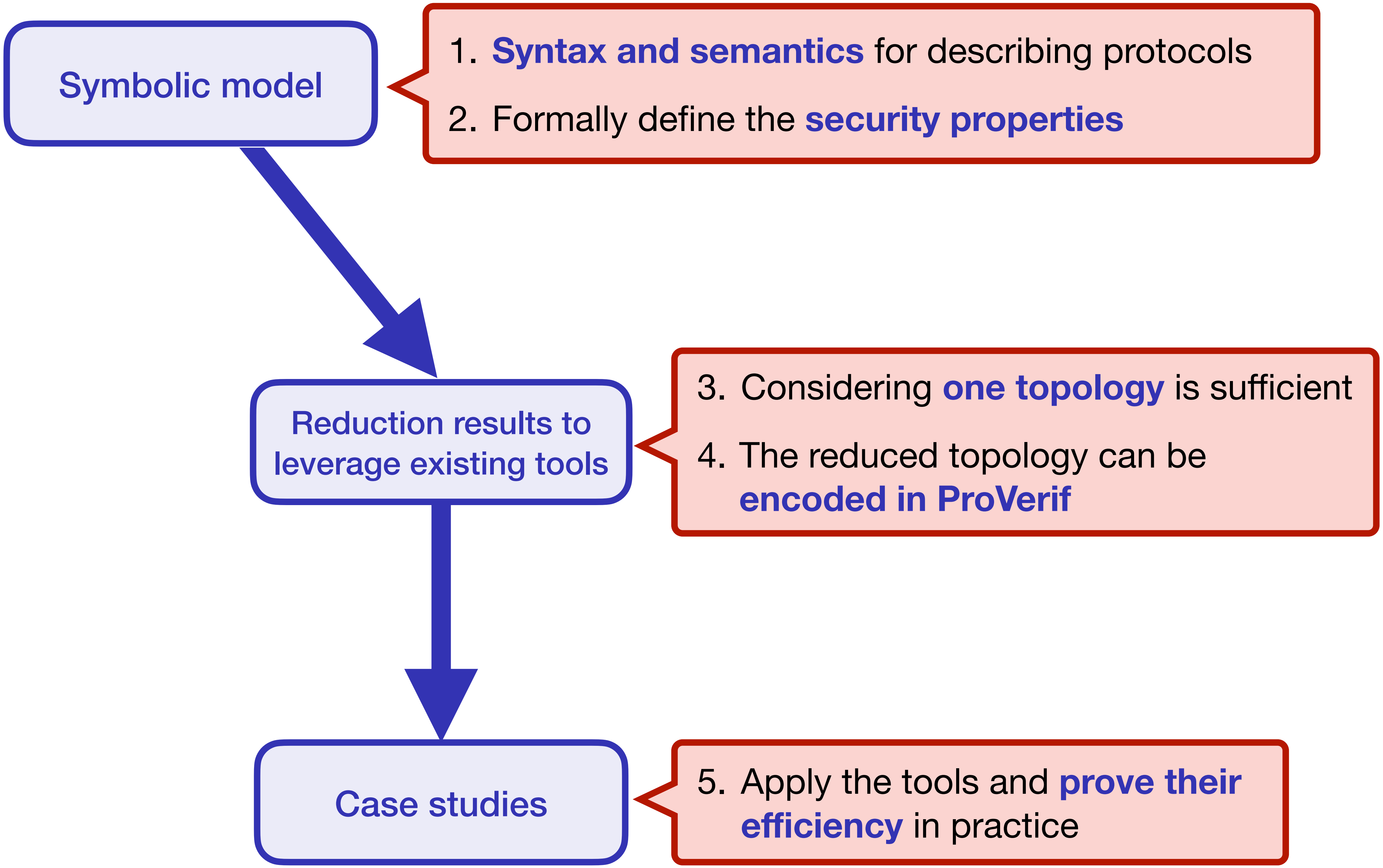
tool limitation

Application to real-world protocols

Protocols	Mafia fraud	Distance hijacking	Terrorist fraud
MasterCard RRP	✓	✗	✗
PaySafe	✓	✗	✗
MIFARE Plus	✓	✗	✗

Conclusion

Finally we have…



Symbolic model

1. **Syntax and semantics** for describing protocols
2. Formally define the **security properties**

