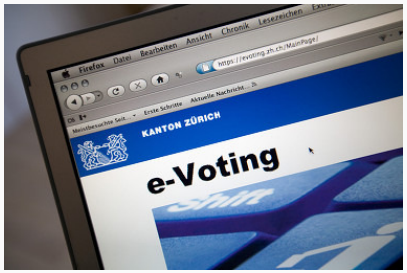
A small bound on the number of sessions for security protocols

Véronique CORTIER, Antoine DALLON, **Stéphanie DELAUNE**

CSF 2022 - August 10, 2022

Security protocols are everywhere !



Cryptographic protocols

Distributed programs which aims at providing some security properties relying on cryptography.

Security properties

Two main families of security properties:

- Reachability properties, *e.g.* weak secrecy, authentication, ...

  Is it possible to reach a bad state?

- Equivalence properties, *e.g.* anonymity, unlinkability, ...

  Is the attacker able to distinguish between two situations?

$\longrightarrow$ In this work, we consider both kinds of security properties.

Verification of security protocols

What do we know?

- A problem well-known to be undecidable when considering an unbounded number of sessions. [Durgin *et al.*,99]
- Decision procedures and tools (*e.g.* DeepSec, SAT-Equiv, . . .) dedicated to a bounded number of sessions exist.
  [Rusinowitch & Turuani, CSFW'01]

Observation:
known attacks only involve very few sessions (less than 6)

Long-standing research question

Identify criteria to bound a priori the number of sessions involved in an attack.

Some related work

Reachability properties (mainly weak secrecy):

- [Lowe, CSFW'98]: one honest agent for each role
  $\longrightarrow$ typed model, no blind copy, no temporary secret
- [Froschle, POST'15]: leakiness is decidable for well-founded protocols
  $\longrightarrow$ typed model, excludes protocols with temporary secret
- [D'Osualdo, CSF'17]: the case of depth-bounded processes relying on the theory of WSTS

Equivalence properties:

- [Chrétien *et al*, CSF'15 & JAR'21]: simple protocols without else branches, classical primitives
  $\longrightarrow$ the resulting bound is far to be practical (*e.g.* 10^{19})

Our contributions

Main result

A practical bound allowing us to rely on some existing tools to perform the security analysis.

1. We prove the correctness of our bound assuming some hypotheses (type-compliance and acyclic dependencies).
2. We implement our algorithm in HowMany to compute and generate the scenarios that need to be analysed.
3. We perform various case studies on protocols from the literature using Howmany + DeepSec/SAT-Equiv.

$\longrightarrow$ This result holds for both for **reachability** and **equivalence properties**.

Part I - Reachability properties

1. Modelling messages and protocols
2. Our main assumptions
3. Our algorithm to compute and generate the scenarios
4. Some case studies

Modelling messages

Cryptographic operations are modelled using function symbols ...

Example: $\Sigma = \Sigma_c \cup \Sigma_d \cup \Sigma_{test}$

$$\Sigma_c = \{\mathsf{aenc},\ \mathsf{pk},\ \mathsf{sign},\ \mathsf{vk},\ \mathsf{ok},\ \langle\ \rangle\}$$

$$\Sigma_d = \{\mathsf{adec},\ \mathsf{getmsg},\ \mathsf{proj}_1,\ \mathsf{proj}_2\},\ \text{and}\ \Sigma_{test} = \{\mathsf{check}\}.$$

... and their properties using a rewriting system.

Example

$$\begin{array}{rcl@{\qquad}rcl} \mathsf{adec}(\mathsf{aenc}(x,\mathsf{pk}(y)),y) & \to & x & \mathsf{proj}_1(\langle x,y\rangle) & \to & x \\ \mathsf{getmsg}(\mathsf{sign}(x,y)) & \to & x & \mathsf{proj}_2(\langle x,y\rangle) & \to & y \end{array}$$

$$\mathsf{check}(\mathsf{sign}(x,y),\mathsf{vk}(y)) \to \mathsf{ok}$$

$\longrightarrow$ we consider shaped rewriting systems allowing one to model most classical primitives.

