Symbolic verification of distance bounding protocols

Alexandre Debant, Stéphanie Delaune

Univ Rennes - IRISA - CNRS

February 26, 2019

Introduction

Security protocols

Distributed programs that use cryptographic primitives to ensure **security properties.**

Secrecy

Authentication

Integrity

Untraceability

Introduction

Security protocols

Distributed programs that use cryptographic primitives to ensure **security properties.**

Secrecy

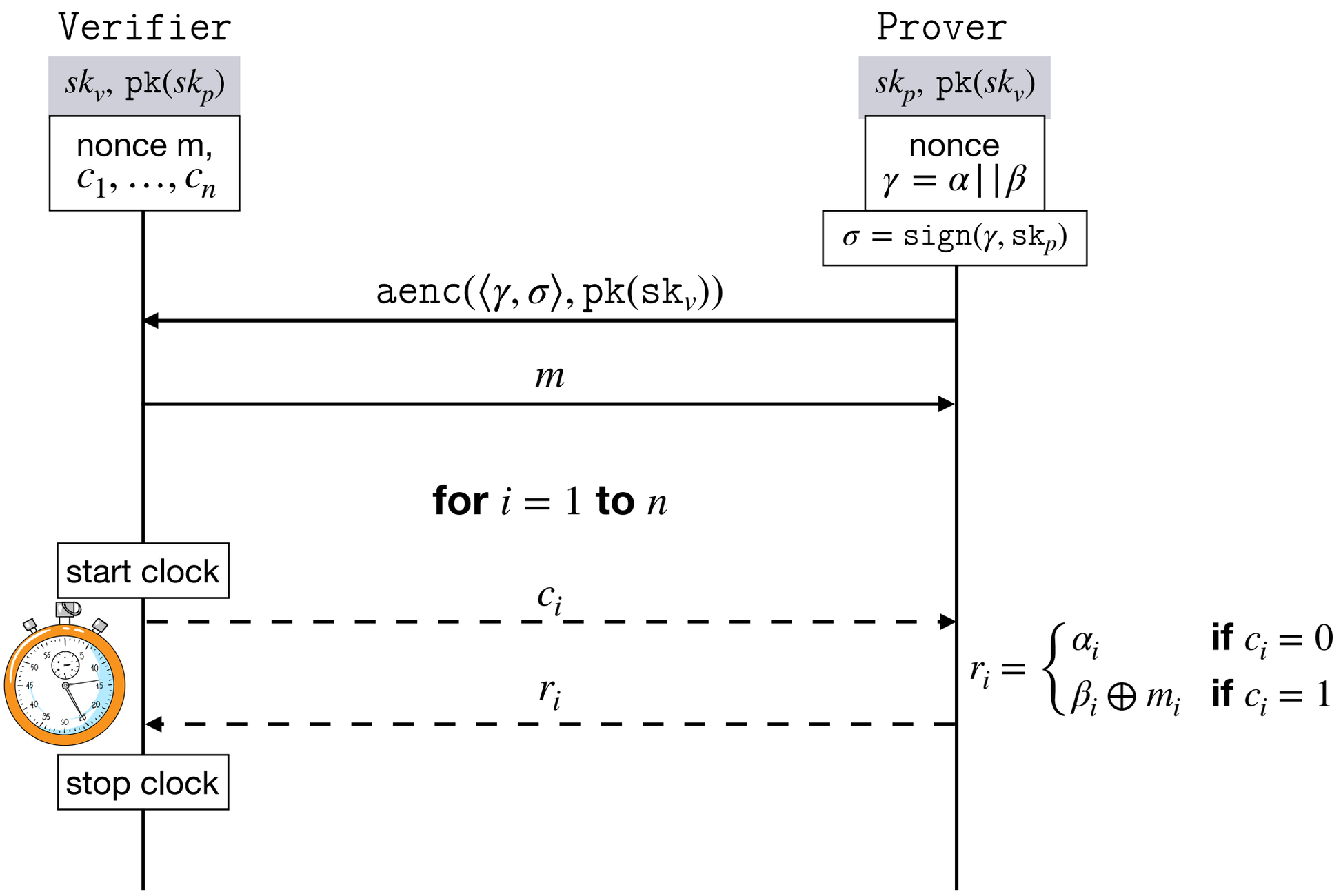
Authentication

Integrity

Untraceability

Authentication with physical proximity

Example: TREAD - 2017



Symbolic verification

Unbounded number of sessions

- undecidable in general [Durgin et al - 99]
- tools may not terminate or lead to false attacks…
- tools: ProVerif, Tamarin…

Bounded number of sessions

- decidable for many classes of protocols

[Rusinovitch et al - 01] [Millen et al - 01] [Comon et al - 03]

- tools implement exact procedures
- tools: OFMC, Akiss…

Symbolic verification

Unbounded number of sessions

- undecidable in general [Durgin et al - 99]
- tools may not terminate or lead to false attacks…
- tools: ProVerif, Tamarin…

Bounded number of sessions

- decidable for many classes of protocols

[Rusinovitch et al - 01] [Millen et al - 01] [Comon et al - 03]

- tools implement exact procedures
- tools: OFMC, Akiss…

But not adapted to analyse DB protocols!

Symbolic verification of DB protocols

Unbounded number of sessions

- existing tools can be used to verify DB protocols
- Tamarin: [Mauw et al - 18]
- ProVerif: [Chothia et al - 18] [Debant et al -18]

Bounded number of sessions

Contribution in this work

We extend the tool **Akiss** to be able to analyse DB protocols:

- adapt the model to take time into account
- provide an exact procedure

Table of contents

Term algebra

Messages: terms but over a set of names $\mathcal{N}$ and a signature Σ given with a rewriting system $\mathbb{R}$ satisfying the finite variant property.

