Decidable Inductive Invariants for Verification of Cryptographic Protocols with Unbounded Sessions

Emanuele D'Osualdo
Imperial College London, UK
e.dosualdo@ic.ac.uk

Felix Stutz
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany
Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
fstutz@mpi-sws.org

Abstract

We develop a theory of decidable inductive invariants for an infinite-state variant of the Applied π-calculus, with applications to automatic verification of stateful cryptographic protocols with unbounded sessions/nonces. Since the problem is undecidable in general, we introduce *depth-bounded protocols*, a strict generalisation of a class from the literature, for which our decidable analysis is sound and complete. Our core contribution is a procedure to check that an invariant is inductive, which implies that every reachable configuration satisfies it. Our invariants can capture security properties like secrecy, can be inferred automatically, and represent an independently checkable certificate of correctness. We provide a prototype implementation and we report on its performance on some textbook examples.

2012 ACM Subject Classification Theory of computation → Program verification

Keywords and phrases Security Protocols, Infinite-State Verification, Ideal Completions for WSTS

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2020.31

Related Version Full version of the paper available at https://arxiv.org/abs/1911.05430.

Supplementary Material Tool available at https://doi.org/10.5281/zenodo.3950846

Funding *Emanuele D'Osualdo*: EU Horizon 2020 Marie Curie Individual Fellowship.
Felix Stutz: supported by Imperial College London and International Max Planck Research School for Computer Science.

Acknowledgements We would like to thank Alwen Tiu, Roland Meyer and Véronique Cortier for the useful feedback.

1 Introduction

Security protocols implement secure communication over insecure channels, by using cryptography. Despite underpinning virtually every communication over the internet, new flaws that compromise security are routinely discovered in deployed protocols. Automatic protocol verification is highly desirable, but also very challenging: the space of possible attacks is infinite. Indeed, even under the assumption of perfect cryptography, security properties are undecidable [21]. The most problematic feature for decidability is the necessity of considering unboundedly many fresh random numbers, called *nonces*, to distinguish between various sessions of the protocol. There has been a proliferation of verification tools [11, 3, 35, 10, 4, 26] which can be categorised according to the way the undecidability issue is resolved. A first approach is to only consider a bounded number of sessions, possibly missing attacks. A second is to over-approximate the protocol's behaviour by representing nonces with less precision, possibly reporting spurious attacks. A third is to implement semi-algorithms, accepting that the tools might never terminate on some protocols.

31st International Conference on Concurrency Theory (CONCUR 2020).
Editors: Igor Konnov and Laura Kovács; Article No. 31; pp. 31:1–31:23
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In this paper, we devise a sound and complete analysis, i.e. one that always terminates with a correct answer, without need for approximations. We obtain this by developing decision procedures for proving invariants of a rich sub-class of protocols with unbounded sessions/nonces. An invariant is any property that holds for every reachable configuration. We introduce *depth-bounded protocols*, a strict generalisation of the class of [18], and prove that a class of invariants, called downward-closed, can be effectively represented using expressions that we call *limits*. Our core technical results are a decision procedure for limit inclusion and an algorithm called $\widehat{\mathrm{post}}$ that computes, from a limit L, a finite union of limits that represent the (infinite) set of configurations reached in one step from L. By using these two components, we obtain an algorithm to check if a limit is inductive, i.e. $\widehat{\mathrm{post}}(L) \subseteq L$. An inductive limit that contains the initial configuration is guaranteed to be an invariant for the protocol. We show how to use this to prove a number of properties including depth-boundedness itself (a semantic property), secrecy, and control-state reachability.

We define depth-bounded protocols as a subclass of a variant of the Applied π-calculus [31], with support for user-defined cryptographic primitives, secure and public channels, stateful principals, a Dolev-Yao-style intruder [17] supporting modelling of dishonest participants and leaks of old keys. In particular, our results apply to any set of cryptographic primitives that satisfy some simple axioms; examples include (a)symmetric encryption, blind signatures, hashes, XOR. To gain intuition about depth-boundedness, consider the set of messages $\Gamma_n = \{\mathsf{e}(k_1)_{k_2}, \mathsf{e}(k_2)_{k_3}, \dots \mathsf{e}(k_{n-1})_{k_n}\}$ which “chains” key k_1 to k_2, k_2 to k_3 and so on, obtaining an encryption chain of length n. A depth-bounded protocol cannot produce, or be tricked to produce, such chains of unbounded length. Note that, when computing depth, we only consider chains that are essential: the set $\Gamma_n \cup \{k_n\}$ for example has depth 1 (for any n) because it is equivalent to the set $\{k_1, \dots, k_n\}$. When there is a bound d on the depth of reachable configurations, we say that the protocol is depth-bounded. We built a proof-of-concept prototype tool to evaluate the approach, showing that many textbook protocols fall into the depth-bounded class.

More precisely, bounding depth alone is not enough to obtain decidability [18]: one needs to bound the size of messages too. For type-compliant protocols [2, 13] message size can be bounded without excluding any security violation. More generally, for typical protocols (including all our benchmarks), our inductive invariants can be computed on the size-bounded model, and then generalised to invariants for the unrestricted version of the protocol.

Our approach has a number of notable properties. First, once a suitable inductive invariant has been found, it can be provided as a certificate of correctness that can be independently checked. Second, the search of a suitable invariant can be performed both automatically (with a trade-off between precision and performance) or interactively. Third, supporting unbounded nonces makes it possible to reason about properties like susceptibility to known-plaintext attacks (Section 3.1). Finally, even coarse invariants inferred with our method can be used to prune the search space of other model checking procedures.

Related work. The pure π-calculus version of depth-boundedness was originally proposed in [27] and developed in [24, 36, 37]. Our work builds directly on [18], which introduced depth-boundedness for the special case of secrecy of protocols using symmetric encryption. We generalise to a strictly more expressive class of primitives and properties, a result that requires much more sophisticated techniques and yields more powerful algorithms.

Our theory of invariants is framed in terms of ideal completions [22], which, to the best of our knowledge, has not been instantiated to cryptographic protocols before. Our decidability proofs introduce substantial new proof techniques to deal with an active intruder while being parametric on the cryptographic primitives.

Types [16, 14, 15] can be used to capture and generalise common safe usages of cryptographic primitives, and reduce verification to constraints which can be solved efficiently. We speculate that our domain of limits and the associated algorithms could be used to define an expressive class of solvable constraints that could be integrated in type systems.

In [14, 23] two classes of protocols with unbounded nonces are shown to enjoy decidable verification. They consider a less general calculus ((a)symmetric encryption only, with atomic keys), different properties (only secrecy [23], trace equivalence [14]), and restrict protocols using (similar) syntactic conditions, obtaining classes that are orthogonal to ours.

ProVerif [4] and Tamarin [26] are two mature tools with support for a wide range of cryptographic primitives and expressive properties, and handle unbounded sessions. Both programs employ semi-algorithms and may diverge on verification tasks. ProVerif is known to terminate on so-called *tagged protocols* [6] which are incomparable to depth-bounded protocols. Tamarin offers an interactive mode when a proof cannot be carried out automatically. To the best of our knowledge, there is no characterisation for a class of protocols on which Tamarin is guaranteed to terminate.

Outline. Section 2 introduces the formal model and depth-bounded protocols. Section 3 presents our main theoretical results. In Section 4 we report on experiments with our tool and discuss limitations. All omitted proofs can be found in the full version of the paper [19].

2 Formal Model

We introduce a variant of the Applied π-calculus as our formal model of protocols. Following the Dolev-Yao intruder model, we treat cryptographic primitives algebraically. Assume an enumerable set of *names* $a, b, \dots \in \mathcal{N}$. A signature Σ of *constructors*, is a finite set of symbols f with their arity $\operatorname{ar}(\mathsf{f}) \in \mathbb{N}$. The set of messages over Σ is the smallest set $\mathbb{M}^\Sigma$ which contains all names, and is closed under application of constructors. The domain of finite sets of messages is $\mathbb{K}^\Sigma := \wp_f(\mathbb{M}^\Sigma)$. We define $\operatorname{size}(\mathsf{f}(M_1, \dots, M_n)) := 1 + \max\{\operatorname{size}(M_i) \mid 1 \leq i \leq n\}$, $\operatorname{size}(a) := 1$, and $\operatorname{names}(a) := \{a\}$, $\operatorname{names}(\mathsf{f}(M_1, \dots, M_n)) := \bigcup_{i=1}^n \operatorname{names}(M_i)$. Given $X \subseteq \mathcal{N}$ and $s \in \mathbb{N}$, we define $\mathbb{M}_s^{\Sigma,X} := \{M \in \mathbb{M}^\Sigma \mid \operatorname{names}(M) \subseteq X, \operatorname{size}(M) \leq s\}$. As is standard, Γ, Γ' and Γ, M stand for $\Gamma \cup \Gamma'$ and $\Gamma \cup \{M\}$ respectively.

A *substitution* is a finite partial function $\theta \colon \mathcal{N} \rightharpoonup \mathbb{M}^\Sigma$; we write $\theta = [M_1/x_1, \dots, M_n/x_n]$, abbreviated with $[\vec{M}/\vec{x}]$, for the substitution with $\theta(x_i) = M_i$ for all $1 \leq i \leq n$. We write $M\theta$ for the application of substitution θ to the message M, and extend the notation to sets of messages $\Gamma\theta := \{M\theta \mid M \in \Gamma\}$. A substitution θ is a *renaming of* $X \subseteq \mathcal{N}$ if it is defined on X, injective, and with $\theta(X) \subseteq \mathcal{N}$.

▶ **Definition 1** (Intruder model). *A* derivability relation *for a signature* Σ, *is a relation* $\vdash \subseteq \mathbb{K}^\Sigma \times \mathbb{M}^\Sigma$. *The pair* $\mathbb{I} = (\Sigma, \vdash)$ *is an* (effective) intruder model *if* $\vdash$ *is a (decidable) derivability relation for* Σ, *and for all* $M, N \in \mathbb{M}^\Sigma$, $\Gamma, \Gamma' \in \mathbb{K}^\Sigma$, $a \in \mathcal{N}$:

$$M \vdash M \tag{Id}$$

$$\Gamma \subseteq \Gamma' \land \Gamma \vdash M \implies \Gamma' \vdash M \tag{Mon}$$

$$\Gamma \vdash M \land \Gamma, M \vdash N \implies \Gamma \vdash N \tag{Cut}$$

$$M_1, \dots, M_n \vdash \mathsf{f}(M_1, \dots, M_n) \quad \textit{for every } \mathsf{f} \in \Sigma \textit{ with } \operatorname{ar}(\mathsf{f}) = n \tag{Constr}$$

$$\Gamma\theta \vdash M\theta \iff \Gamma \vdash M \quad \textit{for any } \theta \textit{ renaming of } \operatorname{names}(\Gamma) \tag{Alpha}$$

$$\Gamma, a \vdash M \land a \notin \operatorname{names}(\Gamma, M) \implies \Gamma \vdash M \tag{Relevancy}$$

The knowledge ordering *for* $\mathbb{I}$ *is the relation* $\leq_{\mathsf{kn}} \subseteq \mathbb{K}^\Sigma \times \mathbb{K}^\Sigma$ *such that* $\Gamma_1 \leq_{\mathsf{kn}} \Gamma_2$ *if and only if* $\forall M \in \mathbb{M}^\Sigma \colon \Gamma_1 \vdash M \implies \Gamma_2 \vdash M$. *We write* $\Gamma_1 \sim_{\mathsf{kn}} \Gamma_2$ *if* $\Gamma_1 \leq_{\mathsf{kn}} \Gamma_2$ *and* $\Gamma_2 \leq_{\mathsf{kn}} \Gamma_1$.

$$\frac{M \in \Gamma}{\Gamma \vdash M}\,\textsc{Id} \qquad \frac{\Gamma \vdash K}{\Gamma \vdash \mathsf{p}(K)}\,\textsc{Pub} \qquad \frac{\Gamma, (M,N), M, N \vdash M'}{\Gamma, (M,N) \vdash M'}\,\textsc{P}_{\textsc{l}} \qquad \frac{\Gamma \vdash M \qquad \Gamma \vdash N}{\Gamma \vdash (M,N)}\,\textsc{P}_{\textsc{r}}$$

$$\frac{\Gamma, \mathsf{e}(M)_K \vdash K \qquad \Gamma, \mathsf{e}(M)_K, M, K \vdash N}{\Gamma, \mathsf{e}(M)_K \vdash N}\,\textsc{S}_{\textsc{l}} \qquad \frac{\Gamma \vdash M \qquad \Gamma \vdash K}{\Gamma \vdash \mathsf{e}(M)_K}\,\textsc{S}_{\textsc{r}}$$

$$\frac{\Gamma, \mathsf{a}(M)_{\mathsf{p}(K)} \vdash K \qquad \Gamma, \mathsf{a}(M)_{\mathsf{p}(K)}, M, K \vdash N}{\Gamma, \mathsf{a}(M)_{\mathsf{p}(K)} \vdash N}\,\textsc{A}_{\textsc{l}} \qquad \frac{\Gamma \vdash M \qquad \Gamma \vdash N}{\Gamma \vdash \mathsf{a}(M)_N}\,\textsc{A}_{\textsc{r}}$$

Figure 1 Deduction rules for the derivability relation of $\mathbb{I}_{\mathsf{en}}$.