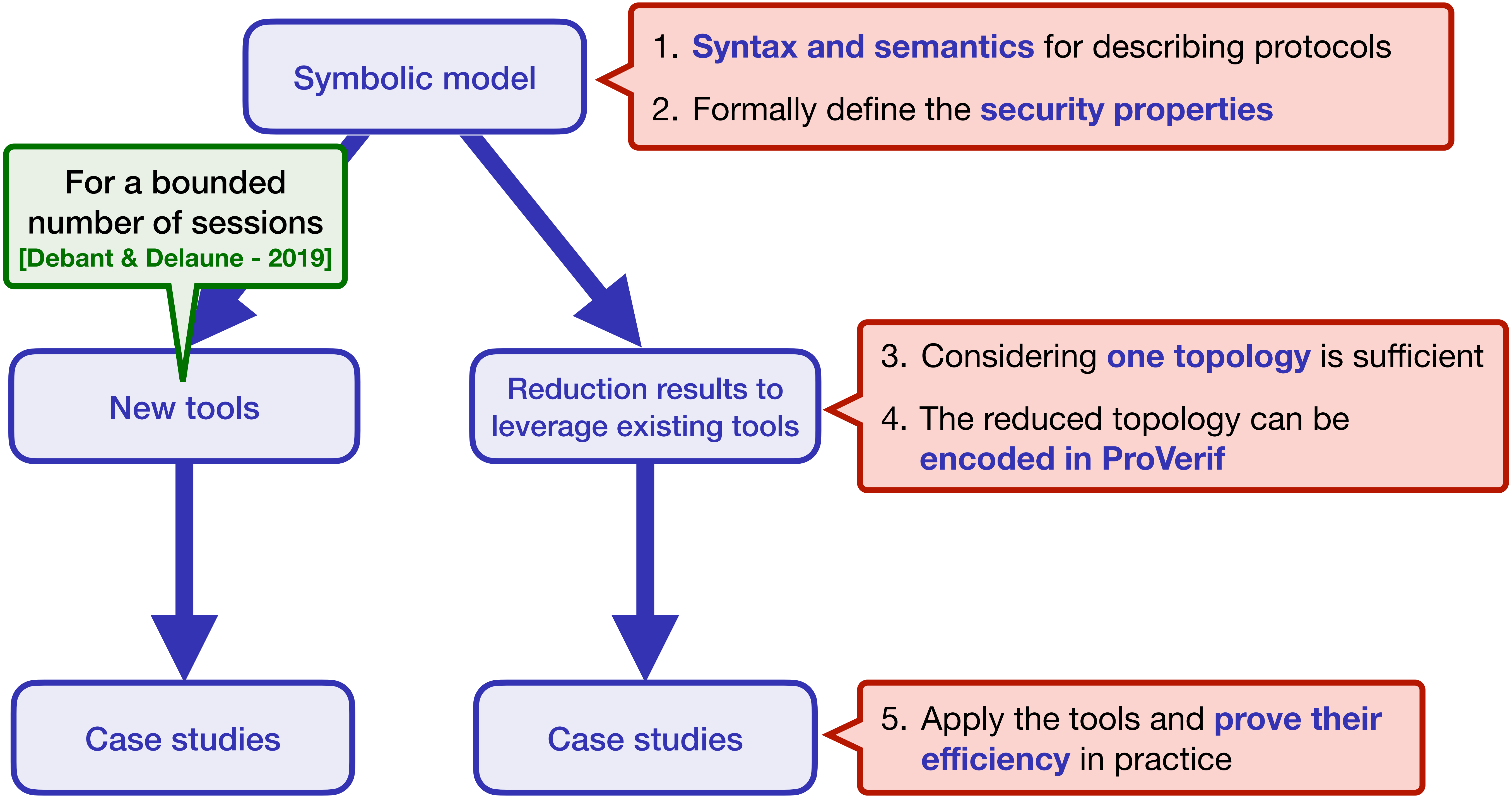
Reduction results to leverage existing tools

3. Considering **one topology** is sufficient
4. The reduced topology can be **encoded in ProVerif**

Case studies

5. Apply the tools and **prove their efficiency** in practice

Finally we have…



Symbolic model

1. **Syntax and semantics** for describing protocols
2. Formally define the **security properties**