Example

- Names: $\mathcal{N} = \{a, n, k\}$
- Signature: $\Sigma = \{\mathtt{aenc}, \mathtt{adec}, \mathtt{pk}, \mathtt{pair}, \mathtt{proj}_1, \mathtt{proj}_2, \mathtt{sign}, \mathtt{getmsg}, \mathtt{check}, \mathtt{ok},\}$

- $\mathtt{adec}(\mathtt{aenc}(x, \mathtt{pk}(y)), y) = x$

$\mathtt{getmsg}(\mathtt{sign}(x, y)) = x$ $\quad$ $\mathtt{check}(\mathtt{sign}(x, y), \mathtt{pk}(y)) = \mathtt{ok}$

$\mathtt{proj}_1(\mathtt{pair}(x, y)) = x$ $\quad$ $\mathtt{proj}_2(\mathtt{pair}(x, y)) = y$

Protocols

A trace T is a finite sequence of pairs, i.e. $T = (a_1, \mathtt{a}_1) \ldots . (a_n, \mathtt{a}_n)$ where each $a_i \in \mathscr{A}$ and $\mathtt{a}_i$ is an action of the form:

- $\mathtt{out}^z(u)$ output
- $\mathtt{in}^z(x)$ input
- $[v = v']$ test equality
- $[z := v]$ affectation of a timing variable
- $[\,|\, t_1 \sim t_2 \,|\,]$ timing constraint

Protocols

A trace T is a finite sequence of pairs, i.e. $T = (a_1, \mathtt{a}_1) \ldots . (a_n, \mathtt{a}_n)$ where each $a_i \in \mathscr{A}$ and $\mathtt{a}_i$ is an action of the form:

- $\mathtt{out}^z(u)$ output
- $\mathtt{in}^z(x)$ input
- $[v = v']$ test equality
- $[z := v]$ affectation of a timing variable
- $[\,|\, t_1 \sim t_2 \,|\,]$ timing constraint

Protocol

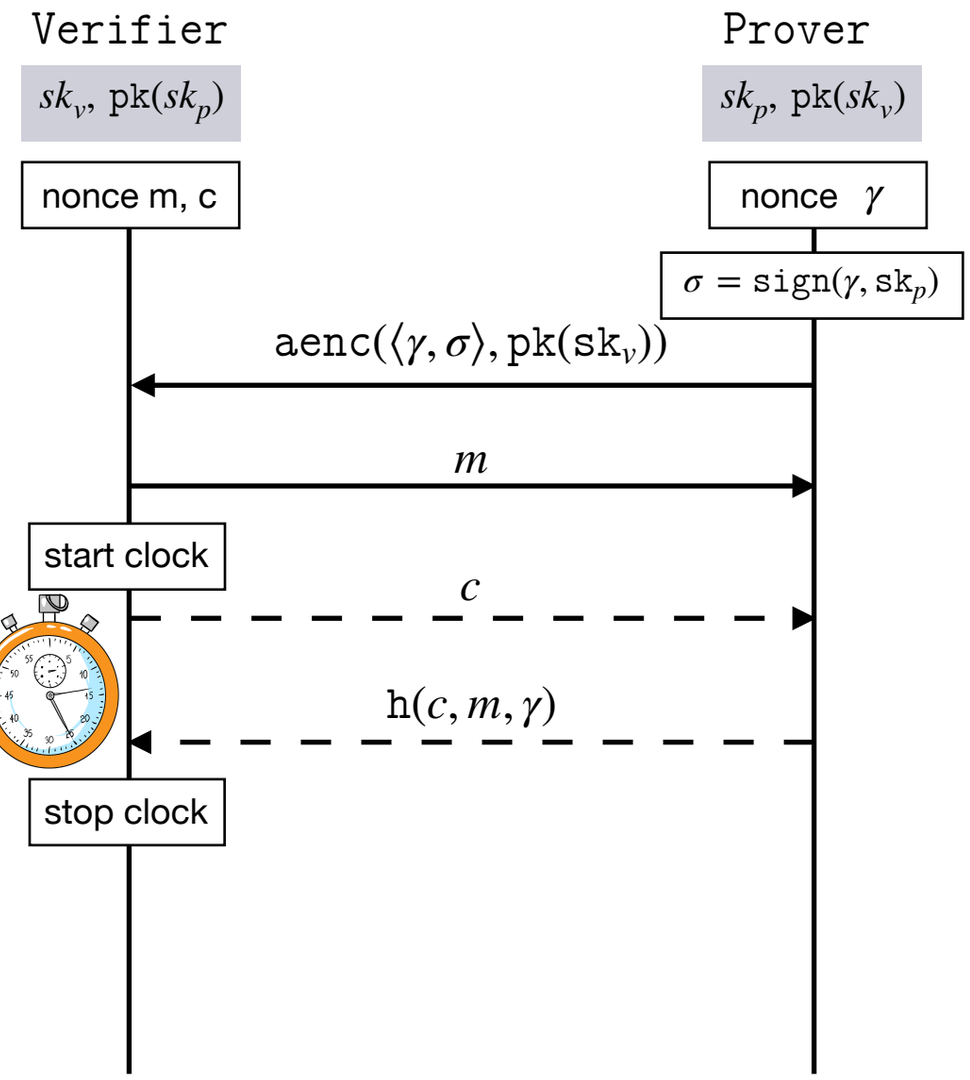
A protocol is a finite set of traces $(T_1, \ldots, T_k)$ describing all the possible interleavings of the considered sessions.

Example: TREAD - 2017

$$
\begin{aligned}
T_V = \quad & (v, \mathtt{in}(x)) . \\
& (v, [\mathtt{check}(t_\sigma, \mathtt{pk}(\mathtt{sk}_p)) = \mathtt{ok}]) . \\
& (v, [t_\gamma = \mathtt{getmsg}(t_\sigma)]) . \\
& (v, \mathtt{out}(m)) . \\
& (v, \mathtt{out}^{z_1}(c)) . \\
& (v, \mathtt{in}^{z_2}(y)) . \\
& (v, [\,|z_2 - z_1 < 2 \times t_0|\,]) . \\
& (v, [y = h(c, m, t_\gamma)])
\end{aligned}
$$