The first three axioms deal exclusively with what it means to be a deduction relation: what is known can be derived (Id); the more is known the more can be derived (Mon); what can be derived is known (Cut). The (Constr) axiom ensures the intruder is able to construct arbitrary messages by composing known messages. The (Alpha) axiom justifies α-renaming in our calculus. The (Relevancy) axiom allows us to only consider boundedly many nonces maliciously injected by the intruder, at each step of the protocol.

In the rest of the paper, unless otherwise specified, we fix an arbitrary effective intruder model $\mathbb{I}$ and omit the corresponding superscripts.

▶ **Proposition 2.** *Given $\Gamma_1, \Gamma_2 \in \mathbb{K}^{\Sigma}$, $\Gamma_1 \leq_{\mathsf{kn}} \Gamma_2$ if and only if $\forall M \in \Gamma_1 \colon \Gamma_2 \vdash M$. As a consequence, if $\vdash$ is decidable, so is $\leq_{\mathsf{kn}}$.*

Our framework uses the derivability relation as a black box and does not rely on the way it is specified (e.g. with a rewriting system or a deduction system). It is possible to formalise as an effective intruder model cryptographic primitives such as XOR, hashes and blind signatures. We present here, for illustration, a model of (a)symmetric encryption, and elaborate on extensions in Appendix A. We find it convenient to specify it with a sequent calculus in the style of [34]. This is an alternative to more intuitive natural-deduction-style rules, which has the key advantage of being CUT-free, while admitting CUT. This simplifies considerably the proofs of properties (e.g. Lemma 21) of the intruder model.

▶ **Example 3** (Model of Encryption). Symmetric and asymmetric encryption can be modelled using the signature $\Sigma_{\mathsf{en}} = \{(\cdot,\cdot)\,, \mathsf{e}(\cdot)_{\cdot}\,, \mathsf{a}(\cdot)_{\cdot}\,, \mathsf{p}(\cdot)\}$, where (M,N) pairs messages M and N, $\mathsf{e}(M)_N$ represents the message M encrypted with symmetric key N, $\mathsf{a}(M)_N$ represents the message M encrypted with asymmetric key N, and $\mathsf{p}(K)$ is the public key associated with the private key K. The intruder model for (a)symmetric encryption is the model $\mathbb{I}_{\mathsf{en}} = (\Sigma_{\mathsf{en}}, \vdash)$ where $\vdash$ is defined by the deduction rules in Figure 1.

▶ **Proposition 4.** *The model Σ_{en} is an effective intruder model.*

2.1 A Calculus for Cryptographic Protocols

A common approach to model cryptographic primitives is to consider both constructors (e.g. encryption) and destructors (e.g. decryption). Here messages only contain constructors, and "destruction" is represented by pattern matching. Fix a finite signature $\mathcal{Q}$ of process names (ranged over by Q) each of which has a fixed arity $\mathrm{ar}(\mathsf{Q}) \in \mathbb{N}$. A protocol specification consists of an initial process P and a finite set Δ of (possibly recursive) definitions of the form $\mathsf{Q}[x_1, \ldots, x_n] := A$, with $\mathrm{ar}(\mathsf{Q}) = n$, where the syntax of P and A follows the grammar:

$$\begin{aligned} P &::= \mathbf{0} \;\mid\; \nu x.P \;\mid\; P \parallel P \;\mid\; \langle M \rangle \;\mid\; \mathsf{Q}[\vec{M}] && \text{(process)} \\ A &::= \overline{a}\langle M \rangle \;\mid\; a(\vec{x} : M).P \;\mid\; A + A && \text{(action)} \end{aligned}$$

$$\frac{\mathsf{Q}_1[\vec{M}_1] \triangleq \overline{c}\langle N[\vec{M}'/\vec{x}]\rangle.P_1 + A_1 \qquad \mathsf{Q}_2[\vec{M}_2] \triangleq c(\vec{x} : N).P_2 + A_2}{\nu\vec{a}.(\langle\Gamma\rangle \parallel \mathsf{Q}_1[\vec{M}_1] \parallel \mathsf{Q}_2[\vec{M}_2] \parallel C) \to_\Delta \nu\vec{a}.(\langle\Gamma\rangle \parallel P_1 \parallel P_2[\vec{M}'/\vec{x}] \parallel C)}\ \textsc{Comm}$$

$$\frac{P \equiv P' \to_\Delta Q' \equiv Q}{P \to_\Delta Q}\ \textsc{Struct} \qquad \frac{\mathsf{Q}[\vec{M}] \triangleq \overline{c}\langle M\rangle.P + A \qquad \Gamma \vdash c}{\nu\vec{a}.(\langle\Gamma\rangle \parallel \mathsf{Q}[\vec{M}] \parallel C) \to_\Delta \nu\vec{a}.(\langle\Gamma\rangle \parallel \langle M\rangle \parallel P \parallel C)}\ \textsc{PubOut}$$

$$\frac{\mathsf{Q}[\vec{M}] \triangleq c(\vec{x} : N).P + A \qquad \Gamma, \vec{y} \vdash N[\vec{M}'/\vec{x}] \qquad \Gamma \vdash c \qquad \vec{y} \text{ fresh}}{\nu\vec{a}.(\langle\Gamma\rangle \parallel \mathsf{Q}[\vec{M}] \parallel C) \to_\Delta \nu\vec{a}.\nu\vec{y}.(\langle\Gamma\rangle \parallel \langle\vec{y}\rangle \parallel P[\vec{M}'/\vec{x}] \parallel C)}\ \textsc{PubIn}$$

Figure 2 Operational semantics.

We use the vector notation $\vec{x} = x_1, \dots, x_n$ for lists of pairwise distinct names. In an action $a(\vec{x} : M).P$, we call $\vec{x} : M$ the *pattern*, and P the *continuation*; processes $\mathsf{Q}[\vec{M}]$ are called *process calls*. If $\Gamma = \{M_1, \dots, M_k\}$ is a finite set of messages, then $\langle\Gamma\rangle := \langle M_1\rangle \parallel \dots \parallel \langle M_k\rangle$. We define $P^0 := \mathbf{0}$ and $P^{n+1} := P \parallel P^n$. For brevity, we assume the special name **in** is known to the intruder. The internal action $\boldsymbol{\tau}$, is an abbreviation for $\mathbf{in}(x : x)$, for a fresh x. Processes of the form $\langle M\rangle$ or $\mathsf{Q}[\vec{a}]$ are called *sequential*. The names $\vec{x}$ are bound in both $\nu\vec{x}.P$ and $c(\vec{x} : M).P$. We denote the set of free names of a term P with $\mathrm{fn}(P)$ and the set of bound names with $\mathrm{bn}(P)$. As is standard, we require, wlog, that $\mathrm{fn}(P) \cap \mathrm{bn}(P) = \emptyset$. When nesting restrictions $\nu\vec{x}.\nu\vec{y}.P$, we implicitly assume wlog that $\vec{x}$ and $\vec{y}$ are disjoint. We assume there is at most one definition for each $\mathsf{Q} \in \mathcal{Q}$, and that for each definition $\mathsf{Q}[x_1, \dots, x_n] := A$, $\mathrm{fn}(A) \subseteq \{x_1, \dots, x_n\}$. The set $\mathbb{P}$ consists of all processes over an underlying signature $\mathcal{Q}$.

Structural congruence. We write $\overset{\alpha}{=}$ for standard α-equivalence. Structural congruence, $\equiv$, is the smallest congruence relation that includes $\overset{\alpha}{=}$, and is associative and commutative with respect to $\parallel$ and $+$ with $\mathbf{0}$ as the neutral element, and satisfies the standard laws: $\nu a.\mathbf{0} \equiv \mathbf{0}$, $\nu a.\nu b.P \equiv \nu b.\nu a.P$, and $P \parallel \nu a.Q \equiv \nu a.(P \parallel Q)$ if $a \notin \mathrm{fn}(P)$. Every process P is congruent to a process in *standard form*:

$$\nu\vec{x}.\big(\langle M_1\rangle \parallel \cdots \parallel \langle M_m\rangle \parallel \mathsf{Q}_1[\vec{N}_1] \parallel \cdots \parallel \mathsf{Q}_k[\vec{N}_k]\big) \tag{SF}$$

where every name in $\vec{x}$ occurs free in some subterm. We write $\mathrm{sf}(P)$ for the standard form of P, which is unique up to α-equivalence, and associativity and commutativity of parallel. We abbreviate standard forms with $\nu\vec{x}.(\langle\Gamma\rangle \parallel Q)$ where all the active messages are collected in Γ, and Q is a parallel composition of process calls. Let $\mathrm{sf}(P)$ be the expression (SF), we define $\mathrm{msg}(P) = \{M_1, \dots, M_m\} \cup \bigcup_{i=1}^{k} \vec{N}_i$. Thus $\mathrm{msg}(P)$ is the set of messages appearing in a term. When $m = 0, k = 0, \vec{x} = \emptyset$, the expression (SF) is $\mathbf{0}$.

Reduction semantics. One can think of standard forms $\nu\vec{x}.(\langle\Gamma\rangle \parallel Q)$ as runtime configurations of the protocol. They capture, at a specific point in time, the current relevant names (which encode nonces/keys/data), the knowledge of the intruder Γ, and the local state of each participant. A sequential term $\mathsf{Q}[\vec{N}]$ represents a single participant in control state Q with local knowledge of messages $\vec{N}$.

Principals can communicate through channels; a channel known by the intruder is considered insecure. An input action over an insecure channel can be fired if the intruder can produce any message that matches the action's pattern. An output $\overline{c}\langle M\rangle$ to an insecure channel c leaks message M to the intruder, who can decide to forward it angelically to a corresponding input over c (modelling an honest step) or hijack the communication.

$$\begin{aligned}
\mathsf{S}[a,b,k_{as},k_{bs}] &:= \mathbf{in}(n_a : (n_a,b)).\nu k.\big(\langle \mathsf{e}(k)_{k_{bs}}\rangle \parallel \big\langle \mathsf{e}(k)_{(n_a,k_{as})}\big\rangle \parallel \mathsf{S}[a,b,k_{as},k_{bs}]\big)\\
\mathsf{A}_1[a,b,k_{as}] &:= \boldsymbol{\tau}.\nu n_a.(\langle (n_a,b)\rangle \parallel \mathsf{A}_2[a,b,k_{as},n_a] \parallel \mathsf{A}_1[a,b,k_{as}])\\
\mathsf{A}_2[a,b,k_{as},n_a] &:= \mathbf{in}\big(k : \mathsf{e}(k)_{(n_a,k_{as})}\big).\mathsf{A}_3[a,b,k_{as},k]\\
\mathsf{A}_3[a,b,k_{as},k] &:= \mathbf{in}(n_b : \mathsf{e}(n_b)_k).\langle \mathsf{e}(n_b)_{(k,k)}\rangle\\
\mathsf{B}_1[a,b,k_{bs}] &:= \mathbf{in}(k : \mathsf{e}(k)_{k_{bs}}).\nu n_b.\big(\langle \mathsf{e}(n_b)_k\rangle \parallel \mathsf{B}_2[a,b,k_{bs},n_b,k] \parallel \mathsf{B}_1[a,b,k_{bs}]\big)\\
\mathsf{B}_2[a,b,k_{bs},n_b,k] &:= \mathbf{in}(\mathsf{e}(n_b)_{(k,k)}).\mathsf{Secret}[k]
\end{aligned}$$

Figure 3 Formal model of Example 9.

We write $\mathsf{Q}[\vec{M}] \triangleq A$ if $\mathsf{Q}[\vec{x}] := A' \in \Delta$ and $A \overset{\alpha}{=} A'[\vec{M}/\vec{x}]$, up to commutativity and associativity of $+$. The transition relation $\rightarrow_\Delta$ is defined in Figure 2. In Rule PUBIN, $\vec{y}$ denotes all fresh names introduced by the intruder in this step. Thanks to (Relevancy), one can wlog ignore transitions where $\mathrm{fn}(\vec{y}) \not\subseteq \mathrm{fn}(\vec{M}')$, since unused names would simply not contribute to the intruder knowledge. The sets $\mathrm{reach}_\Delta(P) := \{Q \mid P \rightarrow^*_\Delta Q\}$ and $\mathrm{traces}_\Delta(P) := \{Q_0 \cdots Q_n \mid P \equiv_{\mathsf{kn}} Q_0 \rightarrow_\Delta \cdots \rightarrow_\Delta Q_n\}$ collect the processes reachable from P and all the transition sequences from P respectively, given the definitions Δ. We omit Δ when unambiguous.

