

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 

MPI-SWS, Kaiserslautern, Germany
Saarland University, 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 \rightarrow Program verification

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

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 [20]. 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, 34, 10, 4, 25] 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.

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 $\widehat{\text{post}}$ 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. $\widehat{\text{post}}(L) \subseteq L$. 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 [30], 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 $\Gamma_n = \{e(k_1)_{k_2}, e(k_2)_{k_3}, \dots, e(k_{n-1})_{k_n}\}$ 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 [26] and developed in [23, 35, 36]. 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 [21], 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 crypto-

graphic 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, 22] 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 [22], trace equivalence [14]), and restrict protocols using (similar) syntactic conditions, obtaining classes that are orthogonal to ours.

ProVerif [4] and Tamarin [25] 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 Appendix.

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 $\text{ar}(f) \in \mathbb{N}$. The set of messages over Σ is the smallest set \mathbb{M}^Σ 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 $\text{size}(f(M_1, \dots, M_n)) := 1 + \max \{\text{size}(M_i) \mid 1 \leq i \leq n\}$, $\text{size}(a) := 1$, and $\text{names}(a) := \{a\}$, $\text{names}(f(M_1, \dots, M_n)) := \bigcup_{i=1}^n \text{names}(M_i)$. Given $X \subseteq \mathcal{N}$ and $s \in \mathbb{N}$, we define $\mathbb{M}_s^{\Sigma, X} := \{M \in \mathbb{M}^\Sigma \mid \text{names}(M) \subseteq X, \text{size}(M) \leq s\}$. 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 $\theta: \mathcal{N} \rightarrow \mathbb{M}^\Sigma$; 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 $M, N \in \mathbb{M}^\Sigma$, $\Gamma, \Gamma' \in \mathbb{K}^\Sigma$, $a \in \mathcal{N}$:

$$\begin{aligned}
M &\vdash M && \text{(Id)} \\
\Gamma \subseteq \Gamma' \wedge \Gamma' &\vdash M &\implies \Gamma &\vdash M && \text{(Mon)} \\
\Gamma &\vdash M \wedge \Gamma, M &\vdash N &\implies \Gamma &\vdash N && \text{(Cut)} \\
M_1, \dots, M_n &\vdash f(M_1, \dots, M_n) && \text{for every } f \in \Sigma \text{ with } \text{ar}(f) = n && \text{(Constr)} \\
\Gamma\theta &\vdash M\theta \iff \Gamma &\vdash M && \text{for any } \theta \text{ renaming of } \text{names}(\Gamma) && \text{(Alpha)} \\
\Gamma, a &\vdash M \wedge a \notin \text{names}(\Gamma, M) &\implies \Gamma &\vdash M && \text{(Relevancy)}
\end{aligned}$$

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

$$\begin{array}{c}
\frac{M \in \Gamma}{\Gamma \vdash M} \text{ID} \qquad \frac{\Gamma \vdash K}{\Gamma \vdash \mathfrak{p}(K)} \text{PUB} \qquad \frac{\Gamma, (M, N), M, N \vdash M'}{\Gamma, (M, N) \vdash M'} \text{P}_L \qquad \frac{\Gamma \vdash M \quad \Gamma \vdash N}{\Gamma \vdash (M, N)} \text{P}_R \\
\\
\frac{\Gamma, \mathfrak{e}(M)_K \vdash K \quad \Gamma, \mathfrak{e}(M)_K, M, K \vdash N}{\Gamma, \mathfrak{e}(M)_K \vdash N} \text{S}_L \qquad \frac{\Gamma \vdash M \quad \Gamma \vdash K}{\Gamma \vdash \mathfrak{e}(M)_K} \text{S}_R \\
\\
\frac{\Gamma, \mathfrak{a}(M)_{\mathfrak{p}(K)} \vdash K \quad \Gamma, \mathfrak{a}(M)_{\mathfrak{p}(K)}, M, K \vdash N}{\Gamma, \mathfrak{a}(M)_{\mathfrak{p}(K)} \vdash N} \text{A}_L \qquad \frac{\Gamma \vdash M \quad \Gamma \vdash N}{\Gamma \vdash \mathfrak{a}(M)_N} \text{A}_R
\end{array}$$

■ **Figure 1** Deduction rules for the derivability relation of \mathbb{I}_{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 $\Gamma_1, \Gamma_2 \in \mathbb{K}^\Sigma$, $\Gamma_1 \leq_{\text{kn}} \Gamma_2$ if and only if $\forall M \in \Gamma_1: \Gamma_2 \vdash M$. As a consequence, if \vdash is decidable, so is \leq_{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 H. We find it convenient to specify it with a sequent calculus in the style of [33]. 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 $\Sigma_{\text{en}} = \{(\cdot, \cdot), \mathfrak{e}(\cdot), \mathfrak{a}(\cdot), \mathfrak{p}(\cdot)\}$, where (M, N) pairs messages M and N , $\mathfrak{e}(M)_N$ represents the message M encrypted with symmetric key N , $\mathfrak{a}(M)_N$ represents the message M encrypted with asymmetric key N , and $\mathfrak{p}(K)$ is the public key associated with the private key K . The intruder model for (a)symmetric encryption is the model $\mathbb{I}_{\text{en}} = (\Sigma_{\text{en}}, \vdash)$ 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 \mathcal{Q}) each of which has a fixed arity $\text{ar}(\mathcal{Q}) \in \mathbb{N}$. A protocol specification consists of an initial process P and a finite set Δ of (possibly recursive) definitions of the form $\mathcal{Q}[x_1, \dots, x_n] := A$, with $\text{ar}(\mathcal{Q}) = n$, where the syntax of P and A follows the grammar:

$$\begin{array}{l}
P ::= \mathbf{0} \mid \nu x.P \mid P \parallel P \mid \langle M \rangle \mid \mathcal{Q}[\vec{M}] \quad (\text{process}) \\
A ::= \bar{a}\langle M \rangle \mid a(\vec{x} : M).P \mid A + A \quad (\text{action})
\end{array}$$

$$\begin{array}{c}
\frac{\mathbf{Q}_1[\vec{M}_1] \triangleq \bar{c}\langle N[\vec{M}'/\vec{x}] \rangle.P_1 + A_1 \quad \mathbf{Q}_2[\vec{M}_2] \triangleq c(\vec{x} : N).P_2 + A_2}{\mathbf{v}\vec{a}.\langle \Gamma \rangle \parallel \mathbf{Q}_1[\vec{M}_1] \parallel \mathbf{Q}_2[\vec{M}_2] \parallel C \rightarrow_{\Delta} \mathbf{v}\vec{a}.\langle \Gamma \rangle \parallel P_1 \parallel P_2[\vec{M}'/\vec{x}] \parallel C} \text{COMM} \\
\\
\frac{P \equiv P' \rightarrow_{\Delta} Q' \equiv Q}{P \rightarrow_{\Delta} Q} \text{STRUCT} \quad \frac{\mathbf{Q}[\vec{M}] \triangleq \bar{c}\langle M \rangle.P + A \quad \Gamma \vdash c}{\mathbf{v}\vec{a}.\langle \Gamma \rangle \parallel \mathbf{Q}[\vec{M}] \parallel C \rightarrow_{\Delta} \mathbf{v}\vec{a}.\langle \Gamma \rangle \parallel \langle M \rangle \parallel P \parallel C} \text{PUBOUT} \\
\\
\frac{\mathbf{Q}[\vec{M}] \triangleq c(\vec{x} : N).P + A \quad \Gamma, \vec{y} \vdash N[\vec{M}'/\vec{x}] \quad \Gamma \vdash c \quad \vec{y} \text{ fresh}}{\mathbf{v}\vec{a}.\langle \Gamma \rangle \parallel \mathbf{Q}[\vec{M}] \parallel C \rightarrow_{\Delta} \mathbf{v}\vec{a}.\vec{y}.\langle \Gamma \rangle \parallel \langle \vec{y} \rangle \parallel P[\vec{M}'/\vec{x}] \parallel C} \text{PUBIN}
\end{array}$$

■ **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 $\mathbf{Q}[\vec{M}]$ 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 τ , is an abbreviation for $\mathbf{in}(x : x)$, for a fresh x . Processes of the form $\langle M \rangle$ or $\mathbf{Q}[\vec{a}]$ are called *sequential*. The names \vec{x} are bound in both $\mathbf{v}\vec{x}.P$ and $c(\vec{x} : M).P$. We denote the set of free names of a term P with $\text{fn}(P)$ and the set of bound names with $\text{bn}(P)$. As is standard, we require, wlog, that $\text{fn}(P) \cap \text{bn}(P) = \emptyset$. When nesting restrictions $\mathbf{v}\vec{x}.\mathbf{v}\vec{y}.P$, we implicitly assume wlog that \vec{x} and \vec{y} are disjoint. We assume there is at most one definition for each $\mathbf{Q} \in \mathcal{Q}$, and that for each definition $\mathbf{Q}[x_1, \dots, x_n] := A$, $\text{fn}(A) \subseteq \{x_1, \dots, x_n\}$. The set \mathbb{P} consists of all processes over an underlying signature \mathcal{Q} .

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

$$\mathbf{v}\vec{x}.\langle \langle M_1 \rangle \parallel \dots \parallel \langle M_m \rangle \parallel \mathbf{Q}_1[\vec{N}_1] \parallel \dots \parallel \mathbf{Q}_k[\vec{N}_k] \rangle \quad (\text{SF})$$

where every name in \vec{x} occurs free in some subterm. We write $\text{sf}(P)$ for the standard form of P , which is unique up to α -equivalence, and associativity and commutativity of parallel. We abbreviate standard forms with $\mathbf{v}\vec{x}.\langle \Gamma \rangle \parallel Q$ where all the active messages are collected in Γ , and Q is a parallel composition of process calls. Let $\text{sf}(P)$ be the expression (SF), we define $\text{msg}(P) = \{M_1, \dots, M_m\} \cup \bigcup_{i=1}^k \vec{N}_i$. Thus $\text{msg}(P)$ 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 $\mathbf{v}\vec{x}.\langle \Gamma \rangle \parallel Q$ 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 $\mathbf{Q}[\vec{N}]$ represents a single participant in control state \mathbf{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 $\bar{c}\langle M \rangle$ 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.

$$\begin{aligned}
S[a, b, k_{as}, k_{bs}] &:= \mathbf{in}(n_a : (n_a, b)).\nu k. (\langle e(k)_{k_{bs}} \rangle \parallel \langle e(k)_{(n_a, k_{as})} \rangle \parallel S[a, b, k_{as}, k_{bs}]) \\
A_1[a, b, k_{as}] &:= \tau.\nu n_a. (\langle (n_a, b) \rangle \parallel A_2[a, b, k_{as}, n_a] \parallel A_1[a, b, k_{as}]) \\
A_2[a, b, k_{as}, n_a] &:= \mathbf{in}(k : e(k)_{(n_a, k_{as})}). A_3[a, b, k_{as}, k] \\
A_3[a, b, k_{as}, k] &:= \mathbf{in}(n_b : e(n_b)_k). \langle e(n_b)_{(k, k)} \rangle \\
B_1[a, b, k_{bs}] &:= \mathbf{in}(k : e(k)_{k_{bs}}).\nu n_b. (\langle e(n_b)_k \rangle \parallel B_2[a, b, k_{bs}, n_b, k] \parallel B_1[a, b, k_{bs}]) \\
B_2[a, b, k_{bs}, n_b, k] &:= \mathbf{in}(e(n_b)_{(k, k)}).\mathbf{Secret}[k]
\end{aligned}$$

■ **Figure 3** Formal model of Example 9.