with $t_\sigma = \mathtt{proj}_2(\mathtt{adec}(x, \mathtt{sk}_v))$

$t_\gamma = \mathtt{proj}_1(\mathtt{adec}(x, \mathtt{sk}_v))$

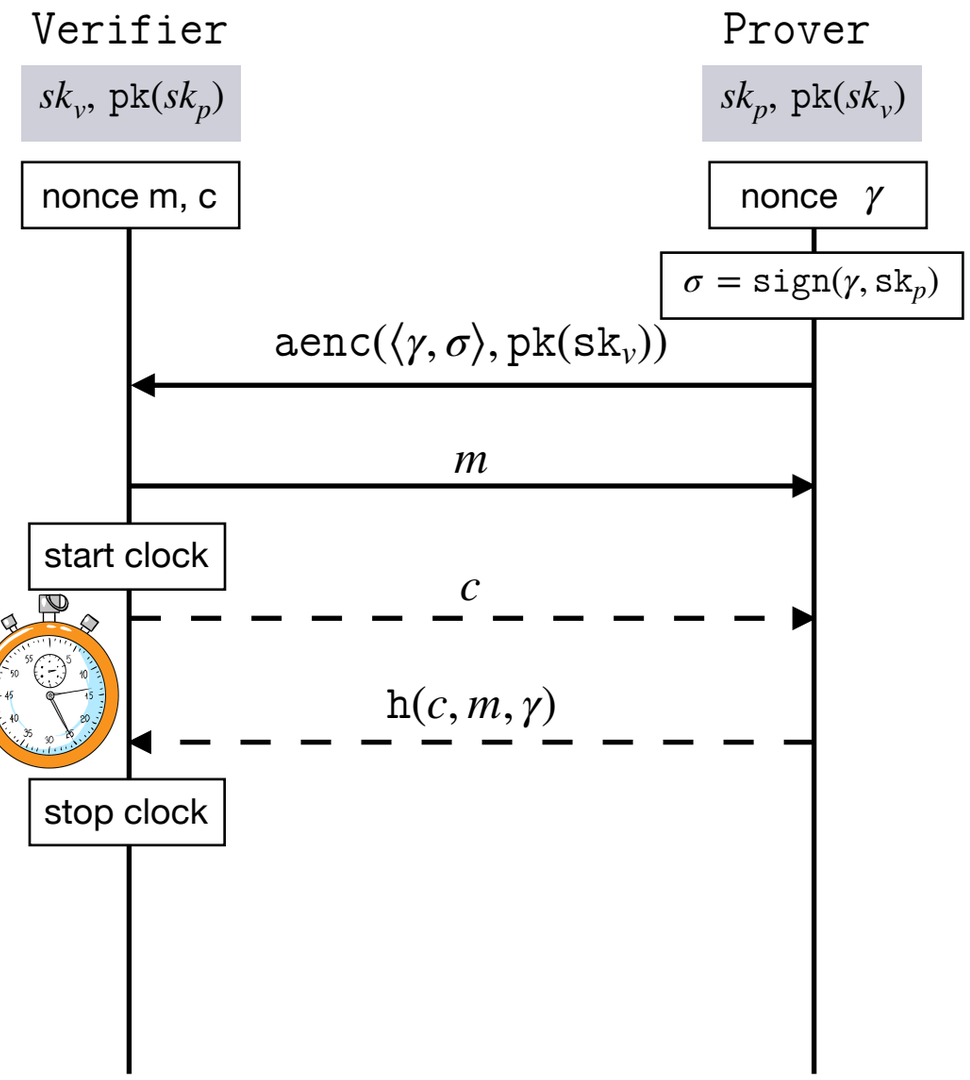


Example: TREAD - 2017

$$
\begin{aligned}
T_V = \quad & (v, \mathtt{in}(x)) . \\
& (v, [\mathtt{check}(t_\sigma, \mathtt{pk}(\mathtt{sk}_p)) = \mathtt{ok}]) . \\
& (v, [t_\gamma = \mathtt{getmsg}(t_\sigma)]) . \\
& (v, \mathtt{out}(m)) . \\
& (v, \mathtt{out}^{z_1}(c)) . \\
& (v, \mathtt{in}^{z_2}(y)) . \\
& (v, [\,|z_2 - z_1 < 2 \times t_0|\,]) . \\
& (v, [y = h(c, m, t_\gamma)])
\end{aligned}
$$

with $t_\sigma = \mathtt{proj}_2(\mathtt{adec}(x, \mathtt{sk}_v))$

$t_\gamma = \mathtt{proj}_1(\mathtt{adec}(x, \mathtt{sk}_v))$

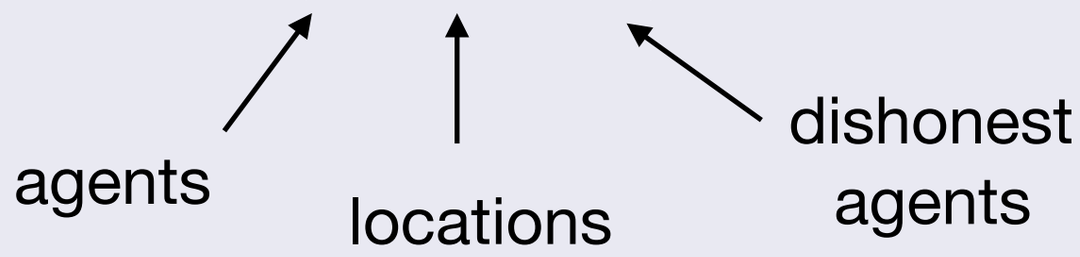


The protocol TREAD is the set of **all the possible interleaving** of T_V and T_P (the trace corresponding to the prover)

Topology and Configuration

Topology

A **topology** is a tuple $\mathscr{T} = (\mathscr{A}, \mathtt{Loc}, \mathscr{M})$.

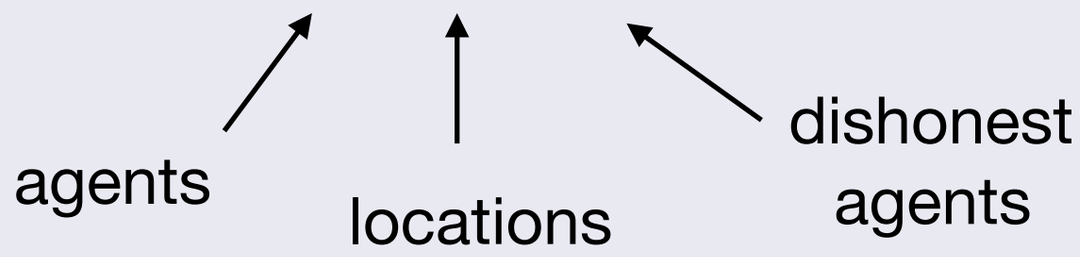


We define $\mathtt{Dist}_{\mathscr{T}}(a, b) = \dfrac{\|\mathtt{Loc}(a) - \mathtt{Loc}(b)\|}{c}$

Topology and Configuration

Topology

A **topology** is a tuple $\mathscr{T} = (\mathscr{A}, \mathtt{Loc}, \mathscr{M})$.