▶ **Definition 5** ($\equiv_{\mathsf{kn}}$)**.** Knowledge congruence*, $P \equiv_{\mathsf{kn}} Q$, is the smallest congruence that includes $\equiv$ and such that $\langle \Gamma_1\rangle \equiv_{\mathsf{kn}} \langle \Gamma_2\rangle$ if $\Gamma_1 \sim_{\mathsf{kn}} \Gamma_2$.*

Knowledge congruence is also characterised by

$$P_1 \equiv_{\mathsf{kn}} P_2 \iff \mathrm{sf}(P_1) \overset{\alpha}{=} \nu\vec{x}.(\langle \Gamma_1\rangle \parallel Q) \land \mathrm{sf}(P_2) \overset{\alpha}{=} \nu\vec{x}.(\langle \Gamma_2\rangle \parallel Q) \land \Gamma_1 \sim_{\mathsf{kn}} \Gamma_2.$$

Intuitively, modulo derivability, two processes $P \equiv_{\mathsf{kn}} Q$ are indistinguishable to the intruder and to the principals. Formally, if $P \equiv_{\mathsf{kn}} Q$ then the transitions systems $(P, \rightarrow_\Delta)$ and $(Q, \rightarrow_\Delta)$ are isomorphic. We thus close the reduction semantics under knowledge congruence: we add the rule that if $P \equiv_{\mathsf{kn}} P' \rightarrow_\Delta Q' \equiv_{\mathsf{kn}} Q$ then $P \rightarrow_\Delta Q$.

While knowledge congruence captures when two configurations are essentially the same, knowledge embedding formalises the notion of "sub-configuration".

▶ **Definition 6** (Knowledge embedding)**.** *The* knowledge embedding *relation $P_1 \sqsubseteq_{\mathsf{kn}} P_2$ holds if $P_1 \equiv \nu\vec{x}.(\langle \Gamma_1\rangle \parallel Q)$, $P_2 \equiv \nu\vec{x}.\nu\vec{y}.(\langle \Gamma_2\rangle \parallel Q \parallel Q')$ and $\Gamma_1 \leq_{\mathsf{kn}} \Gamma_2$.*

▶ **Proposition 7.** *$P_1 \equiv_{\mathsf{kn}} P_2$ if and only if $P_1 \sqsubseteq_{\mathsf{kn}} P_2$ and $P_2 \sqsubseteq_{\mathsf{kn}} P_1$.*

▶ **Theorem 8.** *Knowledge embedding is a simulation, that is, for all P, P' and Q, if $P \rightarrow Q$ and $P \sqsubseteq_{\mathsf{kn}} P'$ then there is a Q' such that $P' \rightarrow Q'$ and $Q \sqsubseteq_{\mathsf{kn}} Q'$.*

▶ **Example 9.** Consider the following toy protocol, given in Alice&Bob notation, meant to establish a new session key K between A and B through a trusted server S:

$$\begin{array}{llllll}
(1) & A \rightarrow S: & N_A, B & (3) & B \rightarrow A: & \mathsf{e}(K)_{(N_A,K_{AS})}, \mathsf{e}(N_B)_K\\
(2) & S \rightarrow B: & \mathsf{e}(K)_{(N_A,K_{AS})}, \mathsf{e}(K)_{K_{BS}} & (4) & A \rightarrow B: & \mathsf{e}(N_B)_{(K,K)}
\end{array}$$

Figure 3 shows the protocol formalised in our calculus. Assume the initial state is

$$P_0 = \nu a, b, k_{as}, k_{bs}.(\mathsf{S}[a,b,k_{as},k_{bs}] \parallel \mathsf{A}_1[a,b,k_{as}] \parallel \mathsf{B}_1[a,b,k_{bs}] \parallel \langle a,b\rangle).$$

Step (1) is initiated by A_1 which sends some new name n_a to the server; since communication is over an insecure channel, the message is just output without indicating the intended recipient. The server receives the message (or any message the intruder may decide to forge instead) and outputs the fresh key k encrypted with k_{bs} (the long-term key between B and S) and with the pair (n_a, k_{as}) (note the use of non-atomic encryption keys). In the protocol, these two messages are sent to B but we model step (2) by B_1 which just receives the message relevant to B. The forwarding of $\mathsf{e}(k)_{(n_a,k_{as})}$ from S to A is performed by the intruder instead of B in the model.

In the last two steps, modelled by B_2 and A_3, B sends a nonce n_b encrypted with k, to challenge A to prove she knows k, which she does by sending back $\mathsf{e}(n_b)_{(k,k)}$. At this point, B is convinced that by encrypting messages with k they will be only accessible to A. We model this by making B_2 transition to $\mathsf{Secret}[k]$ after a successful challenge. We always assume the definition $\mathsf{Secret}[k] := \mathbf{in}(k).\mathsf{Leak}[k]$. A transition to $\mathsf{Leak}[k]$ is only possible when the intruder can derive k so we can check whether the secrecy assertion holds by checking that no reachable process contains a call to $\mathsf{Leak}[k]$.

Notice how A_1 and B_1 spawn both the continuation of the session and (recursively) a process ready to start a new session. This creates the possibility of an unbounded number of sessions, each of which will involve fresh n_a, n_b, and k.

Threat model. Our reduction semantics follows the Dolev-Yao attacker model in representing the intruder's interference: the intruder mediates every communication over insecure channels, is able to create new names and analyse and construct messages from all the messages that have been communicated insecurely so far. Threat models that go beyond Dolev-Yao include dishonest participants and compromised old session keys. These aspects are not embedded in the semantics, but can be modelled through the process definitions. If we wanted to model compromised keys in Example 9, for instance, we could modify the definition of B_2 to $\mathsf{B}_2[a, b, k_{bs}, n_b, k] := \mathbf{in}(\mathsf{e}(n_b)_{(k,k)}).\mathsf{Secret}[k] + \mathbf{in}(\mathsf{e}(n_b)_{(k,k)}).\langle k\rangle$ which makes a non-deterministic choice to declare k a secret, or to consider it as old and reveal it.

▶ Remark 10 (Implementable patterns). Our calculus represents message deconstruction (e.g. decryption) with pattern matching. However, general pattern matching is too powerful: a pattern like $\mathbf{in}(x, k : \mathsf{e}(x)_k)$ would obtain *both* the key k and the plaintext x from an encrypted message! This is only a modelling problem: one should make sure all patterns can be implemented using the cryptographic primitives. Consider a pattern $\vec{x} : M$ and let $Z = \mathrm{names}(M) \setminus \vec{x}$; the pattern is *implementable*, if, for all $\theta\colon Z \rightharpoonup \mathbb{M}$, we have $M\theta, Z\theta \vdash y$ for all $y \in \vec{x}$.

2.2 Depth-Bounded Protocols

We can now define the class of depth-bounded protocols, a strict generalisation of the notion in [18]. While the definitions of [18] depend on fixing the intruder to symmetric encryption only, here we define it fully parametrically to the intruder model.

▶ **Definition 11** (Depth). *The* nesting of restrictions *of a term is given by the function* $\mathrm{nest}_\mathsf{v}(\mathsf{Q}[\vec{a}]) := \mathrm{nest}_\mathsf{v}(\langle M\rangle) := \mathrm{nest}_\mathsf{v}(\mathbf{0}) := 0$, $\mathrm{nest}_\mathsf{v}(\nu x.P) := 1 + \mathrm{nest}_\mathsf{v}(P)$, $\mathrm{nest}_\mathsf{v}(P \parallel Q) := \max(\mathrm{nest}_\mathsf{v}(P), \mathrm{nest}_\mathsf{v}(Q))$. *The* depth *of a term is defined as the minimal nesting of restrictions in its knowledge congruence class,* $\mathrm{depth}(P) := \min\{\mathrm{nest}_\mathsf{v}(Q) \mid Q \equiv_{\mathsf{kn}} P\}$.

▶ **Lemma 12.** *Every Q is α-equivalent to a process Q' such that* $|\mathrm{bn}(Q')| \leq \mathrm{nest}_\mathsf{v}(Q)$.

Consider for example $P = \nu a, b, c.(\langle a\rangle \parallel \langle \mathsf{e}(b)_a\rangle \parallel \langle \mathsf{e}(c)_b\rangle \parallel \langle c\rangle)$ which has $\mathrm{nest}_\mathsf{v}(P) = 3$. The process P is knowledge-congruent to $Q = (\nu a.\langle a\rangle \parallel \nu b.\langle b\rangle \parallel \nu c.\langle c\rangle)$ which has $\mathrm{nest}_\mathsf{v}(Q) = 1$; this gives us $\mathrm{depth}(P) = \mathrm{nest}_\mathsf{v}(Q) = 1$. Although $\mathrm{bn}(Q) = \{a, b, c\}$, by α-renaming all names to x we obtain $Q' = (\nu x.\langle x\rangle \parallel \nu x.\langle x\rangle \parallel \nu x.\langle x\rangle)$ which has the property $|\mathrm{bn}(Q')| \leq \mathrm{nest}_\nu(Q) \leq \mathrm{depth}(P)$. More generally, Lemma 12 says that processes of depth k can always be represented using at most k unique names, by reusing names in disjoint scopes.

Let $\mathbb{S}_s := \{P \in \mathbb{P} \mid \forall M \in \mathrm{msg}(P) \colon \mathrm{size}(M) \leq s\}$ be the set of processes containing messages of size at most s. The set $\mathbb{D}^X_{s,k}$ is the set of processes of depth at most $k \in \mathbb{N}$, with free names in X, and messages not exceeding size s:

$$\mathbb{D}^X_{s,k} := \{P \in \mathbb{S}_s \mid \mathrm{fn}(P) \subseteq X, \exists Q \in \mathbb{S}_s \colon Q \equiv_{\mathsf{kn}} P \wedge \mathrm{nest}_\nu(Q) \leq k\}.$$

When starting from some initial process P_0, every reachable process P has $\mathrm{fn}(P) \subseteq \mathrm{fn}(P_0)$ so X can always be fixed to be $\mathrm{fn}(P_0)$. We therefore omit X from the superscripts to unclutter notation. The set of processes reachable from P while respecting a size bound s is the set $\mathrm{reach}^s_\Delta(P) := \{Q \mid P \cdots Q \in \mathrm{traces}_\Delta(P) \cap \mathbb{S}^*_s\}$.

▶ **Definition 13.** *For some $s, k \in \mathbb{N}$, we say the process P is* (s,k)-bounded *(w.r.t. a finite set Δ of definitions) if $\mathrm{reach}^s_\Delta(P) \subseteq \mathbb{D}_{s,k}$, i.e. from P only processes of depth at most k can be reached, in traces respecting the size bound s.*

▶ **Example 14.** Example 9 is $(3,7)$-bounded. We defer the proof of this fact to Section 3.6.

▶ **Example 15** (Encryption Oracle)**.** The definition $\mathsf{E}[k] := \mathbf{in}(x : x).(\langle \mathsf{e}(x)_k\rangle \parallel E[k])$ leads to unboundedness as soon as the initial process contains $\mathsf{E}[k]$ for some k not known to the intruder, and size bound such that x can match messages of size greater than 1. In such case, the intruder can inject messages (c_i, c_{i+1}) for unboundedly many i, where c_i are intruder-generated nonces. Since k is secret, the resulting reachable configurations would contain "encryption chains" of the form $\nu k.\nu c_1, \dots, c_n.(\langle \mathsf{e}(c_1, c_2)_k\rangle \parallel \langle \mathsf{e}(c_2, c_3)_k\rangle \parallel \dots \langle \mathsf{e}(c_{n-1}, c_n)_k\rangle)$. When such chains appear in a set for unboundedly many $n \in \mathbb{N}$, the set is not depth-bounded. This *encryption oracle* pattern could be considered an anti-pattern because it can be exploited for a chosen-plaintext attack on the key k. The pattern can be usually modified or constrained to obtain a bounded protocol. One option is to limit the verification to only consider traces where x is of size 1.[1]

The two bounds s and k are very different in nature. For size, we ignore any trace that involves messages exceeding size s. Then we determine if the depth bound k is respected by all remaining traces. Ignoring traces exceeding s is acceptable for protocols not susceptible to type confusion attacks and is achieved in other tools by using typing. Our method can however be pushed beyond this limitation: In Section 4.1, we show how the results of our analysis on the traces of bounded message size, can be generalised to results that hold for the unrestricted set of traces.

3 Ideal Completions for Security Protocols

Our main technical contributions are the proofs needed to show that (s,k)-bounded protocols form a post-effective ideal completion in the sense of [8]. First we outline the significance and applications of this result, and then proceed with the proofs.

[1] In Tamarin one would obtain this by typing x:*fresh*.

3.1 Downward-Closed Invariants and Security Properties

Suppose we want to establish that a protocol P fulfils some security requirement. In a typical proof, one needs to establish many intermediate facts about executions of the protocol. For example, part of the argument may hinge on some key k being always unknown to the intruder. This kind of property is an *invariant* of the protocol: it holds at every step of an execution. Formally, an invariant of P (under definitions Δ and size constraint s) is any set of processes that includes $\mathrm{reach}^s_\Delta(P)$. For example, k is never leaked to the intruder in executions of the protocol P if the set of processes $\mathcal{S}_k := \{Q \mid \langle k\rangle \not\sqsubseteq_{\mathsf{kn}} Q\}$ – i.e. all the processes where k is not public – is an invariant of P. We will focus here on the class of $\sqsubseteq_{\mathsf{kn}}$-downward-closed invariants. Formally, given a set of processes X, its $\sqsubseteq_{\mathsf{kn}}$-*downward closure* is the set $X{\downarrow} := \{Q \mid \exists P \in X : Q \sqsubseteq_{\mathsf{kn}} P\}$. A set X is $\sqsubseteq_{\mathsf{kn}}$-*downward closed* if $X = X{\downarrow}$. Many properties of interest are naturally downward closed. For example, the set $\mathcal{S}_k$ above is downward-closed as $\langle k\rangle \not\sqsubseteq_{\mathsf{kn}} Q$ and $Q' \sqsubseteq_{\mathsf{kn}} Q$ implies $\langle k\rangle \not\sqsubseteq_{\mathsf{kn}} Q'$.