We write $Q[\vec{M}] \triangleq A$ if $Q[\vec{x}] := A' \in \Delta$ and $A \triangleq A'[\vec{M}/\vec{x}]$, up to commutativity and associativity of $+$. The transition relation \rightarrow_Δ 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 $\text{fn}(\vec{y}) \not\subseteq \text{fn}(\vec{M}')$, since unused names would simply not contribute to the intruder knowledge. The sets $\text{reach}_\Delta(P) := \{Q \mid P \rightarrow_\Delta^* Q\}$ and $\text{traces}_\Delta(P) := \{Q_0 \cdots Q_n \mid P \equiv_{\text{kn}} Q_0 \rightarrow_\Delta \cdots \rightarrow_\Delta Q_n\}$ 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_{kn}). Knowledge congruence, $P \equiv_{\text{kn}} Q$, is the smallest congruence that includes \equiv and such that $\langle \Gamma_1 \rangle \equiv_{\text{kn}} \langle \Gamma_2 \rangle$ if $\Gamma_1 \sim_{\text{kn}} \Gamma_2$.

Knowledge congruence is also characterised by

$$P_1 \equiv_{\text{kn}} P_2 \iff \text{sf}(P_1) \triangleq \nu \vec{x}. (\langle \Gamma_1 \rangle \parallel Q) \wedge \text{sf}(P_2) \triangleq \nu \vec{x}. (\langle \Gamma_2 \rangle \parallel Q) \wedge \Gamma_1 \sim_{\text{kn}} \Gamma_2.$$

Intuitively, modulo derivability, two processes $P \equiv_{\text{kn}} Q$ are indistinguishable to the intruder and to the principals. Formally, if $P \equiv_{\text{kn}} Q$ then the transitions systems (P, \rightarrow_Δ) and (Q, \rightarrow_Δ) are isomorphic. We thus close the reduction semantics under knowledge congruence: we add the rule that if $P \equiv_{\text{kn}} P' \rightarrow_\Delta Q' \equiv_{\text{kn}} Q$ 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 $P_1 \sqsubseteq_{\text{kn}} P_2$ 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 $\Gamma_1 \leq_{\text{kn}} \Gamma_2$.

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

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

► **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 :

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

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

$$P_0 = \nu a, b, k_{as}, k_{bs}. (S[a, b, k_{as}, k_{bs}] \parallel A_1[a, b, k_{as}] \parallel B_1[a, b, k_{bs}] \parallel \langle a, b \rangle).$$

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 $e(k)_{(n_a, k_{as})}$ 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 $e(n_b)_{(k, k)}$. 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 $\text{Secret}[k]$ after a successful challenge. We always assume the definition $\text{Secret}[k] := \mathbf{in}(k).\text{Leak}[k]$. A transition to $\text{Leak}[k]$ 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 $\text{Leak}[k]$.

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 $B_2[a, b, k_{bs}, n_b, k] := \mathbf{in}(e(n_b)_{(k, k)}).\text{Secret}[k] + \mathbf{in}(e(n_b)_{(k, k)}).\langle k \rangle$ 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 $\mathbf{in}(x, k : e(x)_k)$ 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 $Z = \text{names}(M) \setminus \vec{x}$; the pattern is *implementable*, if, for all $\theta : Z \rightarrow \mathbb{M}$, 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 $\text{nest}_v(Q[\vec{a}]) := \text{nest}_v(\langle M \rangle) := \text{nest}_v(\mathbf{0}) := 0$, $\text{nest}_v(\mathbf{v}x.P) := 1 + \text{nest}_v(P)$, $\text{nest}_v(P \parallel Q) := \max(\text{nest}_v(P), \text{nest}_v(Q))$. The depth of a term is defined as the minimal nesting of restrictions in its knowledge congruence class, $\text{depth}(P) := \min \{ \text{nest}_v(Q) \mid Q \equiv_{\text{kn}} P \}$.*

► **Lemma 12.** *Every Q is α -equivalent to a process Q' such that $|\text{bn}(Q')| \leq \text{nest}_v(Q)$.*

Consider for example $P = \mathbf{v}a, b, c.(\langle a \rangle \parallel \langle e(b)_a \rangle \parallel \langle e(c)_b \rangle \parallel \langle c \rangle)$ which has $\text{nest}_v(P) = 3$. The process P is knowledge-congruent to $Q = (\mathbf{v}a.\langle a \rangle \parallel \mathbf{v}b.\langle b \rangle \parallel \mathbf{v}c.\langle c \rangle)$ which has $\text{nest}_v(Q) = 1$; this gives us $\text{depth}(P) = \text{nest}_v(Q) = 1$. Although $\text{bn}(Q) = \{a, b, c\}$, 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 $|\text{bn}(Q')| \leq \text{nest}_v(Q) \leq \text{depth}(P)$. 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 $\mathbb{S}_s := \{P \in \mathbb{P} \mid \forall M \in \text{msg}(P): \text{size}(M) \leq s\}$ be the set of processes containing messages of size at most s . The set $\mathbb{D}_{s,k}^X$ is the set of processes of depth at most $k \in \mathbb{N}$, with free names in X , and messages not exceeding size s :

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

When starting from some initial process P_0 , every reachable process P has $\text{fn}(P) \subseteq \text{fn}(P_0)$ so X can always be fixed to be $\text{fn}(P_0)$. 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 $\text{reach}_\Delta^s(P) := \{Q \mid P \cdots Q \in \text{traces}_\Delta(P) \cap \mathbb{S}_s^*\}$.

► **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 $\text{reach}_\Delta^s(P) \subseteq \mathbb{D}_{s,k}$, 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 $E[k] := \mathbf{in}(x : x).(\langle \mathbf{e}(x)_k \rangle \parallel E[k])$ leads to unboundedness as soon as the initial process contains $E[k]$ 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 $\nu k.\nu c_1, \dots, c_n.(\langle \mathbf{e}(c_1, c_2)_k \rangle \parallel \langle \mathbf{e}(c_2, c_3)_k \rangle \parallel \dots \langle \mathbf{e}(c_{n-1}, c_n)_k \rangle)$. 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 $x:\text{fresh}$.

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 $\text{reach}_\Delta^s(P)$. For example, k is never leaked to the intruder in executions of the protocol P if the set of processes $\mathcal{S}_k := \{Q \mid \langle k \rangle \not\sqsubseteq_{\text{kn}} Q\}$ —i.e. all the processes where k is not public—is an invariant of P . We will focus here on the class of \sqsubseteq_{kn} -downward-closed invariants. Formally, given a set of processes X , its *\sqsubseteq_{kn} -downward closure* is the set $X\downarrow := \{Q \mid \exists P \in X : Q \sqsubseteq_{\text{kn}} P\}$. A set X is *\sqsubseteq_{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 $\langle k \rangle \not\sqsubseteq_{\text{kn}} Q$ and $Q' \sqsubseteq_{\text{kn}} Q$ implies $\langle k \rangle \not\sqsubseteq_{\text{kn}} Q'$.

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 $X \supseteq \text{reach}_\Delta^s(P)$ which, by downward-closure of X , is equivalent to checking $X \supseteq \text{reach}_\Delta^s(P)\downarrow$. 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 $\text{post}_\Delta^s(X) := \{Q' \mid \exists Q \in X, Q \rightarrow Q' \in \mathbb{S}_s\}$ be the set of processes reachable in one step from processes in X . An invariant X is *inductive* if $X \supseteq \text{post}_\Delta^s(X)$, which is equivalent to requiring $X \supseteq \text{post}_\Delta^s(X)\downarrow$ if X is downward-closed. Any inductive invariant that contains the initial process P will include $\text{reach}_\Delta^s(P)$.

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 $\widehat{\text{post}}_\Delta^s$) to compute, given a finite representation of a downward-closed set D , a finite representation of $\text{post}_\Delta^s(D)\downarrow$.

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 [21, 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 $\text{reach}_\Delta^s(P) \subseteq \mathcal{D}$. 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 $\widehat{\text{post}}_\Delta^s(I) \subseteq I$, and
- (c) $I \subseteq \mathcal{D}$.

If we find such a set I , then we have proven $\text{reach}_\Delta(P) \subseteq \mathcal{D}$, and the overall algorithm can terminate returning “True”. The second procedure, Refuter, enumerates all $Q \in \text{reach}_\Delta(P)$ 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 $\text{reach}_\Delta^s(P)\downarrow$, 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. \blacktriangleleft

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 $Q[\dots]$ for some given Q . Secrecy can be reduced to control-state reachability by introducing a definition $\text{Secret}[m] := \mathbf{in}(m).\text{Leak}[m]$ (for a special process identifier Leak with no definition). In the definition of the protocol one can call $\text{Secret}[m]$ 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 be decided by $P, \Delta \models^s \mathcal{D}_Q$, where \mathcal{D}_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 $\text{Auth}[a, b, n]$ when a believes to share the secret n with b . Absence of misauthentication can be decided by $P, \Delta \models^s \mathcal{A}$ where $\mathcal{A} = \{Q \in \mathbb{D}_{s,k} \mid \forall a, b, c, n. (\text{Auth}[a, b, n] \parallel \text{Auth}[b, c, n]) \not\sqsubseteq_{\text{kn}} Q\}$.

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:

$$\forall n \in \mathbb{N}: \exists Q \in \text{reach}_{\Delta}^s(P): R^n \sqsubseteq_{\text{kn}} Q \quad \text{where } R = \mathbf{vm}.(\langle m \rangle \parallel \langle \mathbf{e}(m)_k \rangle) \quad (\dagger)$$

If (\dagger) 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 (\dagger) is equivalent to $\text{reach}_{\Delta}^s(P) \downarrow \supseteq \{R^n \mid n \in \mathbb{N}\} \downarrow$. 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 (\dagger) does not hold, and P is not susceptible to known-plaintext attacks on k . We can therefore semi-decide (\dagger) 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 $(\mathbb{D}_{s,k}, \sqsubseteq_{kn})$. 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 $(\mathbb{D}_{s,k}, \sqsubseteq_{kn})$ 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 Appendix D.

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 $\text{Idl}(S)$ 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 ${}^\omega$ to represent an arbitrary number of parallel components.