For a bounded number of sessions **[Debant & Delaune - 2019]**

New tools

Reduction results to leverage existing tools

3. Considering **one topology** is sufficient
4. The reduced topology can be **encoded in ProVerif**

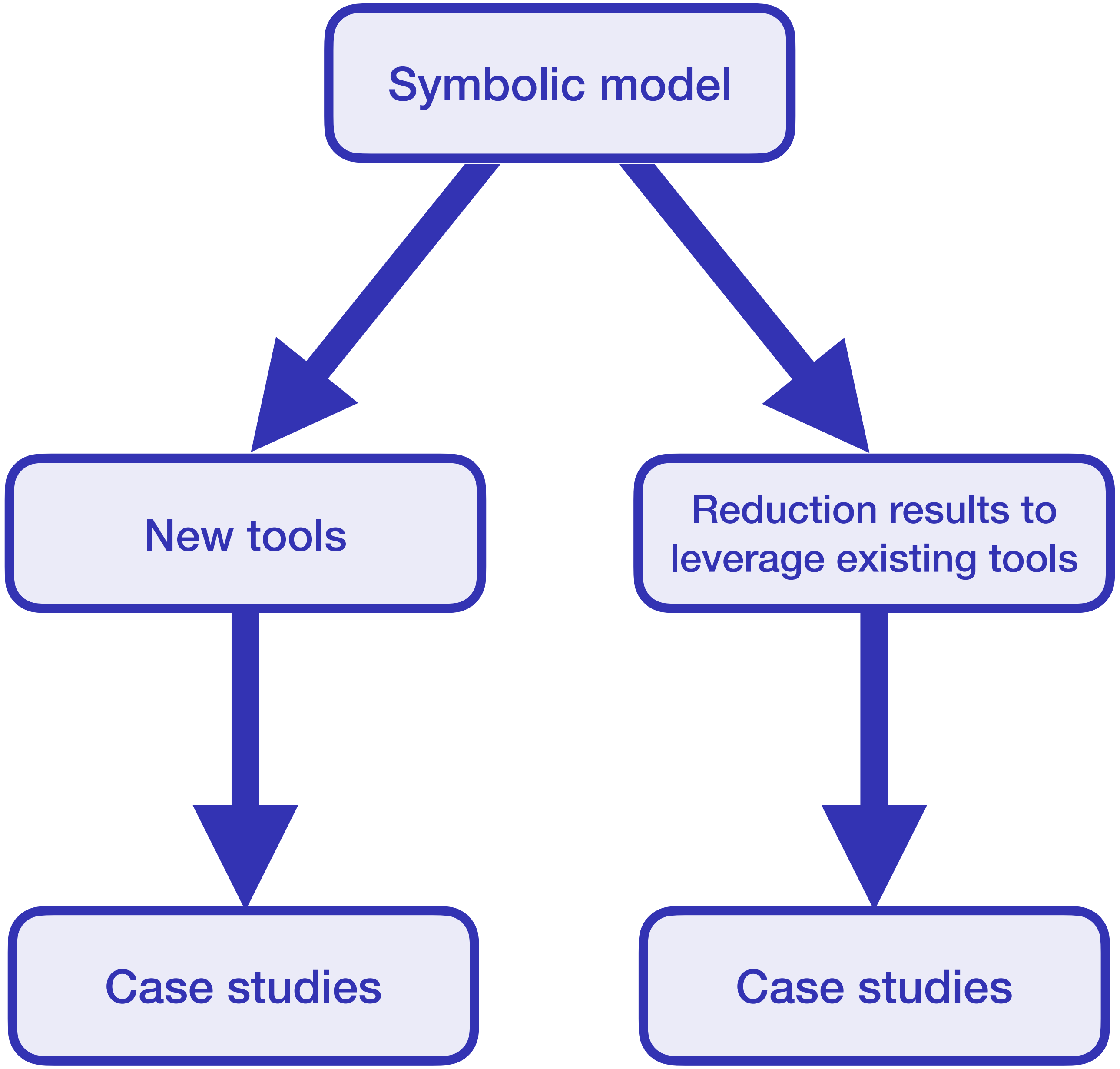
Case studies

Case studies

5. Apply the tools and **prove their efficiency** in practice

Future work