Modelling protocols

Protocols are modelled using a process algebra.

$$
\begin{array}{lll}
P, Q := & & \\
\quad | & \mathsf{in}^{\alpha}(c, u).P & \text{input} \\
\quad | & \mathsf{out}^{\alpha}(c, u).P & \text{output} \\
\quad | & \ldots & \\
\quad | & !P & \text{replication} \\
\quad | & \mathsf{match}\, x\, \mathsf{with}\, (u_1 \to P_1 \mid \ldots \mid u_j \to P_j) & \text{filtering}
\end{array}
$$

Modelling protocols

Protocols are modelled using a process algebra.

$$
\begin{array}{lll}
P, Q := & & \\
\quad | & \mathsf{in}^{\alpha}(c, u).P & \text{input} \\
\quad | & \mathsf{out}^{\alpha}(c, u).P & \text{output} \\
\quad | & \ldots & \\
\quad | & !P & \text{replication} \\
\quad | & \mathsf{match}\ x\ \mathsf{with}\ (u_1 \to P_1 \mid \ldots \mid u_j \to P_j) & \text{filtering}
\end{array}
$$

Semantics

$$(\mathsf{in}^{\alpha}(c, u).P \uplus \mathcal{P}; \phi; \sigma) \xrightarrow{\mathsf{in}^{\alpha}(c,R)} (P \uplus \mathcal{P}; \phi; \sigma \uplus \sigma_0)$$

where R is a recipe such that $R\phi\downarrow$ is a message, and $R\phi\downarrow = (u\sigma)\sigma_0$

...

Example considering a very simple scenario

Denning Sacco protocol

$$A \rightarrow B : \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(ekb))$$

$$P_{\mathsf{DS}} = \left\{ \begin{array}{ll} & !\,\mathsf{new}\, k.\mathsf{out}^{\alpha_1}(c, \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(ekb))) \\ \mid & !\,\mathsf{in}^{\beta_1}(c, \mathsf{aenc}(\mathsf{sign}(x, ska), \mathsf{pk}(ekb))). \\ \mid & \mathsf{out}^{\gamma_1}(c, \mathsf{pk}(ekb)).\mathsf{out}^{\gamma_2}(c, \mathsf{vk}(ska)). \end{array} \right.$$

Example considering a very simple scenario

Denning Sacco protocol

$$A \to B : \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(ekb))$$

$$P_{\mathsf{DS}} = \left\{ \begin{array}{ll} & !\,\mathsf{new}\, k.\mathsf{out}^{\alpha_1}(c, \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(ekb))) \\ \mid & !\,\mathsf{in}^{\beta_1}(c, \mathsf{aenc}(\mathsf{sign}(x, ska), \mathsf{pk}(ekb))).\boxed{\mathsf{in}^{\beta_2}(c, x)} \\ \mid & \mathsf{out}^{\gamma_1}(c, \mathsf{pk}(ekb)).\mathsf{out}^{\gamma_2}(c, \mathsf{vk}(ska)). \end{array} \right.$$

Secrecy property

To analyse secrecy of the key as received by B, it suffices to add $\boxed{\mathsf{in}^{\beta_2}(c, x)}$, and to check whether β_2 is reachable.

Hypothesis: type-compliance

We give types to any atomic data, and types are then extended to arbitrary terms:

$$\delta(\mathsf{f}(t_1, \ldots, t_n)) = \mathsf{f}(\delta(t_1), \ldots, \delta(t_n)).$$

$\longrightarrow$ the typing system is thus structure preserving.

Hypothesis: type-compliance

We give types to any atomic data, and types are then extended to arbitrary terms:

$$\delta(\mathsf{f}(t_1, \ldots, t_n)) = \mathsf{f}(\delta(t_1), \ldots, \delta(t_n)).$$

$\longrightarrow$ the typing system is thus structure preserving.

Going back to Dening Sacco

We consider $\Delta = \{\tau_{ska}, \tau_{ekb}, \tau_{ekc}, \tau_k\}$.

- $\delta(k) = \delta(x) = \tau_k$,
- $\delta(ska) = \tau_{ska}$, $\delta(skb) = \tau_{skb}$, $\delta(ekc) = \tau_{ekc}$.

$\rightarrow$ $\delta(\mathsf{aenc}(\mathsf{sign}(x, ska), \mathsf{pk}(ekb))) = \mathsf{aenc}(\mathsf{sign}(\tau_k, \tau_{ska}), \mathsf{pk}(\tau_{ekb}))$

Hypothesis: type-compliance

We give types to any atomic data, and types are then extended to arbitrary terms:

$$\delta(\mathsf{f}(t_1, \ldots, t_n)) = \mathsf{f}(\delta(t_1), \ldots, \delta(t_n)).$$

$\longrightarrow$ the typing system is thus structure preserving.

Going back to Dening Sacco

We consider $\Delta = \{\tau_{ska}, \tau_{ekb}, \tau_{ekc}, \tau_k\}$.

- $\delta(k) = \delta(x) = \tau_k$,
- $\delta(ska) = \tau_{ska}$, $\delta(skb) = \tau_{skb}$, $\delta(ekc) = \tau_{ekc}$.