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

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

► **Definition 18** (Denotation of limits). *The denotation of L is the set $\llbracket L \rrbracket := [L]_\downarrow$ where:*

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

We call the processes in $\llbracket L \rrbracket$ instances of L . Define $\text{nest}_v(L)$ to be as nest_v on processes with the addition of the case $\text{nest}_v(L^\omega) := \text{nest}_v(L)$. It is easy to check that for each $P \in \llbracket L \rrbracket$, $\text{depth}(P) \leq \text{nest}_v(L)$. We write $\mathbb{L}_{s,k}^X$ for the set of limit expressions L with free names in X that have $\text{nest}_v(L) \leq k$ 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: $I \in \text{Idl}(\mathbb{D}_{s,k}) \iff \exists L \in \mathbb{L}_{s,k} : I = \llbracket L \rrbracket$.*

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 $D_1 = I_1 \cup \dots \cup I_n$ and $D_2 = J_1 \cup \dots \cup J_m$, 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 $\llbracket L \rrbracket = \llbracket L' \rrbracket$. We can define a standard form for limits: every limit is structurally congruent to a limit of the form $\mathbf{v}\vec{x}.(\langle \Gamma \rangle \parallel \prod_{i \in I} \mathbf{Q}_i[\vec{M}_i] \parallel \prod_{j \in J} B_j^\omega)$ where every name in \vec{x} occurs free at least once in the scope of the restriction, and

$$[L]^n := \begin{cases} L & \text{if } L \text{ is sequential or } \mathbf{0} \\ [L_1]^n \parallel [L_2]^n & \text{if } L = L_1 \parallel L_2 \\ \mathbf{v}x.([L']^n) & \text{if } L = \mathbf{v}x.L' \\ ([B]^n)^\omega & \text{if } L = B^\omega \end{cases} \quad L \otimes n := \begin{cases} L & \text{if } L \text{ is sequential or } \mathbf{0} \\ L_1 \otimes n \parallel L_2 \otimes n & \text{if } L = L_1 \parallel L_2 \\ \mathbf{v}x.(L' \otimes n) & \text{if } L = \mathbf{v}x.L' \\ (B \otimes n)^\omega \parallel B^\omega & \text{if } L = B^\omega \end{cases}$$

■ **Figure 4** The grounding $[-]^n: \mathbb{L}_{s,k} \rightarrow \mathbb{D}_{s,k}$ and extension $-\otimes n: \mathbb{L}_{s,k} \rightarrow \mathbb{L}_{s,k}$ operations on limits.

for all $j \in J$, B_j is also in standard form. When we write $\text{sf}(L) \stackrel{\circ}{=} \mathbf{v}\vec{x}.(\langle \Gamma \parallel Q \parallel R \rangle)$ we imply that Q is a parallel composition of process calls $\prod_{i \in I} \mathbf{Q}_i[\vec{M}_i]$ (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 $[L]^n$ and the n -th extension $L \otimes n$, of a limit L . Grounding replaces each $-\omega$ with $-^n$, with the obvious property that $[L]^n \in \llbracket L \rrbracket$. 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: $\llbracket L \rrbracket = \llbracket L \otimes n \rrbracket$.

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 $\text{names}(\Gamma) \cap \vec{y} = \emptyset$. We say \mathbb{I} is absorbing if, for all messages M with $\text{names}(M) \subseteq \text{names}(\Gamma)$, we have that $\Gamma, \Gamma' \vdash M$ if and only if $\Gamma \vdash M$.

► **Lemma 21.** \mathbb{I}_{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 $L = (\mathbf{v}\vec{x}.(\langle \Gamma \parallel Q \rangle))^\omega$. Imagine comparing the difference in knowledge between $[L]^1$ and $[L]^2$: we have $\text{sf}([L]^2) \stackrel{\circ}{=} (\mathbf{v}\vec{x}.\mathbf{v}\vec{x}'.(\langle \Gamma \parallel \Gamma' \parallel Q \parallel Q' \rangle))$, where $\Gamma' = \Gamma[\vec{x}'/\vec{x}]$. The absorption axiom tells us that if we want to check whether a process $\mathbf{v}\vec{x}.M$ is embedded in $\text{sf}([L]^2)$, 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 $\mathbf{v}\vec{x}.M$ is embedded in $[L]^1$.

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 $P \in \llbracket L \rrbracket$. Take $P = \mathbf{v}a, b.(\langle e(a)_b \rangle \parallel A[b])$ and $L = (\mathbf{v}x.(\langle x \rangle \parallel A[x]))^\omega$. Suppose we replicate the ω twice, obtaining the equivalent limit $L \otimes 2 \equiv \mathbf{v}x_0, x_1.(\langle x_0 \rangle \parallel A[x_0] \parallel \langle x_1 \rangle \parallel A[x_1] \parallel L)$. The idea is that we can match P against the fixed part of $L \otimes 2$:

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

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 $P \in \llbracket L \rrbracket$. 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 $P \notin \llbracket L \rrbracket$. In other words, we need to know that after some threshold n , there is no point trying with bigger extensions. Take for example $P = \mathbf{v}x, y.(\langle B[x, y] \parallel B[y, x] \rangle)$ and $L = (\mathbf{v}x, y. \langle B[x, y] \rangle)^\omega$. We can try and embed P into $L \otimes 2$ but we would fail as the fixed part expands to $\mathbf{v}x_0, y_0. \langle B[x_0, y_0] \parallel \mathbf{v}x_1, y_1. \langle B[x_1, y_1] \rangle$.

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 $\text{sf}(L_1) \stackrel{\alpha}{=} \mathbf{v}\vec{x}_1.\langle\Gamma_1\rangle \parallel Q_1 \parallel \prod_{i \in I} B_i^\omega$, and let $n = |\vec{x}_1| + |Q_1| + 1$. Then:*

$$\llbracket L_1 \rrbracket \subseteq \llbracket L_2 \rrbracket \iff \begin{cases} \text{sf}(L_2 \otimes n) \stackrel{\alpha}{=} \mathbf{v}\vec{x}_1, \vec{x}_2.\langle\Gamma_2\rangle \parallel Q_1 \parallel Q_2 \parallel R_2 & \text{and } \Gamma_1 \leq_{\text{kn}} \Gamma_2 & \text{(A)} \\ \llbracket \langle\Gamma_1\rangle \parallel \prod_{i \in I} B_i \rrbracket \subseteq \llbracket \langle\Gamma_2\rangle \parallel R_2 \rrbracket & & \text{(B)} \end{cases}$$

► **Theorem 23.** *Given $L_1, L_2 \in \mathbb{L}$ it is decidable whether $\llbracket L_1 \rrbracket \subseteq \llbracket L_2 \rrbracket$.*

Proof. Theorem 22 leads to a recursive algorithm. Given L_1 and L_2 , one computes $\text{sf}(L_1)$ and $\text{sf}(L_2 \otimes n)$. 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:

$$\begin{aligned} L_1 &= \mathbf{v}x_1.\langle\langle\mathbf{v}x_2.\langle\langle e(x_2)_{x_1} \rangle \parallel A[x_2] \parallel A[x_1] \rangle\rangle^\omega \rangle \\ L_2 &= \mathbf{v}y_1, y_3.\langle\langle y_3 \rangle \parallel A[y_3]^\omega \parallel (\mathbf{v}y_2.\langle\langle y_2 \rangle \parallel A[y_2] \rangle)^\omega \rangle \end{aligned}$$

We prove that $\llbracket L_1 \rrbracket \subseteq \llbracket L_2 \rrbracket$ 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

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

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 $\langle e(x_2)_{y_1} \rangle$ from the knowledge on the right-hand side:

$$\llbracket \mathbf{v}x_2.\langle\langle e(x_2)_{y_1} \rangle \parallel A[x_2] \parallel A[x_1] \rangle \rrbracket \not\subseteq \llbracket \langle y_3 \rangle \parallel \langle e(y_{22})_{y_3} \rangle \parallel A[x_1]^\omega \parallel (\mathbf{v}y_2.\langle\langle e(y_2)_{y_3} \rangle \parallel A[y_2] \rangle)^\omega \rrbracket$$

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

$$\llbracket \mathbf{v}x_2.\langle\langle e(x_2)_{y_3} \rangle \parallel A[x_2] \parallel A[x_1] \rangle \rrbracket \subseteq \llbracket \langle y_3 \rangle \parallel \langle e(y_{22})_{y_3} \rangle \parallel A[x_1]^\omega \parallel (\mathbf{v}y_2.\langle\langle e(y_2)_{y_3} \rangle \parallel A[y_2] \rangle)^\omega \rrbracket$$

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

3.5 Computing Post-Hat

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

$$\begin{aligned}
L &= \nu a, b, k_{as}, k_{bs}. (\langle a, b \rangle \parallel A_1[a, b, k_{as}]^\omega \parallel B_1[a, b, k_{bs}]^\omega \parallel S[a, b, k_{as}, k_{bs}]^\omega \parallel L_1^\omega) \\
L_1 &= \nu n_a. (\langle n_a \rangle \parallel A_2[a, b, k_{as}, n_a] \parallel L_2^\omega) \\
L_2 &= \nu k. (\langle e(k)_{(a, k_{as})} \rangle \parallel \langle e(k)_{(b, k_{as})} \rangle \parallel \langle e(k)_{(n_a, k_{as})} \rangle \parallel \langle e(k)_{k_{bs}} \rangle \parallel \text{Secret}[k]^\omega \parallel A_3[a, b, k_{as}, k]^\omega \parallel L_3^\omega) \\
L_3 &= \nu n_b. (\langle e(n_b)_{(k, k)} \rangle \parallel \langle e(n_b)_k \rangle \parallel B_2[a, b, k_{bs}, n_b, k])
\end{aligned}$$

■ **Figure 5** An inductive invariant for Example 9.

enumerate $\llbracket L \rrbracket$. 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 $\text{sf}(L \otimes b) = \nu \vec{x}. (\langle \Gamma \rangle \parallel Q \parallel R)$,

$$\widehat{\text{post}}_\Delta^s(L) := \{ \nu \vec{y}. (\langle \Gamma' \rangle \parallel Q' \parallel R) \mid \nu \vec{x}. (\langle \Gamma \rangle \parallel Q) \rightarrow_\Delta \nu \vec{y}. (\langle \Gamma' \rangle \parallel Q') \in \mathbb{S}_s \}.$$

► **Theorem 26.** $\widehat{\text{post}}_\Delta^s(L) = \{L_1, \dots, L_n\} \implies \text{post}_\Delta^s(\llbracket L \rrbracket) \downarrow = \llbracket L_1 \rrbracket \cup \dots \cup \llbracket L_n \rrbracket$.

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 $\widehat{\text{post}}_\Delta^3(L)$ is included in L . Since the initial process of Example 9, P_0 , is in $\llbracket L \rrbracket$, the invariant certifies 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 $\llbracket \nu k. (\nu x. (\langle x, e(x)_k \rangle))^\omega \rrbracket \not\subseteq \llbracket L^\omega \rrbracket$, 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 [36]; 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 $P_1 \rightarrow^* P_2$ with $P_1 \sqsubseteq_{\text{kn}} P_2$, one can deduce that the same sequence can be simulated from P_2 (by Theorem 8) obtaining a P_3 with $P_2 \sqsubseteq_{\text{kn}} P_3$ and so on. The embedding between P_1 and P_2 , is justified by $P_1 \equiv_{\text{kn}} \nu \vec{x}. (\langle \Gamma_1 \rangle \parallel Q)$, $P_2 \equiv_{\text{kn}} \nu \vec{x}. (\langle \Gamma_1 \rangle \parallel Q \parallel P)$; 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: **Inference** of invariant fully automatic (F) or interactive (I); **Check** of inductiveness; **Secrecy** 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 $\llbracket \widehat{\text{post}}_{\Delta}^s(L) \rrbracket \subseteq \llbracket L \rrbracket$ implemented by checking that, for each transition considered by $\widehat{\text{post}}$ 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 $C[\mathbb{Q}[\vec{M}]]$ and L' to $C[P]$ for some P . To prove inclusion of $C[P]$ in $C[\mathbb{Q}[\vec{M}]]$ it is then sufficient to show that P is embedded in $C[\mathbb{Q}[\vec{M}]]$. We call this check an *incorporation* of P in C . See Appendix K 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 $P \rightarrow P' \in \llbracket L_2 \rrbracket \setminus \llbracket L_1 \rrbracket$. 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 $\llbracket - \rrbracket$. 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 [19] where we also provide a tutorial-style explanation of the methodology. We summarise our experiments³ 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 I, 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, $E[k] \parallel (\nu x. \langle e(x)_k \rangle)^\omega$ 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 $E[k] \parallel (\nu x. \langle e(x^\star)_k \rangle)^\omega$ 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 [29, §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_{kn} is NP-complete for any intruder model that has polynomially decidable \leq_{kn} [32].