Remove hypotheses in the theorems

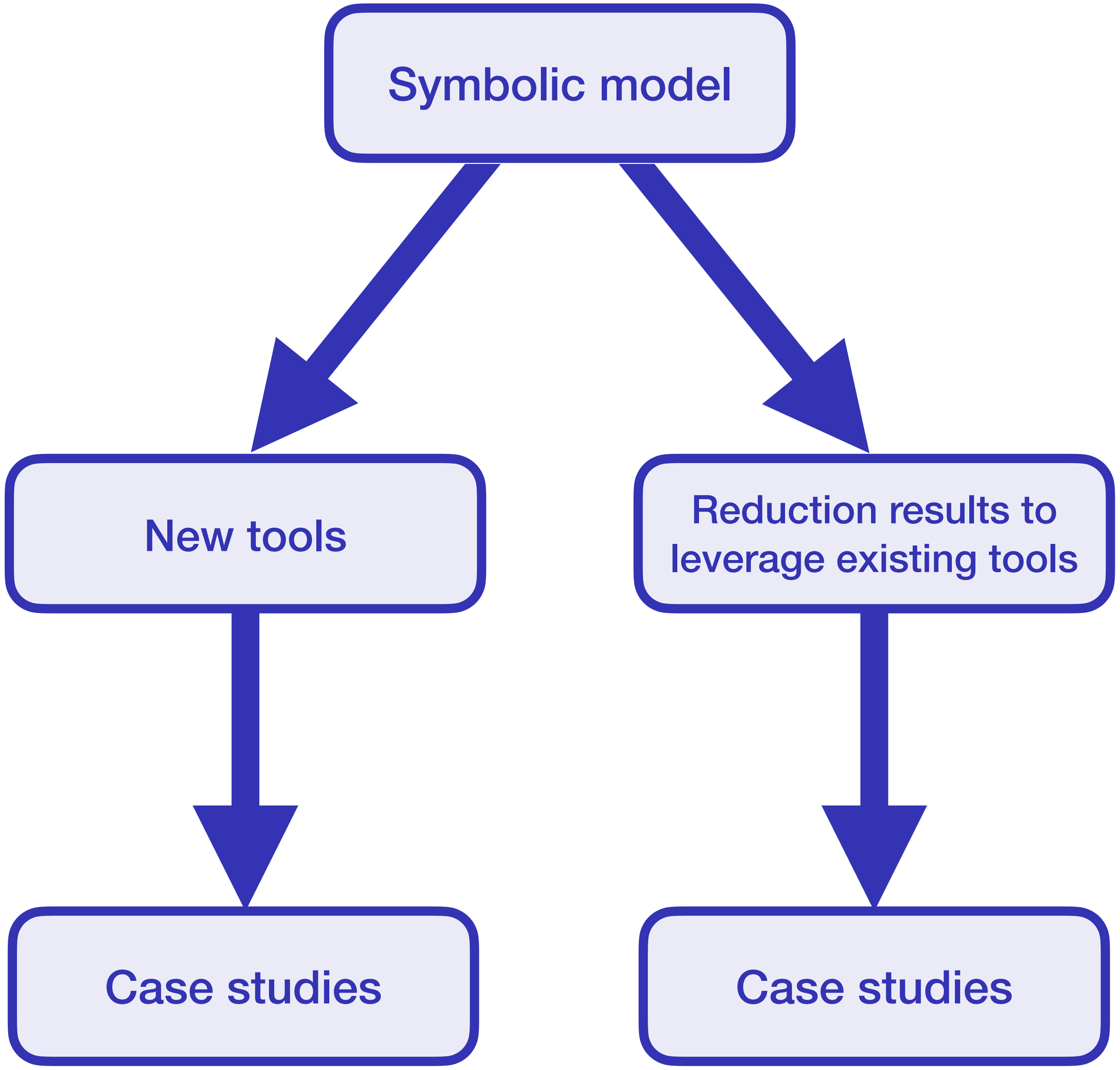


Future work

Remove hypotheses in the theorems

Make the existing tools support exclusive-OR

- extend ProVerif’s procedure
- improve automation for Tamarin



Future work

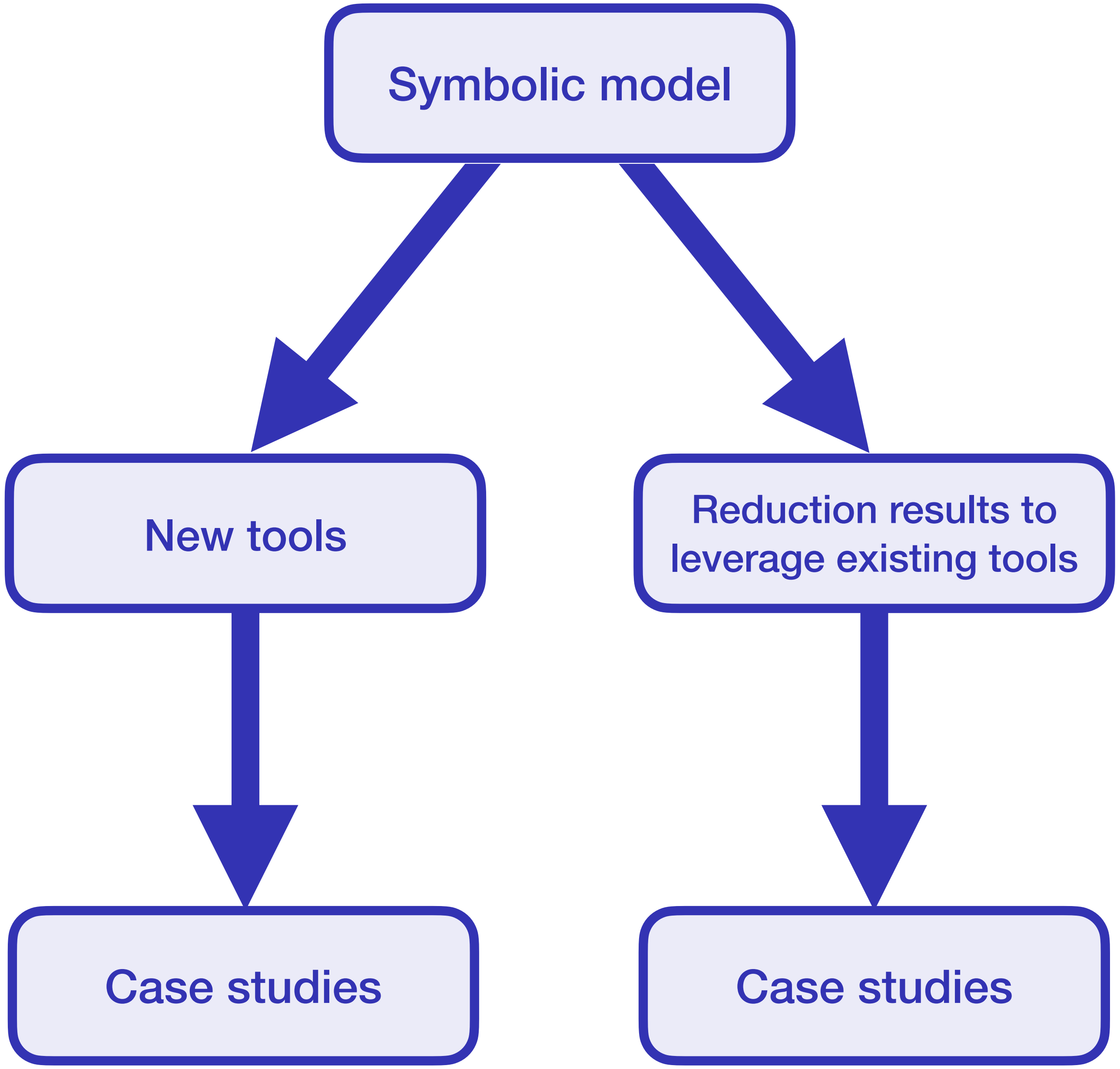
Remove hypotheses in the theorems

Make the existing tools support exclusive-OR

- extend ProVerif’s procedure