$\rightarrow$ $\delta(\mathsf{aenc}(\mathsf{sign}(x, ska), \mathsf{pk}(ekb))) = \mathsf{aenc}(\mathsf{sign}(\tau_k, \tau_{ska}), \mathsf{pk}(\tau_{ekb}))$

A protocol P is type-compliant w.r.t. a typing system (Δ, δ) if any pair of unifiable encrypted subterms have the same type.

Main result

Theorem

Let P be a protocol type-compliant, and α be a label. If $P \xrightarrow{\ell_1} \ldots \xrightarrow{\ell_n} \mathcal{K}$ with $\mathsf{Label}(\ell_n) = \alpha$ then there exists $A \in \mathsf{dep}(\alpha)$ such that:

- $P \xrightarrow{\ell'_1} \ldots \xrightarrow{\ell'_n} \mathcal{K}'$ with $\mathsf{Label}(\ell'_n) = \alpha$, and
- $\mathsf{Label}(\ell'_1.\ldots.\ell'_n) \subseteq A$.

$\longrightarrow$ $\mathsf{dep}(\alpha)$ computes an upper bound of the actions/labels that need to be considered to reach some action labelled α.

Main result

Theorem

Let P be a protocol type-compliant, and α be a label. If $P \xrightarrow{\ell_1} \ldots \xrightarrow{\ell_n} \mathcal{K}$ with $\mathsf{Label}(\ell_n) = \alpha$ then there exists $A \in \mathsf{dep}(\alpha)$ such that:

- $P \xrightarrow{\ell'_1} \ldots \xrightarrow{\ell'_n} \mathcal{K}'$ with $\mathsf{Label}(\ell'_n) = \alpha$, and
- $\mathsf{Label}(\ell'_1.\ldots.\ell'_n) \subseteq A$.

$\longrightarrow$ $\mathsf{dep}(\alpha)$ computes an upper bound of the actions/labels that need to be considered to reach some action labelled α.

Going back to Denning Sacco

We have that $\mathsf{dep}(\beta_2) = \{\{\beta_2, \beta_1, \alpha_1\}\}$.

$\longrightarrow$ if there is an attack, there is one involving at most 3 actions.

Computation of dep

$$\{A_1, A_2\} \otimes \{B_1, B_2\} = \{A_1 \uplus B_1,\ A_1 \uplus B_2,\ A_2 \uplus B_1,\ A_2 \uplus B_2\}$$

Computation of dep

$$\{A_1, A_2\} \otimes \{B_1, B_2\} = \{A_1 \uplus B_1,\ A_1 \uplus B_2,\ A_2 \uplus B_1,\ A_2 \uplus B_2\}$$

Sequential dependencies

- output: $\mathsf{dep}(\alpha) = \{\{\alpha\}\} \otimes \mathsf{dep}(\mathsf{pred}(\alpha))$
- input of type τ: $\mathsf{dep}(\alpha) = \{\{\alpha\}\} \otimes \mathsf{dep}(\mathsf{pred}(\alpha)) \otimes \mathsf{dep}(\tau)$.

Computation of dep

$$\{A_1, A_2\} \otimes \{B_1, B_2\} = \{A_1 \uplus B_1,\ A_1 \uplus B_2,\ A_2 \uplus B_1,\ A_2 \uplus B_2\}$$

Sequential dependencies

- output: $\mathsf{dep}(\alpha) = \{\{\alpha\}\} \otimes \mathsf{dep}(\mathsf{pred}(\alpha))$
- input of type τ: $\mathsf{dep}(\alpha) = \{\{\alpha\}\} \otimes \mathsf{dep}(\mathsf{pred}(\alpha)) \otimes \mathsf{dep}(\tau)$.

Message dependencies

$S_{\mathsf{out}}(\tau)$ explores all the possibilities to extract a term of type τ from the output occurring in $\delta(P)$ (protocol considering only types).

- atomic type: $\mathsf{dep}(\tau) = S_{\mathsf{out}}(\tau)$
- otherwise $\tau = \mathsf{f}(\tau_1, \ldots, \tau_k)$, we have that:

$$\mathsf{dep}(\tau) = S_{\mathsf{out}}(\tau) \cup \big(\mathsf{dep}(\tau_1) \otimes \ldots \otimes \mathsf{dep}(\tau_k)\big)$$

Computation of dep

$$\{A_1, A_2\} \otimes \{B_1, B_2\} = \{A_1 \uplus B_1,\ A_1 \uplus B_2,\ A_2 \uplus B_1,\ A_2 \uplus B_2\}$$

Sequential dependencies

- output: $\mathsf{dep}(\alpha) = \{\{\alpha\}\} \otimes \mathsf{dep}(\mathsf{pred}(\alpha))$
- input of type τ: $\mathsf{dep}(\alpha) = \{\{\alpha\}\} \otimes \mathsf{dep}(\mathsf{pred}(\alpha)) \otimes \mathsf{dep}(\tau)$.