The problem we need to solve is, then, how to show that a given downward-closed set X is an invariant for a given protocol. Formally, that corresponds to checking $X \supseteq \mathrm{reach}^s_\Delta(P)$ which, by downward-closure of X, is equivalent to checking $X \supseteq \mathrm{reach}^s_\Delta(P){\downarrow}$. To prove the latter inclusion, our strategy is to find an inductive invariant that includes the initial state P and that is included in X. Let $\mathrm{post}^s_\Delta(X) := \{Q' \mid \exists Q \in X, Q \to Q' \in \mathbb{S}_s\}$ be the set of processes reachable in one step from processes in X. An invariant X is *inductive* if $X \supseteq \mathrm{post}^s_\Delta(X)$, which is equivalent to requiring $X \supseteq \mathrm{post}^s_\Delta(X){\downarrow}$ if X is downward-closed. Any inductive invariant that contains the initial process P will include $\mathrm{reach}^s_\Delta(P)$.
To turn this proof strategy into an algorithm, we need three components:

1. a recursively enumerable finite representation of downward-closed sets,
2. a way to decide inclusion between two downward-closed sets, given their representation,
3. an algorithm (called $\widehat{\mathrm{post}}^s_\Delta$) to compute, given a finite representation of a downward-closed set D, a finite representation of $\mathrm{post}^s_\Delta(D){\downarrow}$.

Unfortunately, downward-closed sets cannot be finitely represented in general, especially if one considers unbounded sessions/nonces. We will show, however, that we can devise solutions to all three items above for downward-closed subsets of $\mathbb{D}_{s,k}$, under a mild restriction on the intruder model. Solving problems 1 to 3 amounts to proving that $\mathbb{D}_{s,k}$ admits a *post-effective ideal completion* in the sense of [22, 8]. This implies that we can decide if the reachable configurations satisfy any given downward-closed property, by adapting the enumeration scheme presented in [8].

▶ **Theorem 16.** *Given a property $\mathcal{D} \subseteq \mathbb{P}$, and a protocol P with definitions Δ, we write $P, \Delta \models^s \mathcal{D}$ if $\mathrm{reach}^s_\Delta(P) \subseteq \mathcal{D}$. If $\mathcal{D} \subseteq \mathbb{D}_{s,k}$ is downward-closed, then $P, \Delta \models^s \mathcal{D}$ is decidable.*

Proof. The algorithm runs two semi-procedures, Prover and Refuter, in parallel. The first procedure, Prover, enumerates all the downward-closed subsets I of $\mathbb{D}_{s,k}$. For each I, Prover checks if

(a) $P \in I$,
(b) I is inductive, by checking $\widehat{\mathrm{post}}^s_\Delta(I) \subseteq I$, and
(c) $I \subseteq \mathcal{D}$.

If we find such a set I, then we have proven $\mathrm{reach}_\Delta(P) \subseteq \mathcal{D}$, and the overall algorithm can terminate returning "True". The second procedure, Refuter, enumerates all $Q \in \mathrm{reach}_\Delta(P)$ and checks if $Q \notin \mathcal{D}$, in which case the overall algorithm can terminate returning "False".

When $P, \Delta \models^s \mathcal{D}$ holds, then, in the worst case, Prover will eventually consider the finite representation of $\mathrm{reach}^s_\Delta(P){\downarrow}$, which satisfies checks (a) to (c) above. In the case where $P, \Delta \models^s \mathcal{D}$ does not hold, Refuter would eventually find a reachable process not in $\mathcal{D}$. In either case, the algorithm terminates with the correct answer. ◀

The above algorithm can be used to decide the following properties.

Deciding (s,k)-boundedness. The set $\mathbb{D}_{s,k}$ is itself downward-closed, so we can decide if P is (s,k)-bounded, by deciding $P, \Delta \models^{s} \mathbb{D}_{s,k}$.

Deciding control-state reachability and secrecy. Control-state reachability asks whether there is an execution of the protocol which reaches a process containing a process call $\mathsf{Q}[\dots]$ for some given Q. Secrecy can be reduced to control-state reachability by introducing a definition $\mathsf{Secret}[m] := \mathbf{in}(m).\mathsf{Leak}[m]$ (for a special process identifier Leak with no definition). In the definition of the protocol one can call $\mathsf{Secret}[m]$ to mark some message m as a secret, and secrecy corresponds to asking control-state (un)reachability for Leak.

If P is (s,k)-bounded, control-state reachability for Q from P, can decided by $P, \Delta \models^{s} \mathcal{D}_{\mathsf{Q}}$, where $\mathcal{D}_{\mathsf{Q}}$ is the (downward-closed) subset of $\mathbb{D}_{s,k}$ of processes that do not contain calls to Q. Notice that, when P is arbitrary, the algorithm checks (s,k)-boundedness and control-state reachability at the same time.

Absence of misauthentication. A misauthentication happens when a principal a believes she shares a secret n with b but b believes she shares the secret n with some other entity c. To check this situation can never arise, we can produce the process $\mathsf{Auth}[a,b,n]$ when a believes to share the secret n with b. Absence of misauthentication can be decided by $P, \Delta \models^{s} \mathcal{A}$ where $\mathcal{A} = \{\, Q \in \mathbb{D}_{s,k} \mid \nu a,b,c,n.(\mathsf{Auth}[a,b,n] \parallel \mathsf{Auth}[b,c,n]) \not\sqsubseteq_{\mathsf{kn}} Q \,\}$.

Susceptibility to known-plaintext attacks. The task of guessing a symmetric key is made much easier if it is possible for the attacker to have access to an arbitrarily large number of known nonces encrypted with the same key k. We can model this situation by asking if:

$$\forall n \in \mathbb{N} \colon \exists Q \in \mathrm{reach}^{s}_{\Delta}(P) \colon R^{n} \sqsubseteq_{\mathsf{kn}} Q \qquad \text{where } R = \nu m.(\langle m \rangle \parallel \langle \mathsf{e}(m)_{k} \rangle) \tag{†}$$

If (†) holds for P then the intruder does have access to an unbounded supply of known messages m encrypted with the same key k. Interestingly, the property becomes meaningful only when considering unbounded number of nonces. Condition (†) is equivalent to $\mathrm{reach}^{s}_{\Delta}(P){\downarrow} \supseteq \{R^{n} \mid n \in \mathbb{N}\}{\downarrow}$. If we find a downward-closed inductive invariant I for P, such that $I \not\supseteq \{R^{n} \mid n \in \mathbb{N}\}{\downarrow}$, then we can be sure that (†) does not hold, and P is not susceptible to known-plaintext attacks on k. We can therefore semi-decide (†) by enumerating all candidate I. Contrary to the previous algorithms, we are not able to provide a Refuter procedure (we conjecture the problem is undecidable). We can get a decision procedure if, instead of unboundedly many plaintext-encrypted pairs we ask whether a sufficiently high, user-provided number N of such pairs can be produced. The problem can be extended to cover the case where the known-plaintext can be any message of size at most s.

Notice how, if the protocol is found to satisfy the property, the algorithms above can output an inductive invariant acting as an independently checkable certificate of correctness. Although the mentioned security properties alone do not cover the full security requirements, an effectively presented invariant can provide the foundations to prove further properties.

The algorithm of Theorem 16 relies on expensive enumeration schemes, which are mainly a theoretical device to prove decidability. In a more practical setting, the candidate invariants can be supplied by the user and refined interactively, avoiding the need for the enumeration of Prover, or they can be inferred as we describe in Section 3.6.

3.2 Bounded Processes are Well-Quasi-Ordered

We construct finite representations of downward-closed invariants by making use of the algebraic structure of the quasi-order $(\mathbb{D}_{s,k}, \sqsubseteq_{\mathsf{kn}})$. A relation $\sqsubseteq \subseteq S \times S$ over some set S is a *quasi-order* (qo) if it is reflexive and transitive. An infinite sequence $s_0, s_1, \dots$ of elements of S is called *good* if there are two indexes $i < j$ such that $s_i \sqsubseteq s_j$. A qo $(S, \sqsubseteq)$ is called a *well quasi order* (wqo) if all its sequences are good.

We prove $(\mathbb{D}_{s,k}, \sqsubseteq_{\mathsf{kn}})$ is a wqo for any intruder model, by showing a correspondence between processes in $\mathbb{D}_{s,k}$ and finitely-labelled forests of height at most k, which we represent as nested multisets. The details can be found in the full version of the paper [19].

3.3 Limits and Ideal Decompositions

By exploiting the wqo structure of $\mathbb{D}_{s,k}$, we can provide a finite representation for its downward-closed sets. Let $(S, \sqsubseteq)$ be a qo. A set $D \subseteq S$ is an *ideal* if it is downward-closed and directed, i.e. for all $x, y \in D$ there is a $z \in D$ such that $x \sqsubseteq z$ and $y \sqsubseteq z$. We write $\mathrm{Idl}(S)$ for the set of ideals of S. It is well-known that in a well-quasi-order, every downward-closed set is equal to a canonical minimal finite union of ideals, its *ideal decomposition*. To represent downward-closed sets of $\mathbb{D}_{s,k}$ we will only need to provide finite representations of its ideals. We represent ideals using *limits*, which have the same syntax as processes augmented with a construct -^{ω} to represent an arbitrary number of parallel components.

▶ **Definition 17** (Limits). *We call* limits *the terms L formed according to the grammar:*

$$\mathbb{L} \ni L ::= \mathbf{0} \mid (R_1 \parallel \cdots \parallel R_n) \qquad R ::= B \mid B^{\omega} \qquad B ::= \langle M \rangle \mid \mathsf{Q}[\vec{M}] \mid \mathsf{v}x.L$$

▶ **Definition 18** (Denotation of limits). *The denotation of L is the set $[\![L]\!] := [\,L\,]{\downarrow}$ where:*

$$\begin{aligned} [\,\mathbf{0}\,] &:= \{\mathbf{0}\} & [\,L_1 \parallel L_2\,] &:= \{(P_1 \parallel P_2) \mid P_1 \in [\,L_1\,], P_2 \in [\,L_2\,]\} \\ [\,\mathsf{Q}[\vec{M}]\,] &:= \{\mathsf{Q}[\vec{M}]\} & [\,B^{\omega}\,] &:= \textstyle\bigcup_{n \in \mathbb{N}} \{(P_1 \parallel \cdots \parallel P_n) \mid \forall i \leq n : P_i \in [\,B\,]\} \\ [\,\langle M \rangle\,] &:= \{\langle M \rangle\} & [\,\mathsf{v}x.L\,] &:= \{\mathsf{v}x.P \mid P \in [\,L\,]\} \end{aligned}$$

We call the processes in $[\![L]\!]$ instances *of L. Define $\mathrm{nest}_{\mathsf{v}}(L)$ to be as $\mathrm{nest}_{\mathsf{v}}$ on processes with the addition of the case $\mathrm{nest}_{\mathsf{v}}(L^{\omega}) := \mathrm{nest}_{\mathsf{v}}(L)$. It is easy to check that for each $P \in [\![L]\!]$, $\mathrm{depth}(P) \leq \mathrm{nest}_{\mathsf{v}}(L)$. We write $\mathbb{L}^X_{s,k}$ for the set of limit expressions L with free names in X that have $\mathrm{nest}_{\mathsf{v}}(L) \leq k$ and do not contain messages of size exceeding s. We often omit X and understand it is a fixed finite set of names.*

▶ **Theorem 19.** *Limits faithfully represent ideals: $I \in \mathrm{Idl}(\mathbb{D}_{s,k}) \iff \exists L \in \mathbb{L}_{s,k} : I = [\![L]\!]$.*

3.4 Decidability of Inclusion

Now we turn to decidability of inclusion between downward-closed sets. It is well-known that in a wqo, given the ideal decomposition of two downward-closed sets $D_1 = I_1 \cup \ldots \cup I_n$ and $D_2 = J_1 \cup \ldots \cup J_m$, we have $D_1 \subseteq D_2$ if and only if for all $1 \leq i \leq n$, there is a $1 \leq j \leq m$, such that $I_i \subseteq J_j$. Hence, decidability of ideals inclusion implies decidability of downward-closed sets inclusion.