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, Ștefan Ciobăcă, 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. Lemma9, version 1.0, 2020. doi:10.5281/zenodo.3950846.

- 20 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.
- 21 Alain Finkel and Jean Goubault-Larrecq. Forward analysis for WSTS, part I: completions. In *STACS*, volume 3 of *LIPICs*, pages 433–444, 2009.
- 22 Sibylle B. Fröschle. Leakiness is decidable for well-founded protocols. In *POST'15*, pages 176–195, 2015.
- 23 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.
- 24 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.
- 25 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.
- 26 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.
- 27 Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, 1978.
- 28 David J. Otway and Owen Rees. Efficient and timely mutual authentication. *Operating Systems Review*, 21(1):8–10, 1987.
- 29 Lawrence C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1-2):85–128, 1998.
- 30 Mark D. Ryan and Ben Smyth. Applied pi-calculus. In *Formal Models and Techniques for Analyzing Security Protocols*. 2011.
- 31 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.
- 32 Felix Stutz. Decidable inductive invariants for verification of cryptographic protocols with unbounded sessions. Master’s thesis, Saarland University, 2019.
- 33 Alwen Tiu, Rajeev Goré, and Jeremy E. Dawson. A proof theoretic analysis of intruder theories. *Logical Methods in Computer Science*, 6(3), 2010.
- 34 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.
- 35 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.
- 36 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.

Appendix

A Proof of Proposition 4

We check the model Σ_{en} is an effective intruder model. Rule ID implies (Id); it is easy to check that any derivation for $\Gamma \vdash M$ is also a derivation (of same depth) for $\Gamma, \Gamma' \vdash M$, which implies (Mon); (Cut) is a consequence of the admissibility of the CUT rule proven in [33]. The \neg_{R} rules directly entail (Constr). All rules are invariant under renamings so (Alpha) holds. (Relevancy) can be proven by a straightforward induction on the depth of derivations. Since the premises in every rule involve only sub-terms of the consequence, derivability is decidable.

B Proof of Proposition 7

We want to prove $P_1 \equiv_{\text{kn}} P_2$ if and only if $P_1 \sqsubseteq_{\text{kn}} P_2$ and $P_2 \sqsubseteq_{\text{kn}} P_1$. The “only if” direction is trivial. The “if” direction is proved as follows. Let $\text{sf}(P_i) \triangleq \nu \vec{x}_i. (\langle \Gamma_i \rangle \parallel Q_i)$ for $i = 1, 2$. It is easy to check that $P_1 \sqsubseteq_{\text{kn}} P_2$ if and only if $\Gamma_1 \leq_{\text{kn}} \Gamma_2$, $\vec{x}_2 = \vec{x}_1 \vec{y}_2$ for some \vec{y}_2 , and $Q_2 = Q_1 \parallel Q'_2$ for some parallel composition of process calls Q'_2 . Similarly, $P_2 \sqsubseteq_{\text{kn}} P_1$ if and only if $\Gamma_2 \leq_{\text{kn}} \Gamma_1$, $\vec{x}_1 = \vec{x}_2 \vec{y}_1$, and $Q_1 = Q_2 \parallel Q'_1$ for some \vec{y}_1, Q'_1 . We get $\Gamma_1 \sim_{\text{kn}} \Gamma_2$, $\vec{x}_1 = \vec{x}_2$ and $Q_1 = Q_2$ as required to prove $P_1 \equiv_{\text{kn}} P_2$.

C Auxiliary Results

C.1 Properties of Knowledge

► **Lemma 27.** *Let $\Gamma_1, \Gamma_2, \Gamma \in \mathbb{K}$ be such that $\Gamma_1 \leq_{\text{kn}} \Gamma_2$. Then, $\Gamma, \Gamma_1 \leq_{\text{kn}} \Gamma, \Gamma_2$.*

Proof. Let $\Gamma_1 = \{M_1, \dots, M_n\}$. We have to show that for all N , if $\Gamma, \Gamma_1 \vdash N$ then $\Gamma, \Gamma_2 \vdash N$. We apply (Cut) n times obtaining

$$\frac{\Gamma, \Gamma_2, (\Gamma_1 \setminus M_n) \vdash M_n \quad \Gamma, \Gamma_1, \Gamma_2 \vdash N \quad \text{CUT}}{\dots \quad \vdots \quad \text{CUT}} \quad \text{CUT}$$

$$\frac{\Gamma, \Gamma_2, M_1 \vdash M_2 \quad \Gamma, \Gamma_2, M_1, M_2 \vdash N \quad \text{CUT}}{\Gamma, \Gamma_2, M_1 \vdash N \quad \text{CUT}} \quad \text{CUT}$$

$$\frac{\Gamma, \Gamma_2 \vdash M_1 \quad \Gamma, \Gamma_2, M_1 \vdash N \quad \text{CUT}}{\Gamma, \Gamma_2 \vdash N \quad \text{CUT}}$$

From $\Gamma_1 \leq_{\text{kn}} \Gamma_2$ we have $\forall i \leq n: \Gamma_2 \vdash M_i$, which implies, by (Mon), all the left-most premises. By (Mon), from $\Gamma, \Gamma_1 \vdash N$ we know $\Gamma, \Gamma_1, \Gamma_2 \vdash N$, which completes the derivation showing $\Gamma, \Gamma_2 \vdash N$. ◀

► **Corollary 28.** *Let $\Delta_1, \Delta_2, \Gamma_1, \Gamma_2 \in \mathbb{K}$ be such that $\Delta_1 \leq_{\text{kn}} \Delta_2$ and $\Gamma_1 \leq_{\text{kn}} \Gamma_2$. Then, $\Delta_1, \Gamma_1 \leq_{\text{kn}} \Delta_2, \Gamma_2$.*

Proof. Easy corollary of Lemma 27: $\Gamma_1, \Delta_1 \leq_{\text{kn}} \Gamma_2, \Delta_1 \leq_{\text{kn}} \Gamma_2, \Delta_2$. ◀

► **Lemma 29.** $\langle \Gamma \rangle \parallel \langle \Gamma \rangle \equiv_{\text{kn}} \langle \Gamma \rangle$, for all $\Gamma \in \mathbb{K}$.

Proof. Let $\Gamma \in \mathbb{K}$. By definition, $\langle \Gamma \rangle \parallel \langle \Gamma \rangle \equiv_{\text{kn}} \langle \Gamma \rangle$ iff $\Gamma, \Gamma \sim_{\text{kn}} \Gamma$. Notice that these are actually sets, hence $\Gamma, \Gamma = \Gamma$ and the claim follows by reflexivity of \sim_{kn} . ◀

C.2 Properties of Knowledge Embedding

► **Corollary 30.** *Let $\Gamma \in \mathbb{K}$ and $P_1, P_2 \in \mathbb{P}$ with $\text{sf}(P_i) = \text{v}\vec{x}_i.(\langle \Gamma_i \rangle \parallel Q_i)$ for $i \in \{1, 2\}$ and $P_1 \sqsubseteq_{\text{kn}} P_2$. Then, $\langle \Gamma \rangle \parallel P_1 \sqsubseteq_{\text{kn}} \langle \Gamma \rangle \parallel P_2$.*

Proof. We only have to compare knowledge as $Q_1 \sqsubseteq_{\text{kn}} Q_2$ holds by assumption. $\Gamma, \Gamma_1 \leq_{\text{kn}} \Gamma, \Gamma_2$ follows by Lemma 27. ◀

► **Lemma 31.** *Let $P_1, P_2 \in \mathbb{P}$ and $n \in \mathbb{N}$. If $P_1 \sqsubseteq_{\text{kn}} P_2$, then $P_1^n \sqsubseteq_{\text{kn}} P_2^n$.*

Proof. For every single instance of both processes use the matching provided by $P_1 \sqsubseteq_{\text{kn}} P_2$ for names and process calls. The knowledge embedding part of the goal can be obtained by n applications of Corollary 28. ◀

C.3 Properties of Grounding

► **Lemma 32.** *For every $n \in \mathbb{N}$, $\lceil L \rceil^n \in \llbracket L \rrbracket$.*

► **Lemma 33.** *For every $n \leq m \in \mathbb{N}$, $\lceil L \rceil^n \sqsubseteq_{\text{kn}} \lceil L \rceil^m$.*

► **Lemma 34.** *For every $P \in \llbracket L \rrbracket$ there exists an $n \in \mathbb{N}$ such that $P \sqsubseteq_{\text{kn}} \lceil L \rceil^n$.*

Proof of Lemmas 32 to 34. All claims follow by an easy induction on the structure of L . ◀

► **Lemma 35.** *For any $L \in \mathbb{L}$, and $m, n \in \mathbb{N}$, we have $(\lceil L^\omega \rceil^n)^m \sqsubseteq_{\text{kn}} \lceil L^\omega \rceil^{m*n}$.*

Proof. We assume that $n > 0$ as the claim trivially holds if not.

$$\begin{aligned} (\lceil L^\omega \rceil^n)^m &= ((\lceil L \rceil^n)^n)^m = (\lceil L \rceil^n)^{m*n} \\ \lceil L^\omega \rceil^{m*n} &= (\lceil L \rceil^{m*n})^{m*n} \end{aligned}$$

By Lemma 33, we know that $\lceil L \rceil^m \sqsubseteq_{\text{kn}} \lceil L \rceil^{m*n}$ as $n > 0$. The claim follows by Lemma 31. ◀

► **Lemma 36.** $\llbracket L_1 \rrbracket \subseteq \llbracket L_2 \rrbracket \iff \forall n \in \mathbb{N}: \exists m \in \mathbb{N}: \lceil L_1 \rceil^n \sqsubseteq_{\text{kn}} \lceil L_2 \rceil^m$.