Message dependencies

$S_{\mathsf{out}}(\tau)$ explores all the possibilities to extract a term of type τ from the output occurring in $\delta(P)$ (protocol considering only types).

- atomic type: $\mathsf{dep}(\tau) = S_{\mathsf{out}}(\tau) \cup \mathsf{dep}_{\mathsf{init}}(\tau)$
- otherwise $\tau = \mathsf{f}(\tau_1, \ldots, \tau_k)$, we have that:

$$\mathsf{dep}(\tau) = S_{\mathsf{out}}(\tau) \cup \big(\mathsf{dep}(\tau_1) \otimes \ldots \otimes \mathsf{dep}(\tau_k)\big) \cup \mathsf{dep}_{\mathsf{init}}(\tau)$$

with $\mathsf{dep}_{\mathsf{init}}(\tau) = \emptyset$ when τ is a cv-alien type, and $\{\emptyset\}$ otherwise.

Going back to Denning Sacco: $\mathsf{dep}(\beta_2)$

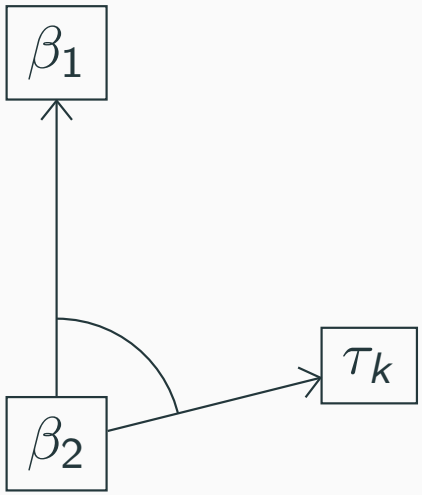
$$
\begin{aligned}
&P_{\mathsf{DS}} = \\
&!\,\mathsf{new}\ k. \\
&\mathsf{out}^{\alpha_1}(c, \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(ekb))) \\
&\mid !\,\mathsf{in}^{\beta_1}(c, \mathsf{aenc}(\mathsf{sign}(x, ska), \mathsf{pk}(ekb))). \\
&\quad \mathsf{in}^{\beta_2}(c, x) \\
&\mid \mathsf{out}^{\gamma_1}(c, \mathsf{pk}(ekb)). \\
&\quad \mathsf{out}^{\gamma_2}(c, \mathsf{vk}(ska)).
\end{aligned}
$$

τ $\quad \mathsf{dep}_{\mathsf{init}}(\tau) = \emptyset$

β_2

Going back to Denning Sacco: $\mathsf{dep}(\beta_2)$

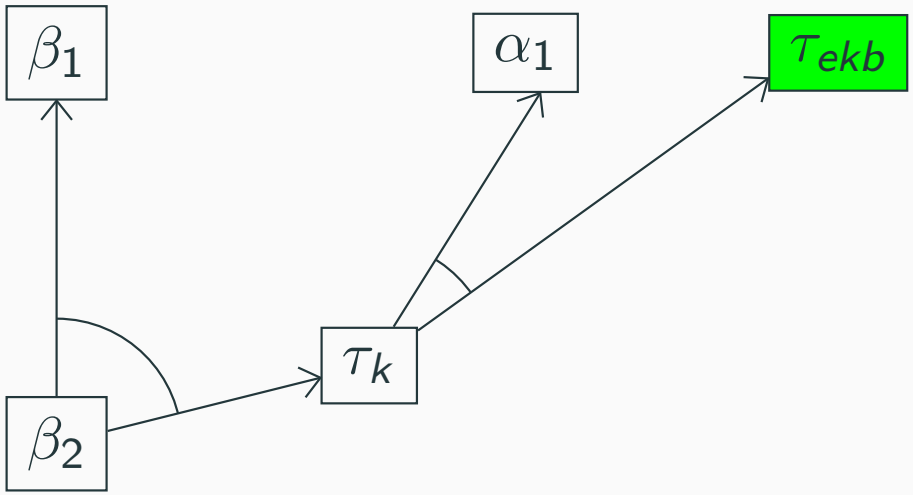
$$
\begin{aligned}
&P_{\mathsf{DS}} = \\
&!\,\mathsf{new}\ k. \\
&\mathsf{out}^{\alpha_1}(c, \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(ekb))) \\
&\mid\ !\,\mathsf{in}^{\beta_1}(c, \mathsf{aenc}(\mathsf{sign}(x, ska), \mathsf{pk}(ekb))). \\
&\quad\ \mathsf{in}^{\beta_2}(c, x) \\
&\mid\ \mathsf{out}^{\gamma_1}(c, \mathsf{pk}(ekb)). \\
&\quad\ \mathsf{out}^{\gamma_2}(c, \mathsf{vk}(ska)).
\end{aligned}
$$