We extend structural congruence to limits in the obvious way, with the addition of the law $\langle M \rangle^{\omega} \equiv \langle M \rangle$ obtaining that $L \equiv L'$ implies $[\![L]\!] = [\![L']\!]$. We can define a standard form for limits: every limit is structurally congruent to a limit of the form $\mathsf{v}\vec{x}.(\langle \Gamma \rangle \parallel \prod_{i \in I} \mathsf{Q}_i[\vec{M}_i] \parallel \prod_{j \in J} B_j^{\omega})$ where every name in $\vec{x}$ occurs free at least once in the scope of the restriction, and

$$\lceil L \rceil^n := \begin{cases} L & \text{if } L \text{ is sequential or } \mathbf{0} \\ \lceil L_1 \rceil^n \parallel \lceil L_2 \rceil^n & \text{if } L = L_1 \parallel L_2 \\ \nu x.(\lceil L' \rceil^n) & \text{if } L = \nu x.L' \\ (\lceil B \rceil^n)^n & \text{if } L = B^\omega \end{cases} \qquad L \otimes n := \begin{cases} L & \text{if } L \text{ is sequential or } \mathbf{0} \\ L_1 \otimes n \parallel L_2 \otimes n & \text{if } L = L_1 \parallel L_2 \\ \nu x.(L' \otimes n) & \text{if } L = \nu x.L' \\ (B \otimes n)^n \parallel B^\omega & \text{if } L = B^\omega \end{cases}$$

Figure 4 The grounding $\lceil\text{-}\rceil^n \colon \mathbb{L}_{s,k} \to \mathbb{D}_{s,k}$ and extension $\text{-} \otimes n \colon \mathbb{L}_{s,k} \to \mathbb{L}_{s,k}$ operations on limits.

for all $j \in J$, B_j is also in standard form. When we write $\mathrm{sf}(L) \overset{\alpha}{=} \nu\vec{x}.(\langle\Gamma\rangle \parallel Q \parallel R)$ we imply that Q is a parallel composition of process calls $\prod_{i \in I} \mathsf{Q}_i[\vec{M}_i]$ (in which case we write $|Q|$ for $|I|$) and R is a parallel composition of iterated limits $\prod_{j \in J} B_j^\omega$.

To better manipulate limits, we introduce, in Figure 4, the n-th grounding $\lceil L \rceil^n$ and the n-th extension $L \otimes n$, of a limit L. Grounding replaces each -^ω with -^n, with the obvious property that $\lceil L \rceil^n \in [\![L]\!]$. An extension $L \otimes n$ produces a new limit with each sub-limit B^ω unfolded n times. Note that extension does not alter semantics: $[\![L]\!] = [\![L \otimes n]\!]$.

The absorption axiom. The decidability proof hinges on a characterisation of inclusion that requires an additional hypothesis on the intruder model.

▶ **Definition 20** (Absorbing intruder). *Fix an intruder model* $\mathbb{I} = (\Sigma, \vdash)$. *Let* $\vec{x}$ *and* $\vec{y}$ *be two lists of pairwise distinct names,* Γ *be a finite set of messages, and* $\Gamma' = \Gamma[\vec{y}/\vec{x}]$. *Moreover, assume that* $\mathrm{names}(\Gamma) \cap \vec{y} = \emptyset$. *We say* $\mathbb{I}$ *is* absorbing *if, for all messages* M *with* $\mathrm{names}(M) \subseteq \mathrm{names}(\Gamma)$, *we have that* $\Gamma, \Gamma' \vdash M$ *if and only if* $\Gamma \vdash M$.

▶ **Lemma 21.** $\mathbb{I}_{\mathsf{en}}$ *is absorbing.*

For the rest of the paper, we assume an absorbing intruder model. The absorption axiom has a technical definition, which becomes more intuitive if understood in the context of limits of the form $L = \big(\nu\vec{x}.(\langle\Gamma\rangle \parallel Q)\big)^\omega$. Imagine comparing the difference in knowledge between $\lceil L \rceil^1$ and $\lceil L \rceil^2$: we have $\mathrm{sf}(\lceil L \rceil^2) \overset{\alpha}{=} \big(\nu\vec{x}.\nu\vec{x}'.(\langle\Gamma\rangle \parallel \langle\Gamma'\rangle \parallel Q \parallel Q')\big)$, where $\Gamma' = \Gamma[\vec{x}'/\vec{x}]$. The absorption axiom tells us that if we want to check whether a process $\nu\vec{x}.\langle M \rangle$ is embedded in $\mathrm{sf}(\lceil L \rceil^2)$, we only need to check if M is derivable from Γ and we can ignore Γ'. In other words, we only need to check if $\nu\vec{x}.\langle M \rangle$ is embedded in $\lceil L \rceil^1$.

We are now ready to prove our main result: a small model property that shows decidability of limit inclusion. Let us present the intuition on the simpler problem of deciding inclusion when one of the limits is a single process P, i.e. deciding if $P \in [\![L]\!]$. Take $P = \nu a, b.(\langle \mathsf{e}(a)_b \rangle \parallel \mathsf{A}[b])$ and $L = \big(\nu x.(\langle x \rangle \parallel A[x])\big)^\omega$. Suppose we replicate the ω twice, obtaining the equivalent limit $L \otimes 2 \equiv \nu x_0, x_1.(\langle x_0 \rangle \parallel A[x_0] \parallel \langle x_1 \rangle \parallel A[x_1] \parallel L)$. The idea is that we can match P against the fixed part of $L \otimes 2$:

$$\begin{aligned} P \equiv_{\mathsf{kn}} \nu x_0, x_1.(\langle \mathsf{e}(x_0)_{x_1} \rangle \parallel \mathsf{A}[x_1]) &\sqsubseteq_{\mathsf{kn}} \nu x_0, x_1.(\langle \mathsf{e}(x_0)_{x_1} \rangle \parallel A[x_0] \parallel A[x_1]) \\ &\sqsubseteq_{\mathsf{kn}} \nu x_0, x_1.(\langle x_0 \rangle \parallel \langle x_1 \rangle \parallel A[x_0] \parallel A[x_1]) \in L \otimes 2 \end{aligned}$$

The first observation is therefore that if we can find some m such that P is embedded in the fixed part of $L \otimes m$, we have proven $P \in [\![L]\!]$. To turn this into an algorithm, we need to prove that there exists an n so that if we failed to embed P in the fixed part of $L \otimes m$, for any $m \leq n$ then P is not going to embed in $L \otimes m'$ for *every* m', and therefore $P \notin [\![L]\!]$. In other words, we need to know that after some threshold n, there is no point trying with bigger extensions. Take for example $P = \nu x, y.(\mathsf{B}[x, y] || B[y, x])$ and $L = (\nu x, y.\mathsf{B}[x, y])^\omega$. We can try and embed P into $L \otimes 2$ but we would fail as the fixed part expands to $\nu x_0, y_0.\mathsf{B}[x_0, y_0] \parallel \nu x_1, y_1.\mathsf{B}[x_1, y_1]$. It is easy to see that expanding further would not introduce new patterns in the fixed part of the limit which would help embed P.

Theorem 22 formalises the idea for general inclusion between two arbitrary limits L_1 and L_2: it proves that the threshold for expansion is the number of fixed restrictions of L_1 plus the number of fixed process calls of L_1, plus one; and it makes use of the absorption axiom to prove the threshold is sound even in the presence of knowledge.

▶ **Theorem 22** (Characterisation of Limits Inclusion). *Let L_1 and L_2 be two limits, with $\mathrm{sf}(L_1) \overset{\alpha}{=} \nu\vec{x}_1.(\langle\Gamma_1\rangle \parallel Q_1 \parallel \prod_{i\in I} B_i^{\omega})$, and let $n = |\vec{x}_1| + |Q_1| + 1$. Then:*

$$[\![L_1]\!] \subseteq [\![L_2]\!] \iff \begin{cases} \mathrm{sf}(L_2 \otimes n) \overset{\alpha}{=} \nu\vec{x}_1, \vec{x}_2.(\langle\Gamma_2\rangle \parallel Q_1 \parallel Q_2 \parallel R_2) \;\; \textit{and} \;\; \Gamma_1 \leq_{\mathsf{kn}} \Gamma_2 & \textbf{(A)} \\ [\![\langle\Gamma_1\rangle \parallel \prod_{i\in I} B_i]\!] \subseteq [\![\langle\Gamma_2\rangle \parallel R_2]\!] & \textbf{(B)} \end{cases}$$

▶ **Theorem 23.** *Given $L_1, L_2 \in \mathbb{L}$ it is decidable whether $[\![L_1]\!] \subseteq [\![L_2]\!]$.*

Proof. Theorem 22 leads to a recursive algorithm. Given L_1 and L_2, one computes $\mathrm{sf}(L_1)$ and $\mathrm{sf}(L_2 \otimes n)$. For every α-renaming that makes condition **(A)** hold, one checks condition **(B)** (recursively). If no renaming makes both true then the inclusion does not hold. In the recursive case, there are fewer occurrences of ω in the limit on the left, eventually leading to the case where L_1 has no occurrence of ω, and only condition **(A)** needs to be checked. ◀

▶ **Example 24** (Limit inclusion). Consider the following two limits:

$$L_1 = \nu x_1.\big((\nu x_2.(\langle \mathsf{e}(x_2)_{x_1}\rangle \parallel \mathsf{A}[x_2] \parallel \mathsf{A}[x_1]))^{\omega}\big)$$
$$L_2 = \nu y_1, y_3.\big(\langle y_3\rangle \parallel \mathsf{A}[y_3]^{\omega} \parallel (\nu y_2.(\langle y_2\rangle \parallel \mathsf{A}[y_2]))^{\omega}\big)$$

We prove that $[\![L_1]\!] \subseteq [\![L_2]\!]$ by applying the recursive algorithm from Theorem 23. By Theorem 22, here the threshold for expansion is $n = 2$, but we will try with $n = 1$. In case we succeed, the inclusion holds. If not, we might have to increase n up to 2. This results in

$$\begin{aligned} L_2 \otimes 1 &= \nu y_1, y_3.\big(\langle y_3\rangle \parallel \mathsf{A}[y_3] \parallel \mathsf{A}[y_3]^{\omega} \parallel (\nu y_2.(\langle y_2\rangle \parallel \mathsf{A}[y_2]))^{\omega} \parallel (\nu y_2.(\langle y_2\rangle \parallel \mathsf{A}[y_2]))\big) \\ &\equiv_{\mathsf{kn}} \nu y_1, y_{22}, y_3.\big(\langle y_3\rangle \parallel \langle y_{22}\rangle \parallel \mathsf{A}[y_{22}] \parallel \mathsf{A}[y_3] \parallel \mathsf{A}[y_3]^{\omega} \parallel (\nu y_2.(\langle y_2\rangle \parallel \mathsf{A}[y_2]))^{\omega}\big) \end{aligned}$$

To try and match the fixed part of L_1 with $L_2 \otimes 1$, we could α-rename x_1 to y_1. This works for **(A)** but the remaining goal **(B)** cannot be shown as it does not hold because we cannot derive $\langle \mathsf{e}(x_2)_{y_1}\rangle$ from the knowledge on the right-hand side:

$$[\![\nu x_2.(\langle \mathsf{e}(x_2)_{y_1}\rangle \parallel \mathsf{A}[x_2] \parallel \mathsf{A}[x_1])]\!] \not\subseteq [\![\langle y_3\rangle \parallel \langle \mathsf{e}(y_{22})_{y_3}\rangle \parallel \mathsf{A}[x_1]^{\omega} \parallel (\nu y_2.(\langle \mathsf{e}(y_2)_{y_3}\rangle \parallel \mathsf{A}[y_2]))^{\omega}]\!]$$

Choosing the α-renaming $[x_1/y_3]$ leaves us instead with:

$$[\![\nu x_2.(\langle \mathsf{e}(x_2)_{y_3}\rangle \parallel \mathsf{A}[x_2] \parallel \mathsf{A}[x_1])]\!] \subseteq [\![\langle y_3\rangle \parallel \langle \mathsf{e}(y_{22})_{y_3}\rangle \parallel \mathsf{A}[x_1]^{\omega} \parallel (\nu y_2.(\langle \mathsf{e}(y_2)_{y_3}\rangle \parallel \mathsf{A}[y_2]))^{\omega}]\!]$$

which holds. It suffices to expand the latter limit by 1 and to choose the renaming $[x_2/y_2]$. Then, $\mathsf{e}(x_1)_{x_2} \leq_{\mathsf{kn}} x_1, x_2$ holds and the process calls match. Note that it is crucial to keep $\mathsf{A}[x_1]^{\omega}$ on the right side.

3.5 Computing Post-Hat

The last result we need is the decidability of a function $\widehat{\mathrm{post}}^{s}_{\Delta}(L)$ which, given a limit L, returns a finite set of limits $\{L_1, \ldots, L_n\}$ such that $\mathrm{post}^{s}_{\Delta}([\![L]\!])\!\downarrow = [\![L_1]\!] \cup \ldots \cup [\![L_n]\!]$. The challenge is representing all the possible successors of processes in $[\![L]\!]$ without having to

$$
\begin{aligned}
L &= \nu a, b, k_{as}, k_{bs}.(\langle a, b\rangle \parallel \mathsf{A}_1[a, b, k_{as}]^\omega \parallel \mathsf{B}_1[a, b, k_{bs}]^\omega \parallel \mathsf{S}[a, b, k_{as}, k_{bs}]^\omega \parallel L_1^\omega)\\
L_1 &= \nu n_a.\big(\langle n_a\rangle \parallel \mathsf{A}_2[a, b, k_{as}, n_a] \parallel L_2^\omega\big)\\
L_2 &= \nu k.\big(\langle \mathsf{e}(k)_{(a,k_{as})}\rangle \parallel \langle \mathsf{e}(k)_{(b,k_{as})}\rangle \parallel \langle \mathsf{e}(k)_{(n_a,k_{as})}\rangle \parallel \langle \mathsf{e}(k)_{k_{bs}}\rangle \parallel \mathsf{Secret}[k]^\omega \parallel \mathsf{A}_3[a, b, k_{as}, k]^\omega \parallel L_3^\omega\big)\\
L_3 &= \nu n_b.\big(\langle \mathsf{e}(n_b)_{(k,k)}\rangle \parallel \langle \mathsf{e}(n_b)_k\rangle \parallel \mathsf{B}_2[a, b, k_{bs}, n_b, k]\big)
\end{aligned}
$$

Figure 5 An inductive invariant for Example 9.

enumerate $[\![L]\!]$. The key idea hinges again on the absorption axiom: we observe that to consider all possible process calls that may cause a transition, it is enough to unfold each ω in L by some bounded number b. Any transition taken from further unfoldings will give rise to successors that are congruent to some of the ones already considered.