Proof. First, let $n \in \mathbb{N}$. We compute the expansion $\lceil L_1 \rceil^n \in \llbracket L_1 \rrbracket$ by Lemma 32. By assumption, $\lceil L_1 \rceil^n \in \llbracket L_2 \rrbracket$ and by Lemma 34 there exists an m such that $\lceil L_1 \rceil^n \sqsubseteq_{\text{kn}} \lceil L_2 \rceil^m$ as required. Second, let $P \in \llbracket L_1 \rrbracket$. We need to show that $P \in \llbracket L_2 \rrbracket$. By Lemma 34, there exists an expansion for some n with $P \sqsubseteq_{\text{kn}} \lceil L_1 \rceil^n$. By assumption, there is an m such that $\lceil L_1 \rceil^n \sqsubseteq_{\text{kn}} \lceil L_2 \rceil^m$. By transitivity of \sqsubseteq_{kn} , we know that $P \sqsubseteq_{\text{kn}} \lceil L_2 \rceil^m$ which is in $\llbracket L_2 \rrbracket$. By downward closure, the claim follows. ◀

C.4 Properties of Extension

► **Lemma 37.** *For every $L \in \mathbb{L}$ and $n \in \mathbb{N}$, $\llbracket L \rrbracket = \llbracket L \otimes n \rrbracket$.*

Proof. We prove the fact by two inclusions. The inclusion $\llbracket L \rrbracket \subseteq \llbracket L \otimes n \rrbracket$ follows trivially by definition. For the reverse inclusion, we apply Lemma 36. Given an arbitrary n' , we need to show that there is m' such that $\llbracket L \otimes n \rrbracket^{n'} \sqsubseteq_{\text{kn}} \llbracket L \rrbracket^{m'}$. It is straightforward to verify that $m' = n \cdot n'$ suffices. ◀

► **Lemma 38.** *For every $L \in \mathbb{L}$ and $n \in \mathbb{N}$, $\lceil L \rceil^n = \lceil L \otimes n \rceil^0$.*

Proof. Straightforward induction on the structure of L . ◀

D Soundness and Completeness of Limits

D.1 Depth-Bounded Processes are Well-Quasi-Ordered

Multisets Given a set X , the multisets over X , are all the functions $\mu: X \rightarrow \mathbb{N}$ such that the set $\text{items}(\mu) := \{(x, i) \in X \times \mathbb{N} \mid 0 < i \leq \mu(x)\}$ is finite. We denote by $\mathcal{M}(X)$ the set of all multisets over X . We write $\{\!\{x_1, \dots, x_n\}\!\}$ for the multiset with elements x_1, \dots, x_n and use \emptyset for the empty multiset. The union two multisets $\mu_1 \oplus \mu_2$ is the pointwise lifting of addition. Assume (X, \sqsubseteq) is a quasi order; its *multiset extension* is the quasi order $\sqsubseteq_{\mathcal{M}}$ over $\mathcal{M}(X)$ defined as follows; for two multisets $\mu_1, \mu_2 \in \mathcal{M}(X)$, $\mu_1 \sqsubseteq_{\mathcal{M}} \mu_2$ holds just if there is an injective function $f: \text{items}(\mu_1) \rightarrow \text{items}(\mu_2)$ such that for each $(x, i) \in \text{items}(\mu_1)$, if $f(x, i) = (y, j)$ then $x \sqsubseteq y$. When X is a set ordered by equality, $\sqsubseteq_{\mathcal{M}}$ coincides with multiset inclusion. If (X, \sqsubseteq) is a wqo then $(\mathcal{M}(X), \sqsubseteq_{\mathcal{M}})$ is a wqo [21].

Forests We define a domain of forests $\mathbb{F}_{s,k}^X$ with sequential processes in $\mathcal{B}_s(X)$ as leaves:

$$\begin{aligned} \mathcal{B}_s(X) &:= \{Q[\vec{M}] \mid Q \in \mathcal{Q}, \text{ar}(Q) = |\vec{M}|, \vec{M} \subseteq \mathbb{M}_s^X\} \cup \mathbb{M}_s^X \\ \mathbb{F}_{s,0}^X &:= \mathcal{M}(\mathcal{B}_s(X)) \quad \mathbb{F}_{s,k+1}^X := \mathcal{M}\left(\mathcal{B}_s(X) \uplus \mathbb{F}_{s,k}^{X \cup \{x_{k+1}\}}\right) \quad (\text{assuming } x_{k+1} \notin X) \end{aligned}$$

The set $\mathcal{B}_s(X)$ contains all sequential processes of $\mathbb{D}_{s,k}^X$. In the base case, forests of height 0, $\mathbb{F}_{s,0}^X$ is simply a multiset of sequential processes. Forests of height $k+1$ are represented in $\mathbb{F}_{s,k+1}^X$ by a multiset of sequential processes and of subforests of height k .

For any given $s \in \mathbb{N}$ and finite $X \subseteq \mathcal{N}$, $\mathcal{B}_s(X)$ is a finite set and thus forms a wqo with equality. With $\sqsubseteq_{\mathbb{F}}$ (forest embedding) we denote the qo over $\mathbb{F}_{s,k}^X$ defined as follows: on $\mathbb{F}_{s,0}^X$ it coincides with the multiset extension of equality on $\mathcal{B}_s(X)$; on $\mathbb{F}_{s,k+1}^X$ it coincides with the multiset extension of the disjoint union of equality on $\mathcal{B}_s(X)$, and the forest embedding on $\mathbb{F}_{s,k}^{X \cup \{x_{k+1}\}}$. By iterating the result on multisets, we obtain that for any $s, k \in \mathbb{N}$ and finite $X \subseteq \mathcal{N}$, $(\mathbb{F}_{s,k}^X, \sqsubseteq_{\mathbb{F}})$ is a wqo.

The function $\mathcal{F}[\cdot]_k$, transforms a process P with $\text{nest}_v(P) \leq k$ into its forest encoding, that is a forest that faithfully represents the structure of P .

► **Definition 39** (Forest encoding). *We define the function $\mathcal{F}[\cdot]_k$ which transforms a process P with $\text{nest}_v(P) \leq k$ into a forest:*

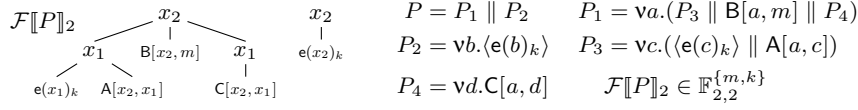
$$\mathcal{F}[P]_k := \begin{cases} \emptyset & \text{if } P = \mathbf{0} \\ \{P\} & \text{if } P \text{ is sequential} \\ \{\mathcal{F}[Q[x_k/x]]_{k-1}\} & \text{if } P = vx.Q \\ \mathcal{F}[Q_1]_k \oplus \mathcal{F}[Q_2]_k & \text{if } P = Q_1 \parallel Q_2 \end{cases}$$

assuming $\{x_1, \dots, x_k\} \cap (\text{bn}(P) \cup \text{fn}(P)) = \emptyset$; when this assumption is not met because of bound names, we implicitly α -rename the term before applying the definition. When $\text{nest}_v(P) > k$, $\mathcal{F}[P]_k$ is undefined.

Figure 6 illustrates forest encodings. The forests in $\mathbb{F}_{s,k}^Y$ have sequential processes as leaves (with messages of size smaller than s), and free names $Y \uplus \{x_1, \dots, x_k\}$. Lemma 12 allows us to use only k distinct bound names: this makes the forests finitely labelled.

► **Lemma 40.** *If $Q \in \mathbb{S}_s^Y$ and $\text{nest}_v(Q) \leq k$, then $\mathcal{F}[Q]_k \in \mathbb{F}_{s,k}^Y$.*

Proof. By easy induction on the structure of Q . ◀



■ **Figure 6** Forest encoding of P . Restrictions at nesting level k are renamed canonically to x_{2-k} .

► **Lemma 41.** *Assume $Q_1, Q_2 \in \mathbb{S}_s^Y$ with $\text{nest}_v(Q_1) \leq k$ and $\text{nest}_v(Q_2) \leq k$. Then $\mathcal{F}[[Q_1]]_k \sqsubseteq_{\mathbb{F}} \mathcal{F}[[Q_2]]_k$ implies $Q_1 \sqsubseteq_{\text{kn}} Q_2$.*

Proof. First we strengthen the statement: we can prove that if $\mathcal{F}[[Q_1]]_k \sqsubseteq_{\mathbb{F}} \mathcal{F}[[Q_2]]_k$ then $\text{sf}(Q_1) = \mathbf{v}\vec{y}.R$ and $\text{sf}(Q_2) = \mathbf{v}\vec{y}.\mathbf{v}\vec{z}.(R \parallel R')$, which clearly implies $Q_1 \sqsubseteq_{\text{kn}} Q_2$.

We proceed by induction on k . The base case is a special case of the induction step, so let us consider the latter first. Assume $\text{nest}_v(Q_1), \text{nest}_v(Q_2) \leq k+1$ and $\varphi_1 = \mathcal{F}[[Q_1]]_{k+1} \sqsubseteq_{\mathbb{F}} \mathcal{F}[[Q_2]]_{k+1} = \varphi_2$, and let $B = \text{items}(\varphi_1) \cap (\mathcal{B}_Y(s) \times \mathbb{N})$ and $C = \text{items}(\varphi_1) \setminus B$ (note that $C \subseteq (\mathbb{F}_{s,k}^{Y \cup \{x_{k+1}\}} \times \mathbb{N})$). Then, by definition, there is an injective function $f: \text{items}(\varphi_1) \rightarrow \text{items}(\varphi_2)$ such that for each $(P, i) \in B$, $f(P, i) = (P, j)$ for some j ; moreover, for each $(\varphi, i) \in C$, if $f(\varphi, i) = (\varphi', j)$ then $\varphi \sqsubseteq_{\mathbb{F}} \varphi'$. By definition of $\mathcal{F}[-]$ we know that for each $(\varphi, i) \in C$, there exist a subterm of Q_1 α -equivalent to $\mathbf{v}x_{k+1}.P_\varphi$ for some P_φ with $\text{nest}_v(P_\varphi) \leq \text{nest}_v(Q_1) - 1 \leq k$ and $\varphi = \mathcal{F}[[P_\varphi]]_k$. Similarly for Q_2 , we have that if $f(\varphi, i) = (\varphi', j)$ then there is a subterm of Q_2 α -equivalent to $\mathbf{v}x_{k+1}.P_{f(\varphi, i)}$ for some $P_{f(\varphi, i)}$ with $\mathcal{F}[[P_{f(\varphi, i)}]]_k = \varphi'$ and $\text{nest}_v(P_{f(\varphi, i)}) \leq \text{nest}_v(Q_2) - 1 \leq k$.