We define $\mathtt{Dist}_{\mathscr{T}}(a, b) = \dfrac{\|\mathtt{Loc}(a) - \mathtt{Loc}(b)\|}{c}$

Configuration

A **configuration** is a tuple $(T; \Phi; t)$ where:

- T is a trace
- $\Phi = \{\mathtt{w}_1 \xrightarrow{a_1, t_1} m_1, \ldots, \mathtt{w}_n \xrightarrow{a_n, t_n} m_n\}$ is a frame
- $t \in \mathscr{R}_+$ is the global time

Semantics

TIME $(T;\Phi;t) \longrightarrow_{\mathscr{T}_0} (T;\Phi;t+\delta)$ with $\delta \geq 0$

TEST $((a,[\,|\,t_1 \sim t_2\,|\,])\,.\,T;\Phi;t) \xrightarrow{a,\mathtt{test}}_{\mathscr{T}_0} (T;\Phi;t)$ when $t_1 \sim t_2$ is true

EQ $((a,[u=v])\,.\,T;\Phi;t) \xrightarrow{a,\mathtt{eq}}_{\mathscr{T}_0} (T;\Phi;t)$ when $u\downarrow = v\downarrow$

LET $((a,[z:=v])\,.\,T;\Phi;t) \xrightarrow{a,\mathtt{let}(v\downarrow)}_{\mathscr{T}_0} (T\{z \mapsto v\downarrow\};\Phi;t)$ when $v\downarrow \in \mathbb{R}^+$

OUT $((a,\mathtt{out}^z(u))\,.\,T;\Phi;t) \xrightarrow{a,\mathtt{out}(u\downarrow)}_{\mathscr{T}_0} (T\{z \mapsto t\}\,;\Phi \uplus \{\mathtt{w} \xrightarrow{a,t} u\downarrow\};t)$

Semantics

TIME $(T;\Phi;t) \longrightarrow_{\mathscr{T}_0} (T;\Phi;t+\delta)$ with $\delta \geq 0$

TEST $((a,[\,|\,t_1 \sim t_2\,|\,]).T;\Phi;t) \xrightarrow{a,\mathtt{test}}_{\mathscr{T}_0} (T;\Phi;t)$ when $t_1 \sim t_2$ is true

EQ $((a,[u=v]).T;\Phi;t) \xrightarrow{a,\mathtt{eq}}_{\mathscr{T}_0} (T;\Phi;t)$ when $u\downarrow = v\downarrow$

LET $((a,[z:=v]).T;\Phi;t) \xrightarrow{a,\mathtt{let}(v\downarrow)}_{\mathscr{T}_0} (T\{z \mapsto v\downarrow\};\Phi;t)$ when $v\downarrow \in \mathbb{R}^+$

OUT $((a,\mathtt{out}^z(u)).T;\Phi;t) \xrightarrow{a,\mathtt{out}(u\downarrow)}_{\mathscr{T}_0} (T\{z \mapsto t\}\,;\Phi \uplus \{\mathtt{w} \xrightarrow{a,t} u\downarrow\};t)$

IN $((a,\mathtt{in}^z(x)).T;\Phi;t) \xrightarrow{a,\mathtt{in}(u)}_{\mathscr{T}_0} (T\{x \mapsto u, z \mapsto t\};\Phi;t)$

if $\exists b \in \mathscr{A}, t_b \in \mathscr{R}_+$ such that $t_b \leq t - \mathtt{Dist}_{\mathscr{T}_0}(b,a)$ and:

- if $b \notin \mathscr{M}$ then $u \in img(\lfloor\Phi\rfloor_b^{t_b}) \cup \mathscr{N}_{\mathtt{pub}} \cup \mathbb{R}^+$
- if $b \in \mathscr{M}$ then u is deducible from $\bigcup_{c\in\mathscr{A}} \lfloor\Phi\rfloor_c^{t_b - \mathtt{Dist}_{\mathscr{T}_0}(c,b)}$

Problem under study

Reachability property

A reachability property is a property that can be encoded in the following decision problem:

Input: A trace T and topology $\mathscr{T}_0$

Output: Do there exist $l_1, \ldots, l_n$, Φ and t such that

$$(T, \varnothing, t_0) \xrightarrow{l_1,\ldots,l_n}_{\mathscr{T}_0} (\epsilon; \Phi; t)?$$

Distance bounding protocols:
Resistance against Distance frauds, Mafia frauds, Distance hijacking attacks is equivalent to: given a topology,

"can a verifier end a session talking with a remote prover?"

Table of contents

Main issue

$$T = (a_1, \mathtt{out}(k)) \,.\, (a_2, \mathtt{out}(k)) \,.\, (b, \mathtt{in}^z(x)) \,.\, (b, [x = k]) \,.\, (b, [\,|\, z < 2 \,|\,])$$

With $\mathtt{Dist}_{\mathcal{T}}(a_1, b) = 10$

$\mathtt{Dist}_{\mathcal{T}}(a_2, b) = 1$

Main issue

$$T = (a_1, \mathtt{out}(k)) \,.\, (a_2, \mathtt{out}(k)) \,.\, (b, \mathtt{in}^z(x)) \,.\, (b, [x = k]) \,.\, (b, [\,|\, z < 2 \,|\,])$$

With $\mathtt{Dist}_{\mathcal{T}}(a_1, b) = 10$

$\mathtt{Dist}_{\mathcal{T}}(a_2, b) = 1$

Is the configuration $(T; \varnothing, 0)$ executable?

Main issue

$$T = (a_1, \texttt{out}(k)) \,.\, (a_2, \texttt{out}(k)) \,.\, (b, \texttt{in}^z(x)) \,.\, (b, [x = k]) \,.\, (b, [\,|\, z < 2 \,|\,])$$

With $\texttt{Dist}_{\mathcal{T}}(a_1, b) = 10$

$\texttt{Dist}_{\mathcal{T}}(a_2, b) = 1$

Is the configuration $(T; \varnothing, 0)$ executable?