τ $\quad \mathsf{dep}_{\mathsf{init}}(\tau) = \emptyset$

Going back to Denning Sacco: $\mathsf{dep}(\beta_2)$

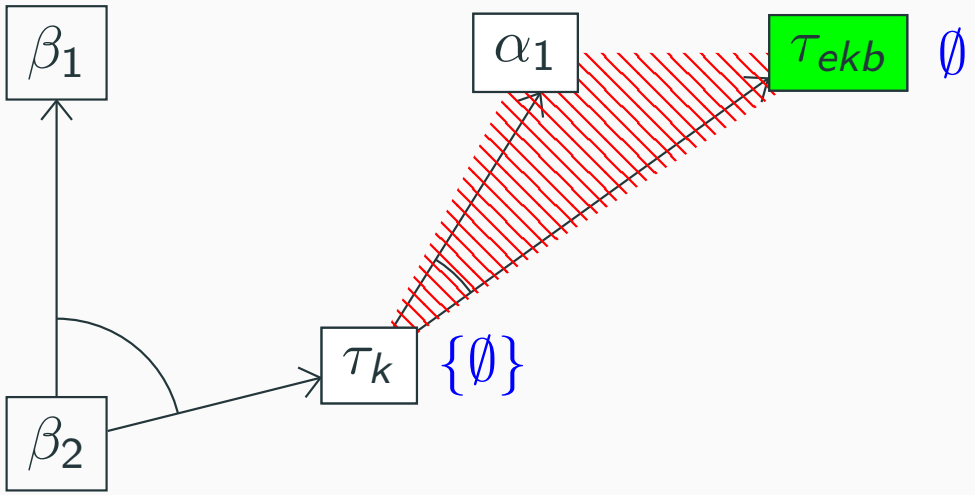
$$
\begin{aligned}
&P_{\mathsf{DS}} = \\
&!\,\mathsf{new}\ k. \\
&\mathsf{out}^{\alpha_1}(c, \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(ekb))) \\
&\mid !\,\mathsf{in}^{\beta_1}(c, \mathsf{aenc}(\mathsf{sign}(x, ska), \mathsf{pk}(ekb))). \\
&\quad \mathsf{in}^{\beta_2}(c, x) \\
&\mid \mathsf{out}^{\gamma_1}(c, \mathsf{pk}(ekb)). \\
&\quad \mathsf{out}^{\gamma_2}(c, \mathsf{vk}(ska)).
\end{aligned}
$$