We can therefore apply the induction hypothesis and get $\varphi \sqsubseteq_{\mathbb{F}} \varphi'$ implies that $\text{sf}(P_\varphi) = \mathbf{v}\vec{y}_\varphi.R_\varphi$ and $\text{sf}(P_{f(\varphi, i)}) = \mathbf{v}\vec{y}_\varphi.\mathbf{v}\vec{z}_\varphi.(R_\varphi \parallel R_{f(\varphi, i)})$. As a consequence we have

$$\begin{aligned} Q_1 &\equiv_{\text{kn}} \left(\prod_{(P, i) \in B} P \parallel \prod_{(\varphi, i) \in C} (\mathbf{v}x_{k+1}.\mathbf{v}\vec{y}_\varphi.R_\varphi) \right) \\ Q_2 &\equiv_{\text{kn}} \left(\prod_{(P, i) \in B} P \parallel \prod_{(\varphi, i) \in C} (\mathbf{v}x_{k+1}.\mathbf{v}\vec{y}_\varphi.\mathbf{v}\vec{z}_\varphi.(R_\varphi \parallel R_{f(\varphi, i)})) \parallel R' \right) \end{aligned}$$

which clearly entails the claim, by application of α -renaming and scope extrusion to get the two standard forms. In the base case, $C = \emptyset$ from which the claim follows straightforwardly. ◀

► **Theorem 42.** *For every finite $Y \subseteq \mathcal{N}$ and $s, k \in \mathbb{N}$, $(\mathbb{D}_{s,k}^Y, \sqsubseteq_{\text{kn}})$ is a wqo.*

Proof. Let P_1, P_2, \dots be an infinite sequence of processes in $\mathbb{D}_{s,k}^Y$. By definition, for all $i \in \mathbb{N}$ there exists a process $Q_i \in \mathbb{S}_s$ such that $Q_i \equiv_{\text{kn}} P_i$ and $\text{nest}_v(Q_i) \leq k$, which implies by Lemma 40 that $\mathcal{F}[[Q_i]]_k \in \mathbb{F}_{s,k}^Y$. The sequence $\mathcal{F}[[Q_1]]_k, \mathcal{F}[[Q_2]]_k, \dots$ must then be a good sequence since $(\mathbb{F}_{s,k}^Y, \sqsubseteq_{\mathbb{F}})$ is a wqo. Therefore, there are $i, j \in \mathbb{N}$ with $i < j$, such that $\mathcal{F}[[Q_i]]_k \sqsubseteq_{\mathbb{F}} \mathcal{F}[[Q_j]]_k$, which by Lemma 41 implies $Q_i \sqsubseteq_{\text{kn}} Q_j$. We therefore have $P_i \sqsubseteq_{\text{kn}} P_j$ which proves that the sequence P_1, P_2, \dots is good. ◀

D.2 Soundness and Completeness (Theorem 19)

We prove the two directions of Theorem 19 independently.

► **Theorem 43.** *For every $L \in \mathbb{L}_{s,k}$, the set $[[L]]$ is an ideal of $(\mathbb{D}_{s,k}, \sqsubseteq_{\text{kn}})$.*

Proof. We want to prove that for every $L \in \mathbb{L}_{s,k}$, we have $[[L]] \subseteq \mathbb{D}_{s,k}$ and $[[L]]$ is \sqsubseteq_{kn} -downward closed and directed.

Clearly $[[L]] \subseteq \mathbb{D}_{s,k}$ since every $P \in [[L]]$ has $\text{nest}_v(P) \leq \text{nest}_v(L) \leq k$ and its active messages are the messages of L up to renaming (which does not alter size).

$[[L]]$ is downward closed by definition. Proving that $[[L]]$ is directed requires showing that for all $P_1, P_2 \in [[L]]$, there is $Q \in [[L]]$ such that $P_1 \sqsubseteq_{\text{kn}} Q$ and $P_2 \sqsubseteq_{\text{kn}} Q$. Such Q can be

constructed by applying Lemma 34 to P_1 and P_2 , obtaining n_1 and n_2 such that $P_i \sqsubseteq_{\text{kn}} [L]^{n_i}$, for $i = 1, 2$. Then $Q = [L]^{\max(n_1, n_2)}$ is the required process by Lemma 33. \blacktriangleleft

► **Theorem 44.** *Every ideal in $\text{Idl}(\mathbb{D}_{s,k})$ is the denotation of some limit $L \in \mathbb{L}_{s,k}$.*

Proof. The idea of the proof is to use the forest encoding $\mathcal{F}[-]$ to apply known facts about the ideal completion for multisets to our setting. We take an ideal of $\mathbb{D}_{s,k}$ and show it corresponds to a downward closed set of $\mathbb{F}_{s,k}$, via forest encoding. For multisets we know exactly how to represent ideals [21], i.e. \otimes -products. These products are expressions that can be readily seen as limits. Let us recall the result on multisets first and then proceed with the proof.

Let X be a wqo with a finite representation of ideals $\mathbb{I}(X)$. Then we can finitely represent ideals of $\mathcal{M}(X)$ by expressions of the form

$$p = I_1^{\otimes} \odot \cdots \odot I_n^{\otimes} \odot J_1^? \odot \cdots \odot J_m^?$$

where $I_i, J_j \in \mathbb{I}(X)$ for all $1 \leq i \leq n$, $1 \leq j \leq m$. When $n, m = 0$ we write $p = \epsilon$. Such expressions are called \otimes -products. Their denotations are the ideals of $\mathcal{M}(X)$, via the map

$$\begin{aligned} \llbracket \epsilon \rrbracket &:= \{\emptyset\} & \llbracket p_1 \odot p_2 \rrbracket &:= \{\mu_1 \oplus \mu_2 \mid \mu_1 \in \llbracket p_1 \rrbracket, \mu_2 \in \llbracket p_2 \rrbracket\} \\ \llbracket I^{\otimes} \rrbracket &:= \mathcal{M}(\llbracket I \rrbracket) & \llbracket I^? \rrbracket &:= \{\{x\} \mid x \in \llbracket I \rrbracket\} \cup \{\emptyset\} \end{aligned}$$

In a wqo, downward closed sets are finite unions of ideals, so any downward closed set of $\mathcal{M}(X)$ can be described as the union of the denotation of finitely many \otimes -products. For the details⁴ see [21].

Another well-known property of ideals we are going to use is the following. Let D be a downward closed set, then the following are equivalent:

- (1) D is an ideal;
- (2) for all downward closed sets $D_1, D_2 \subseteq X$,
if $D \subseteq D_1 \cup D_2$ then $D \subseteq D_1$ or $D \subseteq D_2$;
- (3) for all downward closed sets $D_1, D_2 \subseteq X$,
if $D = D_1 \cup D_2$ then $D = D_1$ or $D = D_2$.

Now we turn to our theorem. Let $D \subseteq \mathbb{D}_{s,k}$, we define

$$\mathcal{F}[D]_k := \{\mathcal{F}[P] \mid P \in D, \text{nest}_v(P) \leq k\}.$$

By definition, $\mathcal{F}[D]_k \subseteq \mathbb{F}_{s,k}$.

We also define the opposite mapping, from forests to processes in the expected way. Let $\varphi \in \mathbb{F}_{s,k}^X$, $B(\varphi) := \text{supp}(\varphi) \cap \mathcal{B}_s(X)$ and $C(\varphi) := \text{supp}(\varphi) \setminus B(\varphi)$. Then

$$\mathcal{P}[\varphi]_k := \left(\prod_{P \in B(\varphi)} P^{\varphi(P)} \parallel \prod_{\psi \in C(\varphi)} (\mathbf{v}x_k \cdot \mathcal{P}[\psi]_{k-1})^{\varphi(\psi)} \right)$$

Since when $k = 0$, $C(\varphi) = \emptyset$, the expression is well-defined. We have $\mathcal{P}[\varphi]_k \subseteq \mathbb{D}_{s,k}$ and⁵ $\mathcal{F}[\mathcal{P}[\varphi]_k]_k = \varphi$ and $\mathcal{P}[\mathcal{F}[Q]_k]_k \equiv Q$ (when defined).

► **Lemma 45.** *For any \sqsubseteq_{kn} -downward closed $D \subseteq \mathbb{D}_{s,k}$, $\mathcal{F}[D]_k$ is $\sqsubseteq_{\mathbb{F}}$ -downward closed.*

⁴ here we use a variation of \otimes -products that can be seen equivalent to the one in [21] by using $(C_1 \cup C_2)^{\otimes} = C_1^{\otimes} \odot C_2^{\otimes}$.

⁵ Note the implicit α -renaming in the application of $\mathcal{F}[-]$.

Proof. First note that for each $P \in D$ there is a $Q \equiv_{\text{kn}} P$ in D such that $\mathcal{F}[[Q]] \in \mathcal{F}[[D]]_k$.

Towards a contradiction, assume there is a forest $\varphi \in \mathbb{F}_{s,k}$ such that there is a $\psi \in \mathcal{F}[[D]]_k$ with $\varphi \sqsubseteq_{\mathbb{F}} \psi$ but $\varphi \notin \mathcal{F}[[D]]_k$. We know that $\psi = \mathcal{F}[[P]]_k$ for some $P \in D$. From Lemma 41 we can derive that $\mathcal{P}[[\varphi]]_k \sqsubseteq_{\text{kn}} P$. But since D is downward closed, $\mathcal{P}[[\varphi]]_k \in D$, which contradicts the assumption that $\varphi \notin \mathcal{F}[[D]]_k$. \blacktriangleleft

Proof of Theorem 44 We prove, by induction on k , that $\text{Idl}(\mathbb{D}_{s,k}^X) \subseteq \llbracket \mathbb{L}_{s,k} \rrbracket$. Again, the base case is a special case of the induction step. Let $k > 0$ and $D \in \text{Idl}(\mathbb{D}_{s,k}^X)$; By induction hypothesis, we can assume that the ideals of $\mathbb{D}_{s,k-1}^X$ are described by limits in $\mathbb{L}_{s,k-1}^X$. Since D is downward closed, by Lemma 45, $\mathcal{F}[[D]]_k \subseteq \mathbb{F}_{s,k}$ is downward closed as well, and thus is the finite union of some ideals of $\mathbb{F}_{s,k}^X$. Now we have,

$$\text{Idl}(\mathbb{F}_{s,k}^X) = \text{Idl}\left(\mathcal{M}\left(\mathcal{B}_s(X) \uplus \mathbb{F}_{s,k}^{X \cup \{x_{k+1}\}}\right)\right)$$

so we can represent $\text{Idl}(\mathbb{F}_{s,k}^X)$ with \otimes -products over $\mathcal{B}_s(X) \uplus \mathbb{L}_{s,k}^{X \cup \{x_{k+1}\}}$. Let p_1, \dots, p_n be such that $\bigcup_{i=1}^n \llbracket p_i \rrbracket = \mathcal{F}[[D]]_k$. From each p_i we can obtain a limit L_{p_i} by replacing:

1. \odot with \parallel ;
2. each $B^?$ with B and each B^\otimes with B^ω , if $B \in \mathcal{B}_s(X)$;
3. each $L^?$ with $\text{v}x_k.L$ and each L^\otimes with $(\text{v}x_k.L)^\omega$, if $L \in \mathbb{L}_{s,k}^{X \cup \{x_{k+1}\}}$.

We now show that $D = \bigcup_{i=1}^n \llbracket L_{p_i} \rrbracket$. It is important to remember that $\llbracket p_i \rrbracket$ are $\sqsubseteq_{\mathbb{F}}$ -ideals while $\llbracket L_{p_i} \rrbracket$ are \sqsubseteq_{kn} -ideals. One can show, by induction on the structure of p_i , that $Q \in \llbracket L_{p_i} \rrbracket$ if and only if $Q \equiv Q'$ for some $Q' \in \mathcal{P}[[p_i]]$. It then follows that $\llbracket L_{p_i} \rrbracket = \{Q \mid \exists \varphi \in \llbracket p_i \rrbracket : Q \sqsubseteq_{\text{kn}} \mathcal{P}[[\varphi]]\}$ and therefore

$$\begin{aligned} D &= \{Q \mid \exists \varphi \in \mathcal{F}[[D]] : Q \sqsubseteq_{\text{kn}} \mathcal{P}[[\varphi]]\} \\ &= \bigcup_{i=1}^n \{Q \mid \exists \varphi \in \llbracket p_i \rrbracket : Q \sqsubseteq_{\text{kn}} \mathcal{P}[[\varphi]]\} = \bigcup_{i=1}^n \llbracket L_{p_i} \rrbracket \end{aligned}$$

We thus established that $D = \bigcup_{i=1}^n \llbracket L_{p_i} \rrbracket$. Since D is an ideal and each $\llbracket L_{p_i} \rrbracket$ is downward closed, we have that $D = \llbracket L_{p_j} \rrbracket$ for some $1 \leq j \leq n$. \blacktriangleleft

E Proof of Absorption for \mathbb{I}_{en} (Lemma 21)

Let \vec{x} and \vec{y} be two lists of pairwise distinct names, $\Gamma \subseteq \mathbb{M}^{\Sigma_{\text{en}}}$ be a finite set of messages, and $\Gamma' = \Gamma[\vec{y}/\vec{x}]$. Moreover, assume that $\text{names}(\Gamma) \cap \vec{y} = \emptyset$. We have to show that, for all messages M with $\text{names}(M) \subseteq \text{names}(\Gamma)$, we have that $\Gamma, \Gamma' \vdash M$ if and only if $\Gamma \vdash M$.