Yes **but only** considering $\texttt{w}_2$ as an input recipe.

Main issue

$$T = (a_1, \mathtt{out}(k)) \,.\, (a_2, \mathtt{out}(k)) \,.\, (b, \mathtt{in}^z(x)) \,.\, (b, [x = k]) \,.\, (b, [\,|\, z < 2 \,|\,])$$

With $\mathtt{Dist}_{\mathscr{T}}(a_1, b) = 10$

$\mathtt{Dist}_{\mathscr{T}}(a_2, b) = 1$

Is the configuration $(T; \varnothing, 0)$ executable?

Yes **but only** considering $\mathtt{w}_2$ as an input recipe.

Issue:

The Akiss procedure will arbitrary drop one of the two outputs deducing the same term…

Akiss procedure

Input: two traces T and P
Output: check if $T \sqsubseteq P$

Akiss procedure

Input: two traces T and P
Output: check if $T \sqsubseteq P$

Step 1: compute the seed statements for the trace T

$$r_{(a_1,\mathtt{out}(k)).(a_2,\mathtt{out}(k)).(b,\mathtt{in}(k)).(b,\mathtt{eq}).(b,\mathtt{test})} \Leftarrow \mathtt{k}_{(a_1,\mathtt{out}(k)).(a_2,\mathtt{out}(k))}(X, k)$$

Akiss procedure

Input: two traces T and P
Output: check if $T \sqsubseteq P$

Step 1: compute the seed statements for the trace T

$$r_{(a_1,\mathtt{out}(k)).(a_2,\mathtt{out}(k)).(b,\mathtt{in}(k)).(b,\mathtt{eq}).(b,\mathtt{test})} \Leftarrow \mathtt{k}_{(a_1,\mathtt{out}(k)).(a_2,\mathtt{out}(k))}(X, k)$$

Step 2: compute the saturated knowledge applying resolution rules

- finite set of **simple** Horn clauses

$$r_w \Leftarrow \mathtt{k}_{w_1}(X_1, x_1), \ldots, \mathtt{k}_{w_n}(X_1, x_n)$$

Akiss procedure

Input: two traces T and P
Output: check if $T \sqsubseteq P$

Step 1: compute the seed statements for the trace T

$$r_{(a_1,\mathtt{out}(k)).(a_2,\mathtt{out}(k)).(b,\mathtt{in}(k)).(b,\mathtt{eq}).(b,\mathtt{test})} \Leftarrow \mathtt{k}_{(a_1,\mathtt{out}(k)).(a_2,\mathtt{out}(k))}(X, k)$$

Step 2: compute the saturated knowledge applying resolution rules

- finite set of **simple** Horn clauses

$$r_w \Leftarrow \mathtt{k}_{w_1}(X_1, x_1), \ldots, \mathtt{k}_{w_n}(X_1, x_n)$$

Step 3: check the trace inclusion

- for all the reachability statement in the saturated knowledge base, Check that it can be executed in P

Our procedure

Input: a trace and a topology
Output: is_secure/attack

Step 2: compute the saturated knowledge base

- finite set of **simple** Horn clauses
- dropping less clauses than Akiss does, to keep completeness

Step 3: check the timing constraints

- for all the witnesses of attack (i.e. a reach predicate) in the saturated knowledge base:
  - compute all the recipes that can fill the inputs
  - check the corresponding timing constraints

Our procedure

Input: a trace and a topology
Output: is_secure/attack

Step 1: compute the seed statements for the trace T

$$r_{(a_1,\mathtt{out}(k)).(a_2,\mathtt{out}(k)).(b,\mathtt{in}(k)).(b,\mathtt{eq}).(b,\mathtt{test})} \Leftarrow \mathtt{k}_{(a_1,\mathtt{out}(k)).(a_2,\mathtt{out}(k))}(X, k)$$

Step 2: compute the saturated knowledge base

- finite set of **simple** Horn clauses
- dropping less clauses than Akiss does, to keep completeness

Step 3: check the timing constraints

- for all the witnesses of attack (i.e. a reach predicate) in the saturated knowledge base:
  - compute all the recipes that can fill the inputs
  - check the corresponding timing constraints

Horn clause

$$H \Leftarrow \mathtt{k}_{w_1}(X_1, u_1), \ldots, \mathtt{k}_{w_n}(X_1, u_n) \quad \text{with } H \in \{\mathtt{r}_{w_0}, \mathtt{k}_{w_0}(R, u)\}$$

Horn clause

$$H \Leftarrow \mathtt{k}_{w_1}(X_1, u_1), \ldots, \mathtt{k}_{w_n}(X_1, u_n) \quad \text{with } H \in \{\mathtt{r}_{w_0}, \mathtt{k}_{w_0}(R, u)\}$$

- reach: $(T_0; \Phi_0) \vdash \mathtt{r}_{l_1,\ldots,l_n}$ if there exists $(T_n; \Phi_n)$ such that $(T_0; \Phi_0) \xrightarrow{l_1, \ldots, l_n} (T_n; \Phi_n)$

Horn clause

$$H \Leftarrow \mathtt{k}_{w_1}(X_1, u_1), \ldots, \mathtt{k}_{w_n}(X_1, u_n) \quad \text{with } H \in \{\mathtt{r}_{w_0}, \mathtt{k}_{w_0}(R, u)\}$$

- reach: $(T_0; \Phi_0) \vDash \mathtt{r}_{l_1,\ldots,l_n}$ if there exists $(T_n; \Phi_n)$ such that $(T_0; \Phi_0) \xrightarrow{l_1, \ldots, l_n} (T_n; \Phi_n)$
- knows: $(T_0; \Phi_0) \vDash \mathtt{k}_{l_1,\ldots,l_n}(R, u)$ if for all $(T_n; \Phi_n)$ such that $(T_0; \Phi_0) \xrightarrow{l_1, \ldots, l_n} (T_n; \Phi_n)$

  we have that $R\Phi_n \downarrow = u$

Horn clause

$$H \Leftarrow \mathtt{k}_{w_1}(X_1, u_1), \ldots, \mathtt{k}_{w_n}(X_1, u_n) \quad \text{with } H \in \{\mathtt{r}_{w_0}, \mathtt{k}_{w_0}(R, u)\}$$

- <u>reach:</u> $(T_0; \Phi_0) \vDash \mathtt{r}_{l_1,\ldots,l_n}$ if there exists $(T_n; \Phi_n)$ such that $(T_0; \Phi_0) \xrightarrow{l_1, \ldots, l_n} (T_n; \Phi_n)$
- <u>knows:</u> $(T_0; \Phi_0) \vDash \mathtt{k}_{l_1,\ldots,l_n}(R, u)$ if for all $(T_n; \Phi_n)$ such that $(T_0; \Phi_0) \xrightarrow{l_1, \ldots, l_n} (T_n; \Phi_n)$

  we have that $R\Phi_n \downarrow = u$

This semantics is given w.r.t. an untimed semantics of traces!