τ $\quad \mathsf{dep}_{\mathsf{init}}(\tau) = \emptyset$

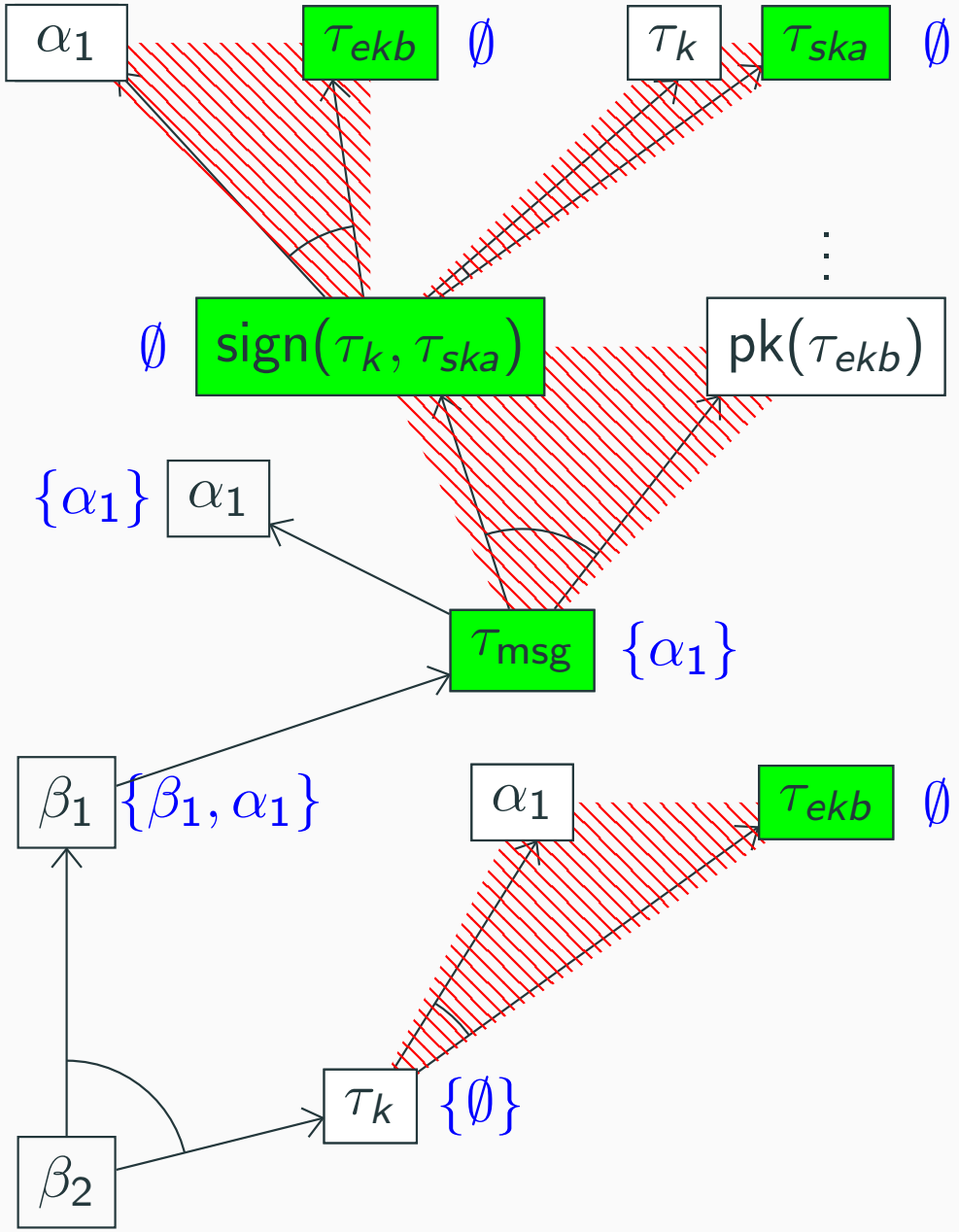
Going back to Denning Sacco: $\mathsf{dep}(\beta_2)$

$$
\begin{aligned}
&P_{\mathsf{DS}} = \\
&!\,\mathsf{new}\ k. \\
&\mathsf{out}^{\alpha_1}(c, \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(ekb))) \\
&\mid !\,\mathsf{in}^{\beta_1}(c, \mathsf{aenc}(\mathsf{sign}(x, ska), \mathsf{pk}(ekb))). \\
&\quad \mathsf{in}^{\beta_2}(c, x) \\
&\mid \mathsf{out}^{\gamma_1}(c, \mathsf{pk}(ekb)). \\
&\quad \mathsf{out}^{\gamma_2}(c, \mathsf{vk}(ska)).
\end{aligned}
$$