The \Leftarrow direction is a direct consequence of (Mon). For the other direction, let $\sigma = [\vec{y}/\vec{x}]$ and $\sigma' = [\vec{x}/\vec{y}]$, so $N\sigma' \in \Gamma$ if and only if $N \in \Gamma'$. Note that $\Gamma\sigma' = \Gamma'\sigma' = \Gamma$. Moreover, if $\text{names}(M) \subseteq \text{names}(\Gamma)$, $M = M\sigma'$ because, by assumption, $\text{names}(\Gamma) \cap \vec{y} = \emptyset$.

We proceed by induction on the depth of the derivation for $\Gamma, \Gamma' \vdash M$. We assume the statement holds for derivations of depth d and do a case analysis on the last rule applied in the $(d+1)$ -deep derivation for $\Gamma, \Gamma' \vdash M$.

Rule Id: $M \in \Gamma \cup \Gamma'$. If $M \in \Gamma$ we are done. Otherwise, if $M \in \Gamma'$, we have $M\sigma' \in \Gamma$ but $M = M\sigma'$ so $M \in \Gamma$.

Rule PL: $(N_1, N_2) \in \Gamma \cup \Gamma'$ and $\frac{\Gamma, \Gamma', N_1, N_2 \vdash M}{\Gamma, \Gamma' \vdash M}$.

We then have two cases:

Case $(N_1, N_2) \in \Gamma$. By (Mon) we can transform the d -deep derivation for $\Gamma, \Gamma', N_1, N_2 \vdash M$ into a derivation for $\Gamma, N_1, N_2, \Gamma', N_1\sigma, N_2\sigma \vdash M$ of the same depth. Since $M \subseteq \text{names}(\Gamma, N_1, N_2) = \text{names}(\Gamma)$ and by definition $\Gamma', N_1\sigma, N_2\sigma = (\Gamma, N_1, N_2)\sigma$, we can apply the induction hypothesis, obtaining $\Gamma, N_1, N_2 \vdash M$ which proves $\Gamma \vdash M$ by application of Rule P_L.

Case $(N_1, N_2) \in \Gamma'$. From $\Gamma, \Gamma', N_1, N_2 \vdash M$ we get $(\Gamma, \Gamma', N_1, N_2)\sigma' \vdash M\sigma'$ by (Alpha). Since we have $\Gamma\sigma' = \Gamma'\sigma' = \Gamma$ and $M\sigma' = M$, we can infer that $\Gamma, N_1\sigma', N_2\sigma' \vdash M$ which proves, together with $(N_1\sigma', N_2\sigma') \in \Gamma$, that $\Gamma \vdash M$ by application of Rule P_L.

Rule S_L: $e(N)_K \in \Gamma \cup \Gamma'$ and $\frac{\Gamma, \Gamma' \vdash K \quad \Gamma, \Gamma', N, K \vdash M}{\Gamma, \Gamma' \vdash M}$.

We distinguish two cases:

Case $e(N)_K \in \Gamma$. Then $\text{names}(K) \subseteq \text{names}(\Gamma)$ and by induction hypothesis we have $\Gamma \vdash K$. We also have $\text{names}(N) \subseteq \text{names}(\Gamma)$, by definition $(\Gamma, N, K)\sigma = \Gamma', N\sigma, K\sigma$ and $\text{names}(M) \subseteq \text{names}(\Gamma, N, K)$. We can transform the d -deep derivation for $\Gamma, \Gamma', N, K \vdash M$ into a derivation for $\Gamma, N, K, \Gamma', N\sigma, K\sigma \vdash M$ with the same depth, so by induction hypothesis we obtain $\Gamma, N, K \vdash M$. By applying Rule S_L to $\Gamma \vdash K$ and $\Gamma, N, K \vdash M$ we obtain $\Gamma \vdash M$.

Case $e(N)_K \in \Gamma'$. Then by (Alpha), $\Gamma, \Gamma' \vdash K$ implies $\Gamma\sigma', \Gamma'\sigma' \vdash K\sigma'$. Since $\Gamma\sigma' = \Gamma$ and $\Gamma'\sigma' = \Gamma$ we have $\Gamma \vdash K\sigma'$. Similarly, $\Gamma, \Gamma', N, K \vdash M$ implies $\Gamma, N\sigma', K\sigma' \vdash M\sigma'$. Since $M\sigma' = M$, we have $\Gamma, N\sigma', K\sigma' \vdash M$, which in conjunction with $\Gamma \vdash K\sigma'$ and $e(N\sigma')_{K\sigma'} \in \Gamma$, proves $\Gamma \vdash M$ by application of Rule S_L.

Rule A_L: Analogous to the case of Rule S_L.

Rule P_R: $M = (N_1, N_2)$ and $\frac{\Gamma, \Gamma' \vdash N_1 \quad \Gamma, \Gamma' \vdash N_2}{\Gamma, \Gamma' \vdash M}$.

We have $\text{names}(N_1), \text{names}(N_2) \subseteq \text{names}(\Gamma)$ so we can apply the induction hypothesis to the two premises and get $\Gamma \vdash N_1$ and $\Gamma \vdash N_2$, which proves $\Gamma \vdash M$ by Rule P_R.

Rule S_R: Analogous to the case of Rule P_R.

Rule A_R: Analogous to the case of Rule P_R. ◀

F Proof of Characterisation of Limits Inclusion

For any limit L with standard form $v\vec{x}.(\langle \Gamma \rangle \parallel Q \parallel \prod_{i \in I} B_i^\omega)$, we call Γ and Q non-iterated, also fixed, components while B_i for $i \in I$ are iterated components.

We recall the statement: Let L_1 and L_2 be two limits, with $\text{sf}(L_1) \stackrel{\alpha}{=} v\vec{x}_1.(\langle \Gamma_1 \rangle \parallel Q_1 \parallel \prod_{i \in I} B_i^\omega)$, and let $n = |\vec{x}_1| + |Q_1| + 1$. Then:

$$\llbracket L_1 \rrbracket \subseteq \llbracket L_2 \rrbracket \iff \begin{cases} \text{sf}(L_2 \otimes n) \stackrel{\alpha}{=} v\vec{x}_1, \vec{x}_2.(\langle \Gamma_2 \rangle \parallel Q_1 \parallel Q_2 \parallel R_2) \text{ and } \Gamma_1 \leq_{\text{kn}} \Gamma_2 & \text{(A)} \\ \llbracket \langle \Gamma_1 \rangle \parallel \prod_{i \in I} B_i \rrbracket \subseteq \llbracket \langle \Gamma_2 \rangle \parallel R_2 \rrbracket & \text{(B)} \end{cases}$$

F.1 Conditions (A) and (B) Are Sound for Inclusion

► **Lemma 46** (Soundness). *If conditions (A) and (B) hold, then $\llbracket L_1 \rrbracket \subseteq \llbracket L_2 \rrbracket$.*

Proof. Given that (A) and (B) hold, it suffices to show that

$$\forall n_1 \in \mathbb{N}, \exists n_2 \in \mathbb{N}. \llbracket L_1 \rrbracket^{n_1} \subseteq_{\text{kn}} \llbracket L_2 \otimes n \rrbracket^{n_2}$$

by Lemmas 36 and 37. Let n_1 be an arbitrary number. With $R_1 := \prod_{i \in I} B_i^\omega$, we have $\llbracket \text{sf}(L_1) \rrbracket^{n_1} = v\vec{x}_1.(\langle \Gamma_1 \rangle \parallel Q_1 \parallel \llbracket R_1 \rrbracket^{n_1})$ and $\text{sf}(\llbracket L_2 \otimes n \rrbracket^0) \stackrel{\alpha}{=} v\vec{x}_1, \vec{x}_2.(\langle \Gamma_2 \rangle \parallel Q_1 \parallel Q_2)$ by (A).

We show that $\mathbf{v}\vec{x}_1.\langle\Gamma_1\rangle \parallel Q_1 \parallel [R_1]^{n_1} \sqsubseteq_{\text{kn}} \mathbf{v}\vec{x}_1, \vec{x}_2.\langle\Gamma_2\rangle \parallel Q_1 \parallel Q_2 \parallel [R_2]^{n_2}$ for some n_2 . Condition (A) induces a knowledge order for both fixed parts. Condition (B) gives rise to some relation between the iterated components we will exploit. This relation is captured in the following fact. For some α -renaming in (A) and some n_2 :

$$\langle\Gamma_1\rangle \parallel [R_1]^{n_1} \sqsubseteq_{\text{kn}} \langle\Gamma_2\rangle \parallel [R_2]^{n_2} \quad (*)$$

Prior to proving (*), we show that this fact and (A) suffice to prove our goal. With $\text{sf}([R_i]^{n_i}) = \mathbf{v}\vec{y}_i.\langle\Gamma_{R_i}^{n_i} \parallel Q'_i\rangle$, we call $\Gamma_{R_i}^{n_i}$ the knowledge of $[R_i]^{n_i}$ for $i \in \{1, 2\}$. Because of (*), we can choose \vec{y}_2 so that $\vec{y}_2 = \vec{y}_1, \vec{z}_2$ and there is no need to rename for the embedding anymore. Equipped with this abbreviation, the knowledge of the left hand side ($\mathbf{v}\vec{x}_1.\langle\Gamma_1\rangle \parallel Q_1 \parallel [R_1]^{n_1}$) is $(\Gamma_1, \Gamma_{R_1}^{n_1})$ while $\Gamma_2, \Gamma_{R_2}^{n_2}$ is the knowledge of the right hand side $\mathbf{v}\vec{x}_1, \vec{x}_2.\langle\Gamma_2\rangle \parallel Q_1 \parallel Q_2 \parallel [R_2]^{n_2}$. This is also exactly the knowledge contained in $(\langle\Gamma_1\rangle \parallel [R_1]^{n_1})$ respectively $(\langle\Gamma_2\rangle \parallel [R_2]^{n_2})$, hence $\Gamma_1, \Gamma_{R_1}^{n_1} \leq_{\text{kn}} \Gamma_2, \Gamma_{R_2}^{n_2}$ by (*). It remains to take care of names and process calls for the embedding. Condition (A) already takes care of non-iterated names and process calls. With our assumption (*), we obtain

$$\mathbf{v}\vec{x}_1.\langle\Gamma_1\rangle \parallel Q_1 \parallel \langle\Gamma_1\rangle \parallel [R_1]^{n_1} \sqsubseteq_{\text{kn}} \mathbf{v}\vec{x}_1, \vec{x}_2.\langle\Gamma_2\rangle \parallel Q_1 \parallel Q_2 \parallel \langle\Gamma_2\rangle \parallel [R_2]^{n_2}.$$

Both parts are knowledge congruent to our goal's sides by Lemma 29 which proves the goal.