The bound b used in extending a limit, is defined as a function of the process definitions and the intruder model. The arity of a pattern $\vec{x} : M$ is $|\vec{x}|$. Given a set of definitions Δ, $\beta(\Delta)$ is the maximum arity of patterns in Δ. The function $\gamma(\mathbb{I})$ returns the maximum arity of the constructors in the signature of $\mathbb{I}$.

▶ **Definition 25** ($\widehat{\text{post}}$). *Let* $b = \beta(\Delta) \cdot \gamma(\mathbb{I})^{s-1} + 1$ *and* $\mathrm{sf}(L \otimes b) = \nu\vec{x}.(\langle\Gamma\rangle \parallel Q \parallel R)$*,*

$$\widehat{\text{post}}^s_\Delta(L) := \left\{\ \nu\vec{y}.(\langle\Gamma'\rangle \parallel Q' \parallel R)\ \middle|\ \nu\vec{x}.\big(\langle\Gamma\rangle \parallel Q\big) \to_\Delta \nu\vec{y}.\big(\langle\Gamma'\rangle \parallel Q'\big) \in \mathbb{S}_s\ \right\}.$$

▶ **Theorem 26.** $\widehat{\text{post}}^s_\Delta(L) = \{L_1, \dots, L_n\} \implies \text{post}^s_\Delta([\![L]\!]){\downarrow} = [\![L_1]\!] \cup \ldots \cup [\![L_n]\!]$.

3.6 Algorithmic Aspects

The limit L in Figure 5 represents an inductive invariant for Example 9, under size bound 3. It can be proven invariant by checking that $\widehat{\text{post}}^3_\Delta(L)$ is included in L. Since the initial process of Example 9, P_0, is in $[\![L]\!]$, the invariant certificates that any reachable process is $(3, 7)$-bounded (note $\text{nest}_\nu(L) = 7$) and satisfies secrecy. In fact, L^ω is also inductive, proving boundedness and secrecy for any process in $(P_0)^\omega$. Since $[\![\nu k.(\nu x.(\langle x, \mathsf{e}(x)_k\rangle))^\omega]\!] \not\subseteq [\![L^\omega]\!]$, L^ω provides proof that the protocol is not susceptible to known-plaintext attacks where arbitrary known nonces are available to the intruder encrypted with the same key.[2] The algorithms for inclusion and $\widehat{\text{post}}$ can be used to check inductiveness given a candidate invariant such as L. This leaves open the question of how to efficiently generate candidates. The algorithm of Theorem 16 can in principle be used to enumerate all candidate invariants, with an impractically high complexity. Luckily, a more directed inference of invariants can be obtained by a *widening operator*, in the style of [37]; in fact, the invariant L was automatically inferred from P_0 using the widening of our prototype tool. The basic observation behind invariant inference, is that from a sequence of transitions $P_1 \to^* P_2$ with $P_1 \sqsubseteq_{\mathsf{kn}} P_2$, one can deduce that the same sequence can be simulated from P_2 (by Theorem 8) obtaining a P_3 with $P_2 \sqsubseteq_{\mathsf{kn}} P_3$ and so on. The embedding between P_1 and P_2, is justified by $P_1 \equiv_{\mathsf{kn}} \nu\vec{x}.(\langle\Gamma_1\rangle \parallel Q)$, $P_2 \equiv_{\mathsf{kn}} \nu\vec{x}.(\langle\Gamma_1\rangle \parallel Q \parallel P)$; we can extrapolate the difference P and accelerate the sequence of transitions by constructing the limit $\nu\vec{x}.(\langle\Gamma_1\rangle \parallel Q \parallel P^\omega)$. This operation can be extended to limits. The end product is a finite union of limits which is inductive by construction. This procedure requires exploration of transitions and many inclusion checks, a costly combination. To obtain a more practical algorithm, we devised two techniques: inductiveness checks through "incorporation", and a coarser widening.

[2] The property could be extended to cover composite known-plaintext messages (of some maximum size s), and the generation of a sufficiently high number N of plaintext-encrypted pair with the same key.

Table 1 Experimental results. Columns: **Infer**ence of invariant fully automatic (F) or interactive (I); **C**heck of inductiveness; **S**ecrecy proved (✓), not holding (×), not modelled (○).

Name	**Infer**	**C**	**S**	**Name**	**Infer**	**C**	**S**	**Name**	**Infer**	**C**	**S**
Ex.9	4.4s F	1.8s	✓	NHS	5.0s F	1.6s	○	YAH	7.8s F	2.5s	○
OR	3.4s F	1.9s	○	NHSs	6.8s F	1.7s	✓	YAHs1	12.3s F	2.6s	✓
ORl	25.0s F	3.5s	×	NHSr	90.9s I	20.8s	×	YAHs2	8.8s F	2.0s	✓
ORa	13.8s I	5.0s	○	KSL	37.0s F	9.8s	✓	YAHlk	11.9s I	17.3s	✓
ORs	9.8s F	2.0s	✓	KSLr	200.6s F	31.4s	✓	ARPC	0.5s F	0.1s	✓

Consider the inductiveness check $\llbracket \widehat{\mathrm{post}}^{s}_{\Delta}(L) \rrbracket \subseteq \llbracket L \rrbracket$ implemented by checking that, for each transition considered by $\widehat{\mathrm{post}}$ the resulting limit L' is included by L. We observe that L' and L will share the context of the transition: L can be rewritten to $C[\mathsf{Q}[\vec{M}]]$ and L' to $C[P]$ for some P. To prove inclusion of $C[P]$ in $C[\mathsf{Q}[\vec{M}]]$ it is then sufficient to show that P is embedded in $C[\mathsf{Q}[\vec{M}]]$. We call this check an *incorporation* of P in C. See Appendix D for an example. Although incomplete in general, incorporation can prove inductiveness in many practical examples, in a remarkably faster way.

There are cases, however, where the incorporation check fails on inductive invariants. Consider for example an inductive invariant represented by the union of two incomparable limits L_1, L_2. Suppose that for some $P \in \llbracket L_1 \rrbracket$ there is P' with $P \to P' \in \llbracket L_2 \rrbracket \setminus \llbracket L_1 \rrbracket$. Then, incorporation of P' in L_1 would fail. To side-step this problem we replace union of limits with parallel composition: $\llbracket L_1 \rrbracket \cup \llbracket L_2 \rrbracket \subseteq \llbracket L_1 \parallel L_2 \rrbracket$ by downward closure of $\llbracket \text{-} \rrbracket$. Using this over-approximation, we can try to aim for an inductive invariant which consists of exactly one limit, and for which the incorporation check suffices. This approximation is incomplete: some protocols require inductive invariants consisting of unions of incomparable limits.

4 Evaluation

We built a proof-of-concept tool implementing limit inclusion, inductivity check, incorporation and a coarse widening. Currently, the tool only supports symmetric encryption, but we plan to add support for asymmetric encryption, signatures and hashing. The source code and all the test protocol models are available at [20] where we also provide a tutorial-style explanation of the methodology. We summarise our experiments[3] in Table 1. The tool is instructed to compute, using the widening, an invariant for the provided model, under given message size assumptions. When an invariant can be found, it represents a proof that the model is depth-bounded. If the inferred invariant is leak-free, secrecy is also proven. We model a number of well-known protocols under various threat scenarios (e.g. with or without leak of old keys): Needham-Schröder (NHS), Otway-Rees (OR), Kehne-Schönwälder-Landendörfer (KSL), Yahalom (YAH) and Andrew RPC (ARPC). For any example containing a problematic encryption oracle pattern (see Example 15) we constrain the input message of the oracle to be a nonce (message of size 1). With the exception of NHSr, ORa and YAHlk, all the invariants were obtained fully automatically. For YAHlk we had to combine two widened limits (the timing is the sum of the time spent computing each limit); for NHSr and ORa we had to tweak a partially widened limit manually to make it inductive. To simulate using invariants as correctness certificates, for each example we re-checked them for inductiveness.

[3] Tests run with Python 2.7, z3-solver v4.8, 8GB RAM, Intel CPU i5, on Linux.

4.1 Limitations

Message size bounds. Bounding message size in the analysis is not always acceptable, as it may miss type confusion attacks. Such patterns arise, for example, in XOR-based protocols, where considering only typed runs is not appropriate. This is problematic because in most practical examples bigger size bounds lead to unboundedness in depth, as illustrated by Example 15. However, not all is lost: our limits can be extended to provide invariants with unbounded message size. The idea, detailed in Appendix B, is to first analyse a protocol with a size bound that ensures depth-boundedness, obtaining an inductive invariant as one or more limits; then we annotate such limits with "wildcards" ($\star$) on occurrences of names. A wildcarded name $a^\star$ stands for an arbitrary message (of arbitrary size). By introducing the appropriate wildcards, one can obtain an inductive invariant for the protocol with no message size bounds. For Example 15, $\mathsf{E}[k] \parallel \big(\nu x.\langle \mathsf{e}(x)_k\rangle\big)^\omega$ is an inductive invariant if we restrict x to be of size 1. Since x is never inspected , injecting larger terms in it would not lead to new behaviour. We can therefore generalise the limit to $\mathsf{E}[k] \parallel \big(\nu x.\langle \mathsf{e}(x^\star)_k\rangle\big)^\omega$ which is an inductive invariant for the protocol with no size bounds.

Since verification with no size bounds is undecidable, this extension is by necessity incomplete: there are downward-closed sets that cannot be represented by extended limits. However, since protocols typically achieve resistance to type confusion attacks by making sure that messages of the wrong type are discarded by honest principals, this extension is very effective. In fact, all the size bounds in our benchmarks can be lifted using this technique.

Inherent unboundedness. There are two ways a protocol may fall outside of our class. The first is when unboundedly many participants in a session form a ring/list topology, like the recursive protocol of [30, §6]. One can provide a partial solution by using an under-approximate model with a ring/list of fixed size, or an over-approximate model by using an unbounded star topology. The second way is when the intruder can produce irreducible encryption chains and the participants would inspect them generating new behaviour. In such cases not even extended limits can help. We deem this situation unlikely to be desirable in a realistic protocol.

5 Conclusions and Future Work

We presented a theory of decidable inductive invariants for depth-bounded cryptographic protocols. We showed how one can infer inductive invariants and evaluated the approach through a prototype implementation.

From a theoretical perspective, it would be interesting to determine precise complexity bounds for inclusion, for general intruder models. We can show that $\sqsubseteq_{\mathsf{kn}}$ is NP-complete for any intruder model that has polynomially decidable $\leq_{\mathsf{kn}}$ [33].

A direction for further improvement is extending the class of supported properties. In particular, we plan to study how invariants can be used to automatically prove diff-equivalence [5] without bounding sessions/nonces.

Finally, we intend to explore ways in which our invariants can be integrated in existing tools such as ProVerif and Tamarin. For instance, Tamarin performs a backwards search to find possible attacks. Our invariants could provide a pruning technique to avoid exploring paths that are unreachable from the initial state. Similar combinations of forward and backward search have been shown to improve performance dramatically for analyses of infinite-state systems such as Petri nets [7].

References

1 Martín Abadi and Véronique Cortier. Deciding knowledge in security protocols under (many more) equational theories. In *CSFW*, pages 62–76. IEEE Computer Society, 2005.

2 Myrto Arapinis and Marie Duflot. Bounding messages for free in security protocols - extension to various security properties. *Inf. Comput.*, 239:182–215, 2014.

3 Alessandro Armando, David A. Basin, Yohan Boichut, Yannick Chevalier, Luca Compagna, Jorge Cuéllar, Paul Hankes Drielsma, Pierre-Cyrille Héam, Olga Kouchnarenko, Jacopo Mantovani, Sebastian Mödersheim, David von Oheimb, Michaël Rusinowitch, Judson Santiago, Mathieu Turuani, Luca Viganò, and Laurent Vigneron. The AVISPA tool for the automated validation of internet security protocols and applications. In *CAV*, volume 3576 of *Lecture Notes in Computer Science*, pages 281–285. Springer, 2005.

4 Bruno Blanchet. Modeling and verifying security protocols with the applied pi calculus and proverif. *Foundations and Trends in Privacy and Security*, 1(1-2):1–135, 2016.

5 Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated verification of selected equivalences for security protocols. *J. Log. Algebr. Program.*, 75(1):3–51, 2008. `doi:10.1016/j.jlap.2007.06.002`.

6 Bruno Blanchet and Andreas Podelski. Verification of cryptographic protocols: tagging enforces termination. *Theor. Comput. Sci.*, 333(1-2):67–90, 2005.

7 Michael Blondin, Alain Finkel, Christoph Haase, and Serge Haddad. Approaching the coverability problem continuously. In *TACAS*, volume 9636 of *Lecture Notes in Computer Science*, pages 480–496. Springer, 2016.