τ $\quad \mathsf{dep}_{\mathsf{init}}(\tau) = \emptyset$

Going back to Denning Sacco: $\mathsf{dep}(\beta_2)$



$P_{\mathsf{DS}} =$

$!\,\mathsf{new}\ k.$

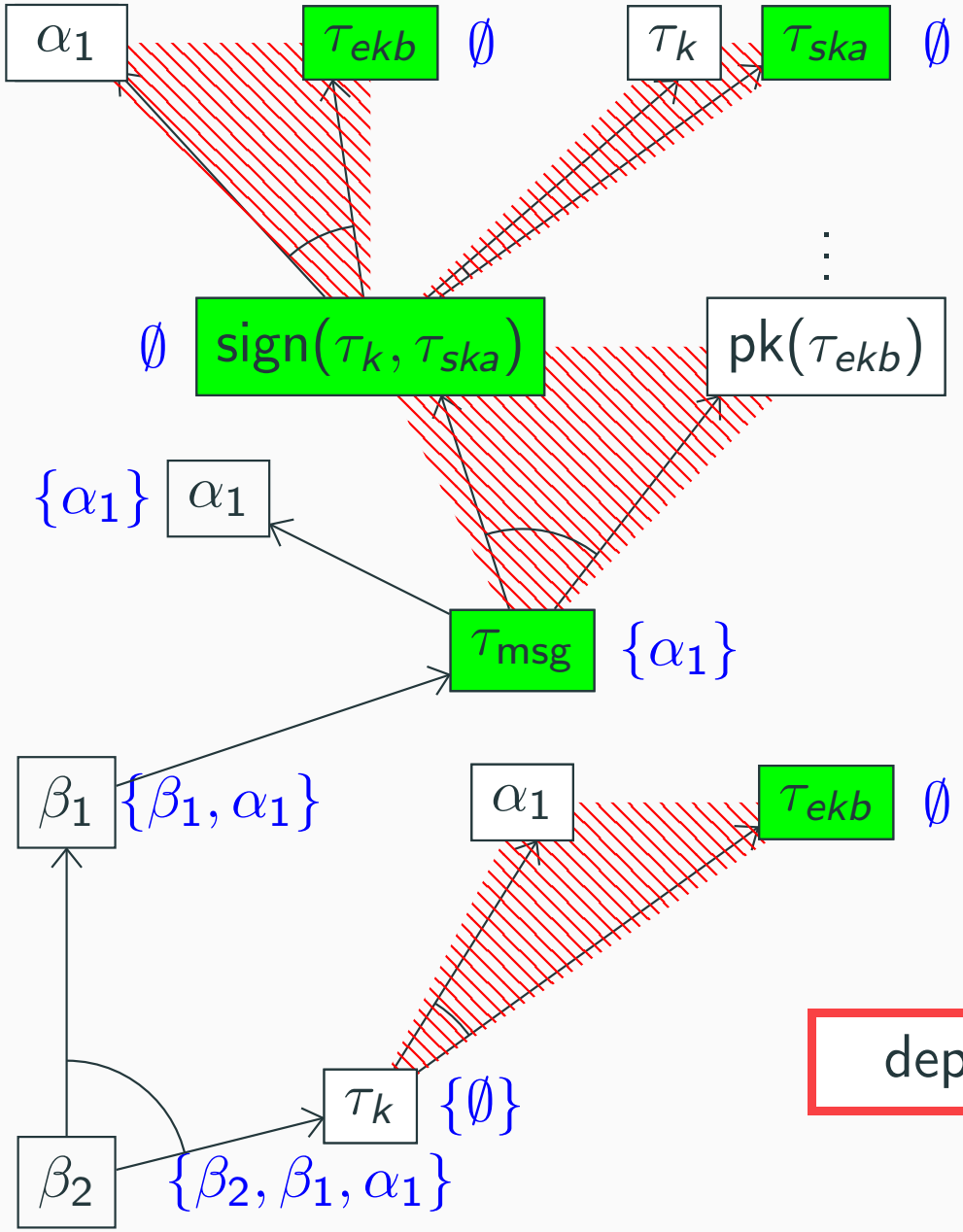
$\mathsf{out}^{\alpha_1}(c, \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(ekb)))$

$|\,!\,\mathsf{in}^{\beta_1}(c, \mathsf{aenc}(\mathsf{sign}(x, ska), \mathsf{pk}(ekb))).$
$\quad \mathsf{in}^{\beta_2}(c, x)$

$|\ \mathsf{out}^{\gamma_1}(c, \mathsf{pk}(ekb)).$
$\quad \mathsf{out}^{\gamma_2}(c, \mathsf{vk}(ska)).$

τ $\quad \mathsf{dep}_{\mathsf{init}}(\tau) = \emptyset$

Going back to Denning Sacco: $\mathsf{dep}(\beta_2)$



$$P_{\mathsf{DS}} =$$
$$!\,\mathsf{new}\ k.$$
$$\mathsf{out}^{\alpha_1}(c, \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(ekb)))$$
$$|\,!\,\mathsf{in}^{\beta_1}(c, \mathsf{aenc}(\mathsf{sign}(x, ska), \mathsf{pk}(ekb))).\ \mathsf{in}^{\beta_2}(c, x)$$
$$|\ \mathsf{out}^{\gamma_1}(c, \mathsf{pk}(ekb)).\ \mathsf{out}^{\gamma_2}(c, \mathsf{vk}(ska)).$$