Saturation

Step 2: compute the saturated knowledge base

- finite set of **simple** Horn clauses
- dropping less clauses than Akiss does, to keep completeness

$$\text{Resolution}\ \frac{f : H \Leftarrow \mathtt{k}_w(X,t), B_1, \ldots, B_n \in K \qquad t \notin \mathcal{X} \qquad g : \mathtt{k}_{w'}(R',t') \Leftarrow B_{n+1}, \ldots, B_m \in \mathtt{solved}(K) \qquad \sigma = \mathtt{mgu}(\mathtt{k}_w(X,t), \mathtt{k}_{w'}(R',t'))}{K := K \Cup (H \Leftarrow B_1, \ldots, B_n, B_{n+1}, \ldots, B_m)\sigma}$$

New update function

1. Do not add the statement if the term in the head is reduced to a variable
2. Simplify the statement i.e. if $B_i = \mathtt{k}_w(X_i, u)$ and $B_j = \mathtt{k}_w(X_j, u)$ keep only one

Saturation

Soundness of the saturation

Given a trace T, we denote $K = \texttt{solved}(\texttt{sat}(\texttt{seed}(T)))$ **and**

$(T, \varnothing) \vDash g$ for any $g \in K \cup \mathscr{H}(K)$

Saturation

Soundness of the saturation

Given a trace T, we denote $K = \texttt{solved}(\texttt{sat}(\texttt{seed}(T)))$ **and**

$$(T, \varnothing) \vdash g \quad \text{for any} \quad g \in K \cup \mathscr{H}(K)$$

Completeness of the saturation

Given a trace T, we denote $K = \texttt{solved}(\texttt{sat}(\texttt{seed}(T)))$ **and**
if $(T, \varnothing) \xrightarrow{l_1,\ldots,l_n}_{\mathscr{T}} (S, \phi)$ **with asap input recipes then**

- $\texttt{r}_{l_1.\ldots.l_n} \in \mathscr{H}(K)$
- **if** $R\phi\downarrow = u$ **for some** R **asap then** $\texttt{k}_{l_1.\ldots.l_p}(R, u) \in \mathscr{H}(K)$

Lemma

Given an execution $(T, \varnothing) \xrightarrow{l_1,\ldots,l_n}_{\mathscr{T}} (S, \phi)$ **w.l.o.g. we can assume that it only involves asap input recipes.**

Algorithm

1 : **procedure** Reachability$(T, t_0, \mathscr{T}_0)$
2 : $K := \mathtt{solved}(\mathtt{sat}(\mathtt{seed}(T)))$
3 : **for all** $\mathtt{r}_{l_1,\ldots,l_n} \Leftarrow \mathtt{k}_{w_1}(X_1, x_1), \ldots, \mathtt{k}_{w_m}(X_m, x_m) \in K$ **do**
4 : let $c_1, \ldots, c_q$ be fresh public names such that
5 : $\rho : vars(l_1, \ldots, l_n) \to \{c_1, \ldots, c_k\}$ is a bijection
6 : **for all** $i \in \mathtt{Rcv}(n)$ **do**
7 : **if** $l_i = (a_i, \mathtt{in}(v_i))$ **then** $\overline{L_i} = \{R \mid \mathtt{k}_{l_1\rho,\ldots,r_n\rho}(R, v_i\rho) \in \mathscr{H}(K)\}$
8 : **end for**
9 : let $\{i_1, \ldots, i_p\} = \mathtt{Rcv}(n)$
10 : **for all** $R_{i_1}, \ldots, R_{i_p} \in \overline{L_{i_1}}, \ldots, \overline{L_{i_p}}$ **do**
11 : let $\psi = \mathtt{Timing}((T, \varnothing, t_0), R_{i_1} \ldots R_{i_p}, l_1, \ldots, l_n)$
12 : **if** ψ satisfiable **then return** true **end if**
13 : **end for**
14 : **end for**
15 : **return** false
16 : **end procedure**