8 Michael Blondin, Alain Finkel, and Pierre McKenzie. Handling infinitely branching well-structured transition systems. *Inf. Comput.*, 258:28–49, 2018.

9 Michael Burrows, Martín Abadi, and Roger M. Needham. A logic of authentication. In *SOSP'89*, pages 1–13. ACM, 1989.

10 Rohit Chadha, Vincent Cheval, Ştefan Ciobâcă, and Steve Kremer. Automated verification of equivalence properties of cryptographic protocols. *ACM Trans. Comput. Log.*, 17(4):23:1–23:32, 2016.

11 Vincent Cheval, Steve Kremer, and Itsaka Rakotonirina. DEEPSEC: deciding equivalence properties in security protocols theory and practice. In *IEEE Symposium on Security and Privacy*, pages 529–546. IEEE Computer Society, 2018.

12 Yannick Chevalier, Ralf Küsters, Michaël Rusinowitch, and Mathieu Turuani. An NP decision procedure for protocol insecurity with XOR. In *LICS*, pages 261–270. IEEE Computer Society, 2003.

13 Rémy Chrétien, Véronique Cortier, and Stéphanie Delaune. Typing messages for free in security protocols: The case of equivalence properties. In *CONCUR*, volume 8704 of *Lecture Notes in Computer Science*, pages 372–386. Springer, 2014.

14 Rémy Chrétien, Véronique Cortier, and Stéphanie Delaune. Decidability of trace equivalence for protocols with nonces. In *CSF'15*, pages 170–184. IEEE Computer Society, 2015.

15 Véronique Cortier, Niklas Grimm, Joseph Lallemand, and Matteo Maffei. A type system for privacy properties. In *ACM Conference on Computer and Communications Security*, pages 409–423. ACM, 2017.

16 Morten Dahl, Naoki Kobayashi, Yunde Sun, and Hans Hüttel. Type-based automated verification of authenticity in asymmetric cryptographic protocols. In *ATVA*, volume 6996 of *Lecture Notes in Computer Science*, pages 75–89. Springer, 2011.

17 Danny Dolev and Andrew Chi-Chih Yao. On the security of public key protocols. *IEEE Trans. Information Theory*, 29(2):198–207, 1983.

18 Emanuele D'Osualdo, Luke Ong, and Alwen Tiu. Deciding secrecy of security protocols for an unbounded number of sessions: The case of depth-bounded processes. In *CSF*, pages 464–480. IEEE Computer Society, 2017.

19 Emanuele D'Osualdo and Felix Stutz. Decidable inductive invariants for verification of cryptographic protocols with unbounded sessions. *CoRR*, abs/1911.05430, 2019. arXiv:1911.05430.

20 Emanuele D'Osualdo and Felix Stutz. Lemma9, version 1.0, 2020. doi:10.5281/zenodo.3950846.

21 Nancy A. Durgin, Patrick D. Lincoln, John C. Mitchell, and Andre Scedrov. Undecidability of bounded security protocols. In *Workshop on Formal Methods and Security Protocols*, 1999.

22 Alain Finkel and Jean Goubault-Larrecq. Forward analysis for WSTS, part I: completions. In *STACS*, volume 3 of *LIPIcs*, pages 433–444, 2009.

23 Sibylle B. Fröschle. Leakiness is decidable for well-founded protocols. In *POST'15*, pages 176–195, 2015.

24 Reiner Hüchting, Rupak Majumdar, and Roland Meyer. Bounds on mobility. In *CONCUR*, volume 8704 of *Lecture Notes in Computer Science*, pages 357–371. Springer, 2014.

25 Gavin Lowe. Some new attacks upon security protocols. In *Ninth IEEE Computer Security Foundations Workshop, March 10 - 12, 1996, Dromquinna Manor, Kenmare, County Kerry, Ireland*, pages 162–169. IEEE Computer Society, 1996.

26 Simon Meier, Benedikt Schmidt, Cas Cremers, and David A. Basin. The TAMARIN prover for the symbolic analysis of security protocols. In *CAV*, volume 8044 of *Lecture Notes in Computer Science*, pages 696–701. Springer, 2013.

27 Roland Meyer. On boundedness in depth in the π-calculus. In Giorgio Ausiello, Juhani Karhumäki, Giancarlo Mauri, and C.-H. Luke Ong, editors, *IFIP TCS*, volume 273 of *IFIP*, pages 477–489. Springer, 2008.

28 Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, 1978.

29 David J. Otway and Owen Rees. Efficient and timely mutual authentication. *Operating Systems Review*, 21(1):8–10, 1987.

30 Lawrence C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1-2):85–128, 1998.

31 Mark D. Ryan and Ben Smyth. Applied pi-calculus. In *Formal Models and Techniques for Analyzing Security Protocols*, volume 5 of *Cryptology and Information Security Series*, pages 112–142. IOS Press, 2011.

32 Benedikt Schmidt, Simon Meier, Cas J. F. Cremers, and David A. Basin. Automated analysis of Diffie-Hellman protocols and advanced security properties. In *CSF*, pages 78–94. IEEE Computer Society, 2012.

33 Felix Stutz. Decidable inductive invariants for verification of cryptographic protocols with unbounded sessions. Master's thesis, Saarland University, 2019.

34 Alwen Tiu, Rajeev Goré, and Jeremy E. Dawson. A proof theoretic analysis of intruder theories. *Logical Methods in Computer Science*, 6(3), 2010.

35 Alwen Tiu, Nam Nguyen, and Ross Horne. SPEC: an equivalence checker for security protocols. In *APLAS*, volume 10017 of *Lecture Notes in Computer Science*, pages 87–95, 2016.

36 Thomas Wies, Damien Zufferey, and Thomas A. Henzinger. Forward analysis of depth-bounded processes. In *FoSSaCS*, volume 6014 of *Lecture Notes in Computer Science*, pages 94–108. Springer, 2010.

37 Damien Zufferey, Thomas Wies, and Thomas A. Henzinger. Ideal abstractions for well-structured transition systems. In *VMCAI'12'*, volume 7148 of *LNCS*, pages 445–460. Springer, 2012.

A Support for Other Cryptographic Primitives

We claim the assumptions we make on the intruder model are mild, and are satisfied by the symbolic models of many cryptographic primitives. We illustrated the treatment of (a)symmetric encryption; treatment of hashes and blind signatures is entirely analogous. By using the sequent calculus formalisation of [34] one can trivially extend our proofs to prove all these primitives form an effective absorbing intruder.

Supporting XOR requires a bit more analysis on the algebraic properties of the primitives. We take as reference the model of XOR analysed in [1, 12]. The two constructors $\oplus$ (of arity 2) and 0 (of arity 0) are added to the set of constructors. Their algebraic properties are formalised through a congruence relation:

$$M_1 \oplus (M_2 \oplus M_3) \cong (M_1 \oplus M_2) \oplus M_3 \qquad M_1 \oplus M_2 \cong M_2 \oplus M_1 \tag{1}$$

$$M \oplus 0 \cong M \qquad M \oplus M \cong 0 \tag{2}$$

The results of [1, 12] establish that the laws (2) can be always orientated from left to right. Formally, one can define the rewriting system $\rightsquigarrow$ with two rules $M \oplus 0 \rightsquigarrow M$ and $M \oplus M \rightsquigarrow 0$; and the congruence $\cong_{\mathsf{AC}}$ defined by laws (1). Then the relation $\rightsquigarrow_{\mathsf{AC}} := (\cong_{\mathsf{AC}} \circ \rightsquigarrow)$ is terminating, and confluent modulo $\cong_{\mathsf{AC}}$. The set of normal forms $M\Downarrow := \{N \mid M \rightsquigarrow^*_{\mathsf{AC}} N \not\rightsquigarrow_{\mathsf{AC}}\}$ is then guaranteed to be finite, computable, and such that $M \cong N \iff (M\Downarrow \cap N\Downarrow) \neq \emptyset$.

We can harmonise the equational theory of XOR with the deduction system of Figure 1 by adding the rules

$$\frac{}{\Gamma \vdash 0}\;\textsc{Xor}_0 \qquad \frac{\Gamma \vdash M_1 \qquad \Gamma \vdash M_2}{\Gamma \vdash M_1 \oplus M_2}\;\textsc{Xor}_\mathrm{R} \qquad \frac{\Gamma, M_1, M_2, M_1 \oplus M_2 \vdash N}{\Gamma, M_1, M_2 \vdash N}\;\textsc{Xor}_\mathrm{L}$$

$$\frac{\Gamma, M, M\Downarrow \vdash N}{\Gamma, M \vdash N}\;\Downarrow_\mathrm{L} \qquad \frac{\Gamma \vdash N \qquad N \in M\Downarrow}{\Gamma \vdash M}\;\Downarrow_\mathrm{R}$$

The deduction system accurately models XOR, even if it uses $\Downarrow$ instead of $\cong$: as proven in [12, Prop. 1], one can always restrict the intruder to manipulate messages in normal form without loosing expressive power.

We are left to prove that the derivability satisfies the effective absorbing intruder axioms. Decidability has been proven in [1, 12]. The axioms of Definitions 1 and 20 are easily satisfied by the same arguments we used for $\mathbb{I}_{\mathsf{en}}$. The proofs of (Relevancy) and absorption make use of the fact that if $N \in M\Downarrow$ then $\mathrm{fn}(N) \subseteq \mathrm{fn}(M)$.

We conjecture Diffie-Hellman exponentiation (following the model of e.g. [32]) can be shown to satisfy our axioms in the same way we treated XOR. The main issue with Diffie-Hellman, with respect to (Relevancy) and absorption, is the inverses law $M * M^{-1} \cong 1$: by using the law from right to left, one can involve arbitrary names in a derivation. This could be handled in the same way we handle cancellation of XOR, by normalising derivations so that the law is always applied left to right. We leave the formal development of this remark as future work.

A delicate point is the bounded message size assumption. With advanced primitives like XOR it is easier (but not inevitable) to encounter protocols for which it is impossible to extend the results on the bounded model to the unbounded case.

B Towards Unbounded Message Size

The analysis presented in Section 3 considers only traces (and attacks) involving messages of size smaller than some given bound s. We show here how the results of the analysis can be generalised to inductive invariants for the full set of traces, with no restriction on the size. Because of undecidability of the general problem, this generalisation will not be precise for every protocol: there are protocols for which the most precise generalised limit is trivial (i.e. does not ensure any non-trivial property of the protocol). For the protocols in our benchmarks however, one can get precise invariants by generalising the ones inferred by the tool.

We first introduce the syntax and semantics of generalised limits, and describe how we can use them to generalise our benchmarks. Finally, although studying how to automate this generalisation is beyond the scope of this paper, we briefly sketch how $\widehat{\text{post}}$ can be adapted to work on generalised limits.

Aside: finer size bounds via typing. In our development, we assumed a global size bound s. To have finer control on the message sizes, as we do in our tool, one can introduce a primitive form of typing. Assuming wlog that all pattern variables are unique, a *typing* is a partial function $\mathsf{ty}\colon \mathcal{N} \rightharpoonup \mathbb{N}$, assigning to each pattern variable a maximum size for the messages it can match. A typing induces a typed transition relation $\to_{\Delta,\mathsf{ty}}$ which only matches patterns with subsitutions respecting ty; $\text{reach}^{\mathsf{ty}}_{\Delta}(P)$ collects all the terms reachable from P through $\to_{\Delta,\mathsf{ty}}$. A typing ty of P, Δ is s-bounding if $\text{reach}^{\mathsf{ty}}_{\Delta}(P) \subseteq \mathbb{S}_s$. One can check if ty is s-bounding on-the-fly while computing $\widehat{\text{post}}$.

B.1 Generalised Limits

Recall the "encryption oracle" of Example 15: $E[k] = \mathbf{in}(x : x).(\langle \mathsf{e}(x)_k \rangle \parallel E[k])$. Without any restrictions on the pattern variable x, there is no way to prevent the encryption chains described in Example 15. However, we can use the 2-bounding typing $\mathsf{ty} = [x \mapsto 1]$, to obtain the limit $\mathsf{E}[k] \parallel \big(\nu m.\langle \mathsf{e}(m)_k \rangle\big)^{\omega}$ which is inductive (wrt $\to_{\Delta,\mathsf{ty}}$) and contains the initial state $\mathsf{E}[k]$. Since x is never inspected, injecting larger messages in it would not lead to new behaviour. We represent this arbitrary injection of messages in a limit by introducing a "wildcard" annotation on occurrences of names $a^{\star}$.

Technically, we duplicate the set of names $\mathcal{N}$ to a disjoint set of $\star$-annotated names $\mathcal{N}^{\star} := \{a^{\star} \mid a \in \mathcal{N}\}$, and we allow messages to contain names from $\mathcal{N} \uplus \mathcal{N}^{\star}$. All the definitions of this paper can be adapted straightforwardly to support wildcards by simply not distinguishing between a and $a^{\star}$. To stress the fact that a set of processes/limits contains annotations, we annotate the set with a wildcard, e.g. $\mathbb{L}^{\star}_{s,k}$.