τ $\quad \mathsf{dep}_{\mathsf{init}}(\tau) = \emptyset$

$$\mathsf{dep}(\beta_2) = \{\{\beta_2, \beta_1, \alpha_1\}\}$$

Case studies

	Reachability (Wsec)						
	HowMany			SAT-Equiv		DeepSec	
	nb	size		mult.	unique	mult.	unique
Symmetric protocols							
Denning-Sacco	1	3 (8)		<1s		<1s	
Needham-Schroeder	16	20 (45)	28 (63)	12s	5s	32s	18m
Otway-Rees*	4	12 (20)	16 (28)	2s	1s	< 1s	1s
Wide-Mouth-Frog*	1	3 (6)		<1s		<1s	
Kao-Chow (variant)*	48	15 (27)	28 (47)	4m	1m	3s	2m
Yahalom-Paulson*	25	19 (35)	30 (56)	2m	44s	4h	TO
Yahalom-Lowe*	-	-	-	-	-	-	-
Asymmetric protocols							
Denning-Sacco	1	2 (4)		<1s		<1s	
Needham-Schroeder*	-	-	-	-	-	-	-
NS-Lowe*	2	7 (16)	8 (18)	<1s	<1s	< 1s	<1s

Trace inclusion

A protocol P is trace included in Q, written $P \sqsubseteq_t Q$, if for every $(\mathsf{tr}, \phi) \in \mathsf{trace}(P)$, there exists $(\mathsf{tr}', \phi') \in \mathsf{trace}(Q)$ such that $\mathsf{tr} =_{\mathcal{L}} \mathsf{tr}'$, and $\phi \sqsubseteq_s \phi'$.

$\longrightarrow$ trace equivalence: $P \approx_t Q \Leftrightarrow P \sqsubseteq_t Q$ and $Q \sqsubseteq_t P$

Trace inclusion

A protocol P is trace included in Q, written $P \sqsubseteq_t Q$, if for every $(\mathrm{tr}, \phi) \in \mathrm{trace}(P)$, there exists $(\mathrm{tr}', \phi') \in \mathrm{trace}(Q)$ such that $\mathrm{tr} =_{\mathcal{L}} \mathrm{tr}'$, and $\phi \sqsubseteq_s \phi'$.

$\longrightarrow$ trace equivalence: $P \approx_t Q \Leftrightarrow P \sqsubseteq_t Q$ and $Q \sqsubseteq_t P$

1. Some additional assumptions
2. Our algorithm to compute and generate the scenarios
3. Some case studies

Additional assumptions

$\longrightarrow$ We consider protocols with a simple structure.

Simple processes: each process in parallel has its own dedicated channel.

Protocols without else branches: our processes do not use the match construct.

Main result

Theorem

Let P be a simple protocol type-compliant w.r.t. some typing system (Δ, δ). Let Q be another simple protocol such that $P \not\sqsubseteq_t Q$. There exists a trace $(\mathsf{tr}, \phi) \in \mathsf{trace}(P)$ witnessing this non-inclusion such that $\mathsf{Label}(\mathsf{tr}) \subseteq A$ for some $A \in \mathsf{dep}(P)$.

We have that:

$$\mathsf{dep}(P) = \{\emptyset\} \cup S_{\mathsf{reach}}(P) \cup S_{\mathsf{test}}(P)$$

where:

- $S_{\mathsf{reach}}(P) = \bigcup_{\alpha \in \mathsf{Label}(P)} \mathsf{dep}(\alpha)$
- $S_{\mathsf{test}}(P) = \bigcup_{\tau \in St(\delta(P))} \mathsf{dep}(\tau) \otimes S_{\mathsf{out}}(\tau)$

$\longrightarrow$ this requires a precise characterization of static inclusion

Case studies

	Equivalence (Kpriv)						
	HowMany			**SAT-Equiv**		**DeepSec**	
	nb	size		mult.	unique	mult.	unique
Symmetric protocols							
Denning-Sacco	5	5 (12)	14 (31)	<1s	<1s	<1s	<1s
Needham-Schroeder	83	33 (72)	47 (107)	6m	1m	TO	TO
Otway-Rees⋆	22	15 (23)	27 (48)	8s	7s	1s	48m
Wide-Mouth-Frog⋆	4	5 (8)	12 (20)	<1s	<1s	<1s	<1s
Kao-Chow (variant)⋆	385	29 (48)	55 (91)	10h	2h	TO	TO
Yahalom-Paulson⋆	147	29 (50)	45 (85)	45m	8m	TO	TO
Yahalom-Lowe⋆	-	-	-	-	-	-	-
Asymmetric protocols							
Denning-Sacco	5	3 (4)	8 (11)	<1s	<1s	<1s	<1s
Needham-Schroeder⋆	-	-	-	-	-	-	-
NS-Lowe⋆	20	9 (19)	14 (31)	<1s	<1s	<1s	<1s

Conclusion

Our contribution

A practical bound allowing us to rely on some existing tools to perform the security analysis both for **reachability** and **equivalence properties**.

1. We prove the correctness of our bound.
2. We implement our algorithm in HowMany.
3. We perform various case studies.

Future work

- authentication properties
- trade-off between unique and multiple scenarios