Procedure

Soundness and completeness of the procedure

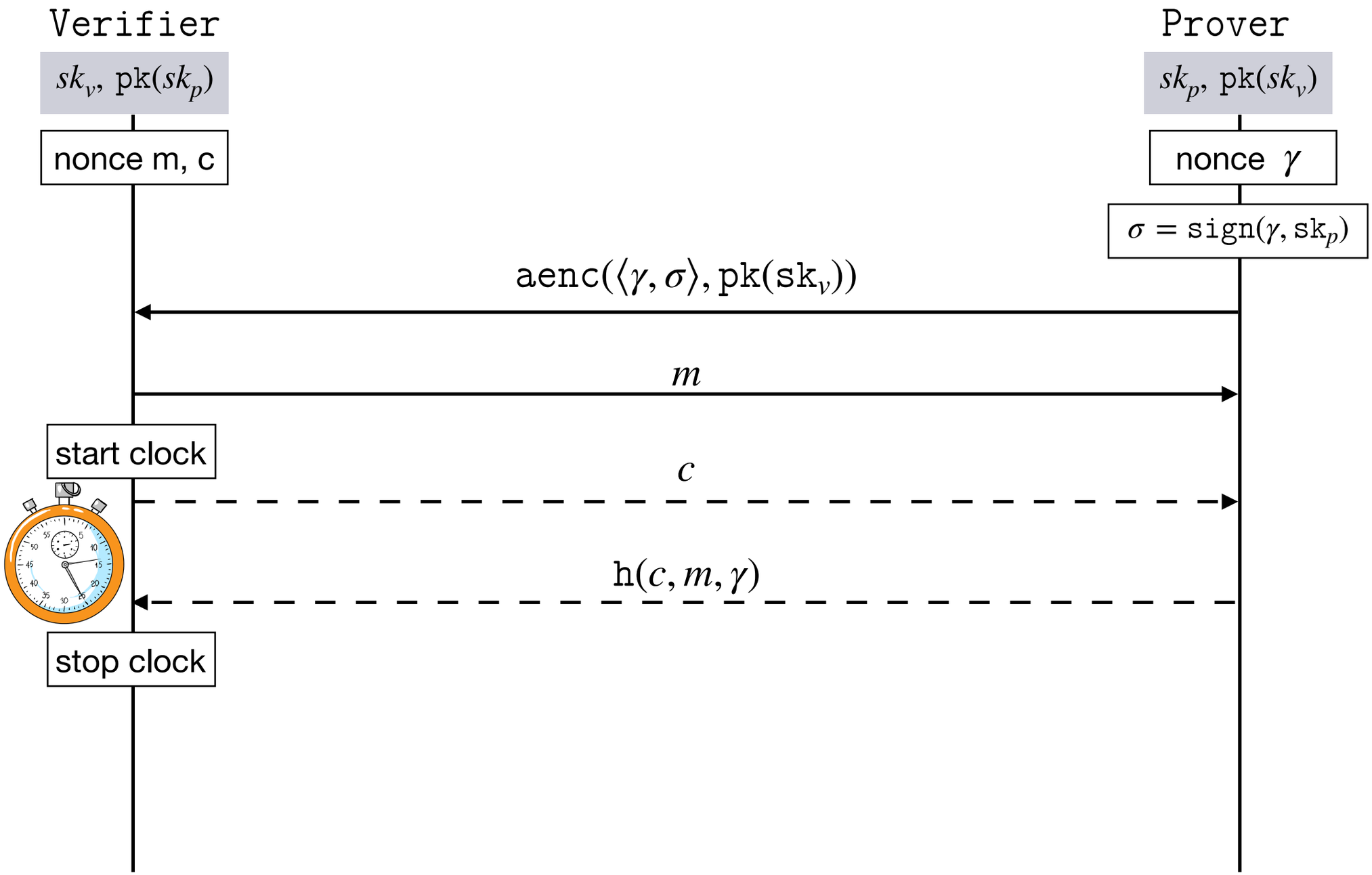
Given a trace T:

- If our procedure returns "attack" then there exists $l_1, \ldots, l_n, \Phi_n, t_n$ such that:$(T, \varnothing) \xrightarrow{l_1,\ldots,l_n} (\epsilon, \Phi_n)$
- If our procedure returns "is_secure" then the empty trace ϵ is not reachable.

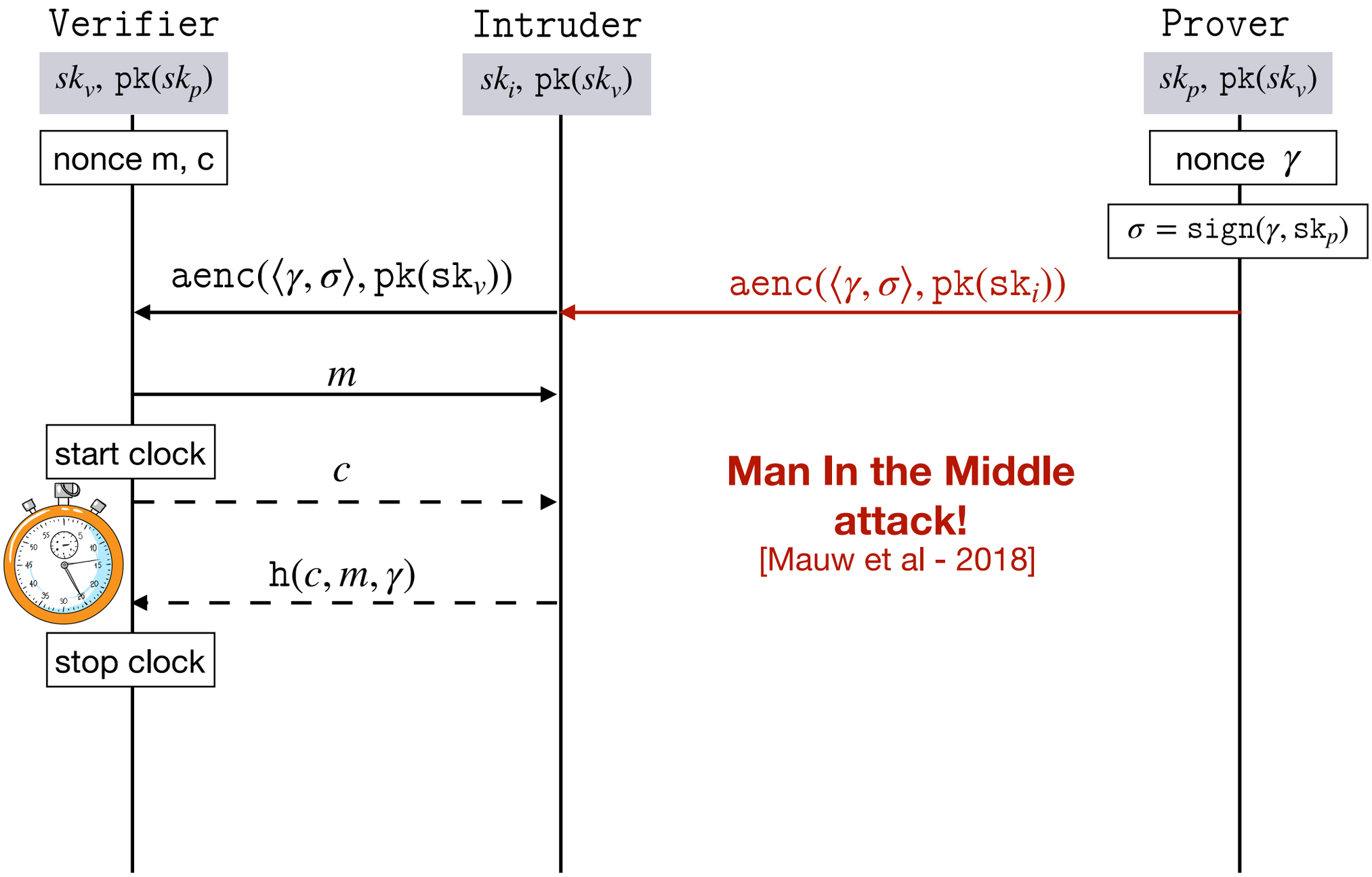
Limitation: we did not proved the termination of our procedure.