- improve automation for Tamarin

Improve the model of time

- consider computation time
- design procedures for unbounded #sessions



Symbolic model

New tools

Reduction results to leverage existing tools

Case studies

Case studies

Future work

Remove hypotheses in the theorems

Make the existing tools support exclusive-OR

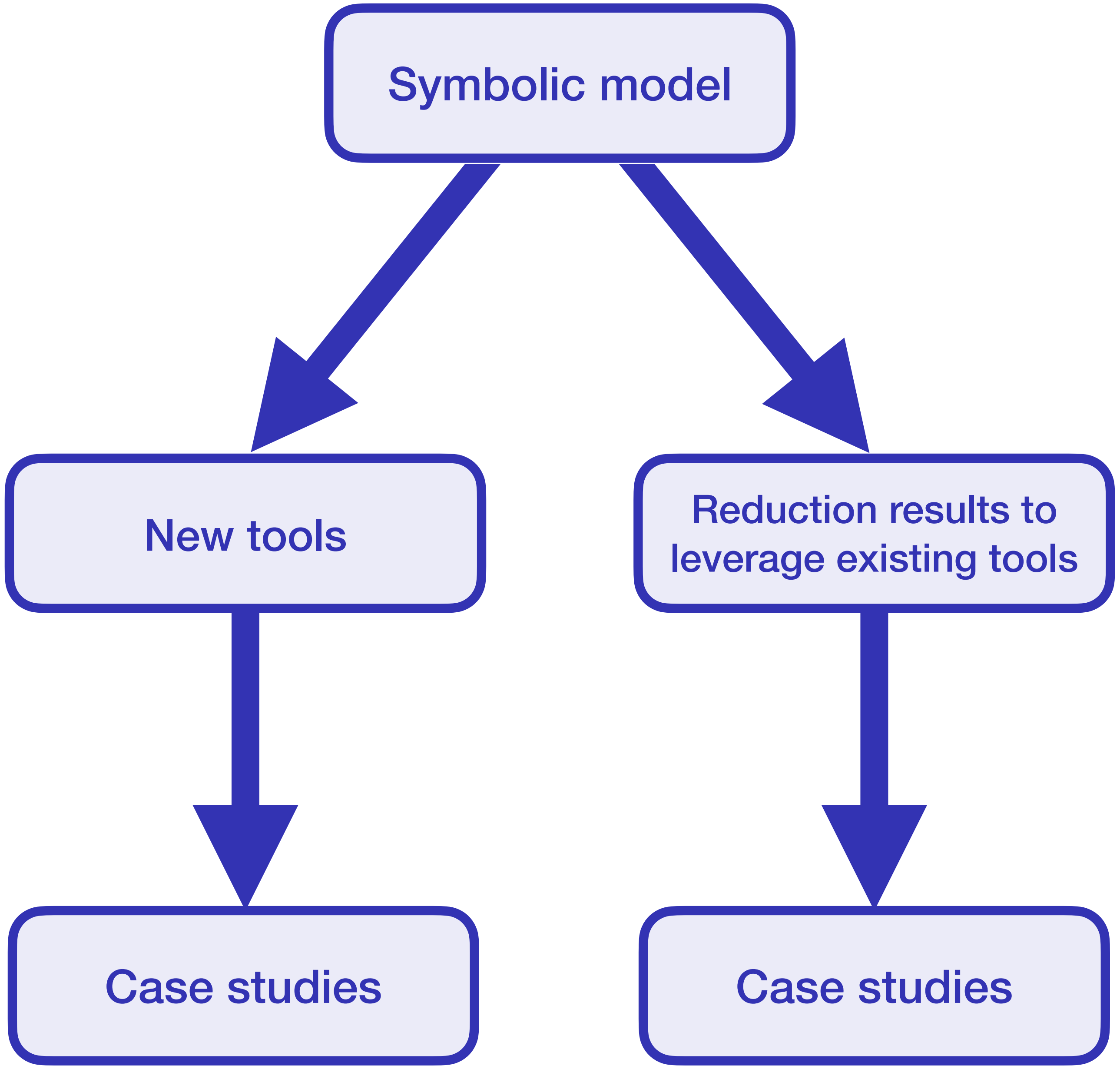
- extend ProVerif's procedure
- improve automation for Tamarin

Improve the model of time

- consider computation time
- design procedures for unbounded #sessions

Model bit-level operations

- consider probabilistic processes and properties
- model messages with bitstrings


Symbolic model
New tools
Reduction results to leverage existing tools
Case studies
Case studies