▶ **Definition 27** (Wildcard semantics). *Given $P \in \mathbb{P}^{\star}$, its* wildcard semantics *is defined as*

$$\mathcal{W}[\![P]\!] := \left\{ P' \;\middle|\; \begin{array}{l} \mathrm{sf}(P) = \nu\vec{x}.(\langle\Gamma\rangle \parallel Q),\ P' \equiv_{\mathsf{kn}} \nu\vec{x},\vec{c}.\big((\langle\Gamma\rangle \parallel Q)[\vec{M}/\vec{y}^{\star}]\big) \\ \vec{y}^{\star} = \mathrm{names}(P) \cap \mathcal{N}^{\star}, \mathrm{names}(\vec{M}) \subseteq \vec{x} \cup \vec{c} \end{array} \right\}$$

For $L \in \mathbb{L}^{\star}_{s,k}$, define $\mathcal{W}[\![L]\!] := \{\, \mathcal{W}[\![P]\!] \mid P \in [\![L]\!] \,\}$.

With the help of wildcards, we can then take a limit L, annotate it with wildcards obtaining a limit $L^{\star}$. Then we can check that the wildcards generalise the limit enough to make it inductive wrt $\to_{\Delta}$ (i.e. without restricting the message size):

$$\forall P \in \mathcal{W}[\![L^\star]\!], \forall Q \colon P \to_\Delta Q \implies Q \in \mathcal{W}[\![L^\star]\!] \tag{3}$$

For our encryption oracle example, the annotated limit $\mathsf{E}[k] \parallel \big(\nu m.\langle \mathsf{e}(m^\star)_k\rangle\big)^\omega$ represents an inductive invariant for the unrestricted semantics.

For a more realistic application of generalised limits, consider our running Example 9. The limit of Figure 5 is inductive with respect to the typing $\mathsf{ty} = [n_a \mapsto 1]$. To remove the size bound we only need to annotate the occurrence of n_a in L_2 obtaining the limit $L^\star$:

$$\begin{aligned}
L^\star &= \nu a, b, k_{as}, k_{bs}.(\langle a, b\rangle \parallel \mathsf{A}_1[a, b, k_{as}]^\omega \parallel \mathsf{B}_1[a, b, k_{bs}]^\omega \parallel \mathsf{S}[a, b, k_{as}, k_{bs}]^\omega \parallel L_1^\omega)\\
L_1 &= \nu n_a.\big(\langle n_a\rangle \parallel \mathsf{A}_2[a, b, k_{as}, n_a] \parallel L_2^\omega\big)\\
L_2 &= \nu k.\big(\langle \mathsf{e}(k)_{(n_a^\star, k_{as})}\rangle \parallel \langle \mathsf{e}(k)_{k_{bs}}\rangle \parallel \mathsf{Secret}[k]^\omega \parallel \mathsf{A}_3[a, b, k_{as}, k]^\omega \parallel L_3^\omega\big)\\
L_3 &= \nu n_b.\big(\langle \mathsf{e}(n_b)_{(k,k)}\rangle \parallel \langle \mathsf{e}(n_b)_k\rangle \parallel \mathsf{B}_2[a, b, k_{bs}, n_b, k]\big)
\end{aligned}$$

Indeed this generalised limit is inductive: the intruder can send any message (M, b) to the server $\mathsf{S}[a, b, k_{as}, k_{bs}] := \mathbf{in}(x : (x, b)).\big(\dots \langle \mathsf{e}(k)_{(x,k_{as})}\rangle \dots\big)$, which will use it as part of the encryption key (M, k_{as}). None of the input patterns of the other processes would however be able to match the message $\mathsf{e}(k)_{(M,k_{as})}$ unless $M = n_a$; thus this attack attempt will not generate new behaviour, and our generalised limit captures all the reachable configurations without assuming bounds on the size of messages.

Similarly, it is not difficult to annotate the limits of our benchmarks[4] so that the generalised limits satisfy condition (3):

- ARPC: There is only one reasonable annotation: $(n_x^\star, (k, b))$. This is still inductive even though it can be fed back as we can have a different substitutions for $n_x^\star$.
- KSL: We annotate $\mathsf{B}_2[a, b, k_{bb}, k_{bs}, n_x^\star, n_y]$ in L_2 and $\mathsf{e}(nz, (b, n_x^\star,))_{k_{xy}}$ in L_4. A knows the actual n_x as it produced it and hence it is still inductive.
- KSLr: Additionally to KSL, we annotate $\mathsf{e}(m_x^\star, m_y)_{k_{xy}}$ and $\mathsf{B}_5[a, b, k_{bb}, k_{bs}, k_{ab}, t_y, m_x^\star, m_y]$ in L_7. The new message cannot flow into $\mathsf{B}_3[-]$ or $\mathsf{D}_2[-]$ as both pattern-match names on the second position which are different from m_y. However, due to the wildcard as parameter which could become m_y, $\mathsf{B}_4[-]$ can produce $\mathsf{B}_5[-]$’s input message. This is still covered and inductive but does not comply with a normal re-authentification run.
- NHS: We annotate $\mathsf{e}(n^\star, (b, \mathsf{e}(k, a)_{k_{bs}}))_{k_{as}}$ in L_2. A can only match on the original n so it is still inductive.
- NHSr: Same annotations as in NHS. Additionally, we annotate $\mathsf{e}(one, s^\star)_k$. in L_3 and hence have to annotate the s in $\mathsf{Secret}[s^\star]$ as well as $\mathsf{Leak}[s^\star]$. This covers all the enabled transitions and is hence inductive.
- NHSs: The same annotation as in NHS works.
- OR/ORl: We annotate $\mathsf{e}(n_y, (m^\star, (a, b)))_{k_{bs}}$ in L_2. This does not enable any new transition so it is inductive.
- ORs: Same annotation as OR/ORl. But we also have to annotate the key k_{xy} which we declare as secret which is the difference to ORl and hence does not work here.
- ORa: Similar annotation to OR/ORl. Analogously, inductivity is preserved.
- YAH: We annotate $\mathsf{e}(a, (n_a^\star, n_b))_{k_{bs}}$ in L_3. This can be fed back to $\mathsf{B}_2[-]$ and hence we also annotate $\mathsf{e}(n_b)_{(n_a^\star, n_b)}$ in L_5. This is still inductive.

[4] For the full limits for each benchmark see [20].

- YAHs1: We annotate as in YAH. The declaration of a secret happens only if A plays back and hence it is inductive.
- YAHs2: Without the size constraint for the key, the invariant obtained for YAHs1 is also one for YAHs2.
- YAHlk: We annotate $\mathsf{e}(one, s^\star)_k$ in L_4 which represents the case where the key k is leaked. This annotation covers all newly enabled transitions. In case k is not leaked, there are no new transitions. Hence, it is inductive.

B.2 Towards Automating Generalisation

Automating the check of condition (3) is beyond the scope of this paper, but we can sketch how one would approach the problem by adapting our $\widehat{\text{post}}$ definition to work on generalised limits. The idea is that one can design a computable function $\widehat{\text{post}}^{\mathsf{ty}}_{\Delta,\star}$, a "symbolic" version of $\widehat{\text{post}}$, and $\Subset$ satisfying:

$$L_1^\star \Subset L_2^\star \implies \mathcal{W}[\![L_1^\star]\!] \subseteq \mathcal{W}[\![L_2^\star]\!] \tag{4}$$

$$\text{post}(\mathcal{W}[\![L^\star]\!]) \subseteq \mathcal{W}[\![\widehat{\text{post}}^{\mathsf{ty}}_{\Delta,\star}(L^\star)]\!] \tag{5}$$

With these components, one can check (3) by checking $\widehat{\text{post}}^{\mathsf{ty}}_{\Delta,\star}(L^\star) \Subset L^\star$. Note that (4) is an implication, and (5) a subset relation: an equivalence would be impossible to achieve due to undecidability of the general problem; we therefore only require a sound over-approximation.

Designing $\Subset$ requires defining an approximate version of $\leq_{\mathsf{kn}}$ which can work on $\star$-annotated sets of messages. A precise version of this can only be defined on a per-intruder-model basis. A generic definition of $\Subset$ could simply extend the non-annotated inclusion check with the axioms $x^\star \leq_{\mathsf{kn}} x^\star$ and $M \leq_{\mathsf{kn}} x^\star$.

Designing a suitable $\widehat{\text{post}}^{\mathsf{ty}}_{\Delta,\star}(L^\star)$ similarly depends on the choice of intruder-model. One could, for example, define a symbolic matching function $\text{match}(\Gamma^\star, \vec{x} : N^\star)$ returning a finite set of substitutions such that

$$\Gamma \vdash N^\star\theta \wedge \Gamma \in \mathcal{W}[\![\Gamma^\star]\!] \implies \exists\sigma \in \text{match}(\Gamma^\star, \vec{x} : N^\star) \colon N^\star\theta \in \mathcal{W}[\![N^\star\sigma]\!]$$

and use it in $\widehat{\text{post}}^{\mathsf{ty}}_{\Delta,\star}$ to find all symbolic redexes. It is relatively straightforward to define a match function that is precise enough to check the generalised limits of our benchmarks. Exploring the design of these symbolic analyses in general is left for future work.

C Benchmarks

The Needham-Schröder protocol [28] is modelled with and without secrecy (NHS/NHSs). The NHSr version models leaks of old session keys, which leads to a replay attack (and hence the invariant is leaky). We provide four models of the Otway-Rees protocol [29]. OR does not model secrecy and is used to prove the protocol depth-bounded. ORl models secrecy but the inferred invariant contains a genuine leak, which is the result of a known type-confusion attack. The attack substitutes a composite message for some input x that is (wrongly) assumed to be a nonce by a principal. ORa models authentication and the invariant shows the genuine misauthentication based on this attack. ORs models the same situation with the assumption that x is of size one; with this assumption the inferred invariant is not leaky.

ARPC models Lowe's modified BAN concrete ARPC protocol [25] (we model succ(-) with pairs, i.e. succ((zero, −)) is (one, −)). We modelled the Kehne-Schönwälder-Landendörfer protocol [25], as modified by Lowe, with (KSLr) and without (KSL) re-authentication.

We produced four models of the Yahalom protocol [9]. Our first model (YAH) does not model secrecy and is used to establish depth-boundedness. The protocol has a type-confusion attack (similar to the case of Otway-Rees) which does not lead to a leak of a secret. This is modelled in YAHs1. We rule out this type-confusion by adding a size assumption in YAHs2. YAHlk is a variant where an additional fresh nonce is exchanged at the beginning of each session, which makes the protocol secret even when old session keys can be leaked, a property entailed by the inferred invariant.

D Example of Incorporation

Consider the limit $L = \nu y.\Big(\big(\nu x.(\mathsf{A}[x] \parallel \mathsf{B}[y,x]^\omega)\big)^\omega\Big)$. To compute $\widehat{\mathrm{post}}$ we consider the limit $L \otimes 1 = \nu y.\Big(\big(\nu x.(\mathsf{A}[x] \parallel \mathsf{B}[y,x] \parallel \mathsf{B}[y,x]^\omega)\big) \parallel \big(\nu x.(\mathsf{A}[x] \parallel \mathsf{B}[y,x]^\omega)\big)^\omega\Big)$ and assume

$$\mathsf{A}[x] \to \nu z.(A[z] \parallel \mathsf{B}[y,z]) \parallel \mathsf{B}[y,x] = P.$$

Then, $\widehat{\mathrm{post}}$ will contain the limit $L' = C[P]$ for $C[\bullet] = \nu y.\Big(\big(\nu x.(\,\bullet \parallel \mathsf{B}[y,x] \parallel \mathsf{B}[y,x]^\omega)\big) \parallel \big(\nu x.(\mathsf{A}[x] \parallel \mathsf{B}[y,x]^\omega)\big)^\omega\Big)$. Since we are trying to prove inclusion between L and $L' = C[P]$, and L is equivalent to $L \otimes 1 = C[\mathsf{A}[x]]$, it seems likely that the inclusion could be proven by matching C with C in the two limits, and by matching P in $L \otimes 1$. Intuitively:

$$\nu y.\Big(\big(\nu x.(\,\textcolor{red}{\mathsf{A}[x]} \parallel \mathsf{B}[y,x] \parallel \underbrace{\mathsf{B}[y,x]^\omega}\,)\big) \parallel \big(\underbrace{\nu x.(\mathsf{A}[x] \parallel \mathsf{B}[y,x]^\omega)}\big)^\omega\Big)$$

$$\nu y.\Big(\big(\nu x.(\,\textcolor{blue}{\overbrace{\nu z.(\mathsf{A}[z] \parallel \mathsf{B}[y,z])} \parallel \overbrace{\mathsf{B}[y,x]}} \parallel \mathsf{B}[y,x] \parallel \mathsf{B}[y,x]^\omega\,)\big) \parallel \big(\nu x.(\mathsf{A}[x] \parallel \mathsf{B}[y,x]^\omega)\big)^\omega\Big)$$

where the part in blue is P and the context C is the part in black. The arrows indicate the sublimits that can be further unfolded to show the inclusion of $C[P]$.

Formally, we want to check that $\forall L' \in \widehat{\mathrm{post}}^s_\Delta(L)\colon [\![L']\!] \subseteq [\![L]\!]$. By construction, $L' = C[P]$ for some context C such that $L \otimes b = C[\mathsf{Q}[\vec{M}]]$. Then, if we can find P in the fixed part of $\mathrm{sf}(L \otimes b)$, conditions **(A)** and **(B)** of Theorem 22 are automatically satisfied because of the shared context C.