Table of contents

TREAD - 2017


Verifier
sk_v, $\mathtt{pk}(sk_p)$
nonce m, c
Prover
sk_p, $\mathtt{pk}(sk_v)$
nonce γ
$\sigma = \mathtt{sign}(\gamma, \mathtt{sk}_p)$
$\mathtt{aenc}(\langle\gamma, \sigma\rangle, \mathtt{pk}(\mathtt{sk}_v))$
m
start clock
c
$\mathtt{h}(c, m, \gamma)$
stop clock

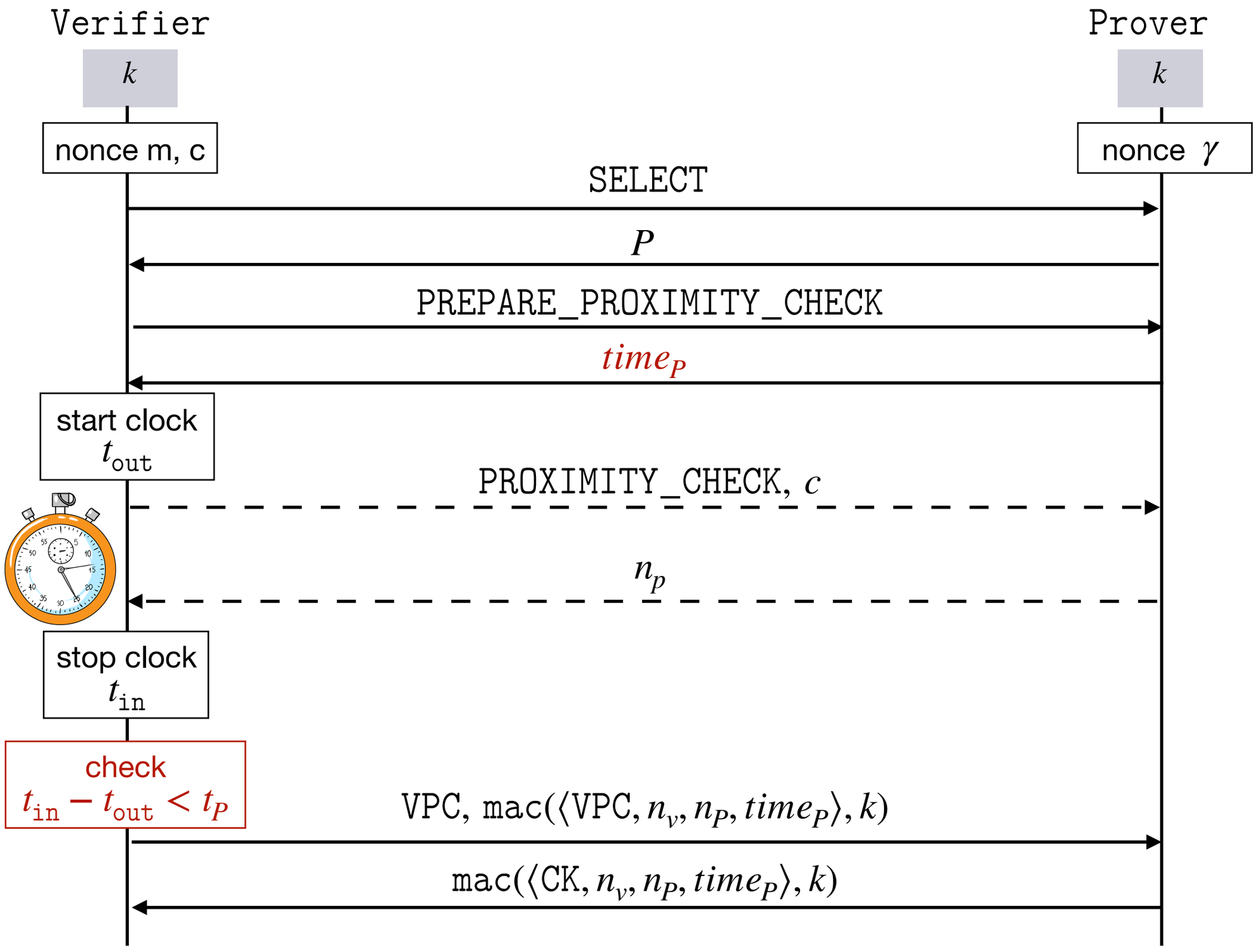
TREAD - 2017



Man In the Middle attack!

[Mauw et al - 2018]

NXP protocol - 2017



Case studies

Protocols	MF			DH		
	roles/tr	time	status			
TREAD-Asymmetric	2/-	1s	✗	2/-	1s	✗
SPADE	2/-	2s	✗	2/-	4s	✗
TREAD-Symmetric	4/7500	18min	✓	2/-	1s	✗
Brands and Chaum	4/5635	37min	✓	2/-	1s	✗
Swiss-Knife	3/1080	25s	✓	3/7470	4min	✓
Hancke and Kuhn	2/20	1s	✓	2/20	1s	✓
	4/3368	58s	✓	4/3368	47s	✓
	5/30240	14min	✓	5/30240	12min	✓
NXP	2/126	4s	✓			
Mastercard RRP	2/35	6min	✓			
PaySafe	2/4	5min	✓			

(✗ : attack found, ✓ : proved secure)

Conclusion

As expected we have adapted the underlying procedure of Akiss to propose an **exact procedure** to analyze distance bounding protocols. We have **successfully applied** it to many protocols.

Future work

- Improve the POR optimization to limit the interleaving blow up
- Improve the update function to avoid non-termination issues
- Adapt this procedure to model XOR