It remains to show that (*), i.e. $(\langle\Gamma_1\rangle \parallel [R_1]^{n_1}) \sqsubseteq_{\text{kn}} (\langle\Gamma_2\rangle \parallel [R_2]^{n_2})$ for some n_2 . We start with (B) and apply Lemma 36 so that we know that for every $n_1 \in \mathbb{N}$, there is a $m \in \mathbb{N}$ such that $\langle\Gamma_1\rangle \parallel [\prod_{i \in I} B_1]^{n_1} \sqsubseteq_{\text{kn}} \langle\Gamma_2\rangle \parallel [R_2]^m$. By Lemma 31, we multiply on both sides:

$$(\langle\Gamma_1\rangle \parallel [\prod_{i \in I} B_1]^{n_1})^{n_1} \sqsubseteq_{\text{kn}} (\langle\Gamma_2\rangle \parallel [R_2]^m)^{n_1}.$$

By Lemma 29, we pull the messages out of both replications:

$$\langle\Gamma_1\rangle \parallel ([\prod_{i \in I} B_1]^{n_1})^{n_1} \sqsubseteq_{\text{kn}} \langle\Gamma_2\rangle \parallel ([R_2]^m)^{n_1}.$$

We continue on both sides individually. On the left, we omit messages for simplicity as we already know that $\Gamma_1 \leq_{\text{kn}} \Gamma_2$.

$$([\prod_{i \in I} B_1]^{n_1})^{n_1} = (\prod_{i \in I} [B_1]^{n_1})^{n_1} = [R_1]^{n_1}.$$

While we use Lemma 35 and Corollary 30 on the right in order to obtain that:

$$\langle\Gamma_2\rangle \parallel ([R_2]^m)^{n_1} \sqsubseteq_{\text{kn}} \langle\Gamma_2\rangle \parallel [R_2]^{n_1 \cdot m}$$

Combining both paths leads to: $\langle\Gamma_1\rangle \parallel [R_1]^{n_1} \sqsubseteq_{\text{kn}} \langle\Gamma_2\rangle \parallel [R_2]^{n_1 \cdot m}$. By choosing $n_2 = n_1 \cdot m$, the claim follows. \blacktriangleleft

F.2 Conditions (A) and (B) Are Complete for Inclusion

► **Lemma 47** (Completeness). *If $\llbracket L_1 \rrbracket \subseteq \llbracket L_2 \rrbracket$, then conditions (A) and (B) hold.*

For this proof we are assuming the intruder model is absorbing. Before we give the proof, we need some auxiliary definitions and results.

► **Corollary 48.** *Let \vec{x} and \vec{y} be two lists of pairwise distinct names, Γ, Γ_1 be two finite sets of messages, and $\Gamma_2 = \Gamma_1[\vec{y}/\vec{x}]$. Moreover, assume that $\text{names}(\Gamma_1) \cap \vec{y} = \emptyset$ and $\text{names}(\Gamma) \cap \vec{y} = \emptyset = \text{names}(\Gamma) \cap \vec{x}$. Then, for all messages M with $\text{names}(M) \subseteq \text{names}(\Gamma, \Gamma_1)$, we have that $\Gamma, \Gamma_1, \Gamma_2 \vdash M$ if and only if $\Gamma, \Gamma_1 \vdash M$.*

Proof. The direction from right to left is obvious. For the reverse direction, it is equivalent to show that $\Gamma, \Gamma_1, \Gamma, \Gamma_2 \vdash M$. Now, $\Gamma, \Gamma_1 = (\Gamma, \Gamma_2)[\vec{y}/\vec{x}]$. The claim follows by the assumption that the intruder is absorbing (Definition 20). ◀

► **Lemma 49.** *Let L, L' be two limits such that $L \equiv L'$. Then for every $n \in \mathbb{N}$: $[L]^n \equiv [L']^n$.*

Proof. The proof is a straightforward structural induction on L . ◀

We introduce a refinement of grounding and a function folding the right hand side but preserving the knowledge embedding. The (n, k, m) -th grounding takes a limit and unfolds each ω n times for the outer k nested levels of ω , and m times for the inner ones.

► **Definition 50** (Step-indexed grounding). *For a limit L in standard form, we define the (n, k, m) -th grounding of L to be the process $[L]^{n,k,m}$ recursively:*

$$[L]^{n,k,m} := \begin{cases} L & \text{if } L \text{ is sequential or } \mathbf{0} \\ [L_1]^{n,k,m} \parallel [L_2]^{n,k,m} & \text{if } L = L_1 \parallel L_2 \\ \mathbf{v}x.([L']^{n,k,m}) & \text{if } L = \mathbf{v}x.L' \\ ([B]^{n,k-1,m})^n & \text{if } L = B^\omega \wedge k > 0 \\ ([B]^m)^m & \text{if } L = B^\omega \wedge k = 0 \end{cases}$$

► **Definition 51** (ω -height). *For a limit L , we define the ω -height as follows:*

$$\omega\text{-height}(L) := \begin{cases} 0 & \text{if } L \text{ is sequential or } \mathbf{0} \\ \max(\omega\text{-height}(R_1), \dots, \omega\text{-height}(R_n)) & \text{if } L = R_1 \parallel \dots \parallel R_n \\ \omega\text{-height}(L') & \text{if } L = \mathbf{v}x.L' \\ \omega\text{-height}(B) + 1 & \text{if } L = B^\omega \end{cases}$$

► **Lemma 52.** *Let L be a limit and $m, n, k \in \mathbb{N}$. If $k \geq \omega\text{-height}(L)$, then $[L]^{n,k,m} = [L]^n$.*

Proof. The parameter k only decreases when recursing into a limit under ω . If $k \geq \omega\text{-height}(L)$ the last case of the definition will never apply, which makes the definition coincide with the one of n -grounding. ◀

The idea of the following parametrised function is to fold an m -grounding to an n -grounding up to a certain ω -height k of the limit. Since we want to be very specific about the domains, we define some sets of groundings.

► **Definition 53** (Set of groundings). *Let $L \in \mathbb{L}$ and $m, n, k \in \mathbb{N}$.*

We define the following two sets of processes:

- $\mathcal{R}_L^m := \{[\text{sf}(L)]^m\}$
- $\mathcal{R}_L^{n,k,m} := \{[\text{sf}(L)]^{n,k,m}\}$

► **Definition 54.** *Let L be a limit in recursive standard form and $m, k, n \in \mathbb{N}$. The function $\Phi_{k,L}^{n,m} : \mathcal{R}_L^m \rightarrow \mathcal{R}_L^{n,k,m}$ and is parametrised in all four variables.*

$$\Phi_{k,P}^{n,m} := \begin{cases} P & \text{if } k = 0 \\ \mathbf{v}\vec{x}.(\langle \Gamma \rangle \parallel Q \parallel \prod_{j \in J} (\Phi_{k,L_j}^{n,m}([\mathcal{L}_j^\omega]^m))) & \text{if } k > 0 \wedge P = \mathbf{v}\vec{x}.(\langle \Gamma \rangle \parallel Q \parallel \prod_{j \in J} \mathcal{L}_j^\omega) \\ (\Phi_{k-1,L}^{n,m}([\mathcal{L}]^m))^n & \text{if } k > 0 \wedge P = [\mathcal{L}^\omega]^m \end{cases}$$

We may omit the parameters n and m in the following if obvious from context as they do not change over the process of folding.

► **Lemma 55** (Folding is sound wrt. knowledge embedding). *Let L_1, L_2 be two limits in standard form $L_i = \mathbf{v}\vec{x}_i.(\langle \Gamma_i \rangle \parallel Q_i \parallel R_i)$ with $n = |\vec{x}_1| + |Q_1| + 1$. If there is a $m \in \mathbb{N}$ such that $m > n$ and $\lceil L_1 \rceil^0 \sqsubseteq_{\text{kn}} \lceil L_2 \rceil^m$, then $\forall k : \lceil L_1 \rceil^0 \sqsubseteq_{\text{kn}} \Phi_{k, L_2}^{n, m}(\lceil L_2 \rceil^m)$.*

Proof. We prove the claim by induction on k .

For $k = 0$, the claim trivially follows as $\Phi_{0, L_2}^{n, m}(\lceil L_2 \rceil^m) = \lceil L_2 \rceil^m$ by the assumption that $\lceil L_1 \rceil^0 \sqsubseteq_{\text{kn}} \lceil L_2 \rceil^m$ holds.

For the induction step, we assume that $\lceil L_1 \rceil^0 \sqsubseteq_{\text{kn}} \Phi_{k, L_2}^{n, m}(\lceil L_2 \rceil^m)$ and prove

$$\lceil L_1 \rceil^0 \sqsubseteq_{\text{kn}} \Phi_{k+1, L_2}^{n, m}(\lceil L_2 \rceil^m).$$

By definition of folding, both $\Phi_{k, L}(\lceil L_2 \rceil^m)$ and $\Phi_{k+1, L}(\lceil L_2 \rceil^m)$ are folding in exactly the same way up to the k -th recursive calls, i.e. calls in which k decreases. This means that up to the calls to $\Phi_{0, L'}$ and $\Phi_{1, L'}$, both $\Phi_{k, L}(\lceil L_2 \rceil^m)$ and $\Phi_{k+1, L}(\lceil L_2 \rceil^m)$ will have constructed the same context $C[-, \dots, -]$ around these final calls. We thus characterise $A = \Phi_{k, L_2}^{n, m}(\lceil L_2 \rceil^m)$, and $B = \Phi_{k+1, L_2}^{n, m}(\lceil L_2 \rceil^m)$ as follows:

$$\begin{aligned} A &= C[\Phi_{0, F_1^\omega}^{n, m}(F_1^\omega), \dots, \Phi_{0, F_j^\omega}^{n, m}(F_j^\omega)] \\ &= C[\lceil F_1 \rceil^m, \dots, \lceil F_j \rceil^m] = C[(\lceil F_1 \rceil^m)^m, \dots, (\lceil F_j \rceil^m)^m] \\ B &= C[\Phi_{1, F_1^\omega}^{n, m}(F_1^\omega), \dots, \Phi_{1, F_j^\omega}^{n, m}(F_j^\omega)] = C[(\lceil F_1 \rceil^m)^n, \dots, (\lceil F_j \rceil^m)^n] \end{aligned}$$

Recall that $L_1 = \mathbf{v}\vec{x}_1.(\langle \Gamma_1 \rangle \parallel Q_1 \parallel R_1)$ and hence $\lceil L_1 \rceil^0 = \mathbf{v}\vec{x}_1.(\langle \Gamma_1 \rangle \parallel Q_1)$. By assumption, we have

$$\mathbf{v}\vec{x}_1.(\langle \Gamma_1 \rangle \parallel Q_1) \sqsubseteq_{\text{kn}} \Phi_{k, L_2}^{n, m}(\lceil L_2 \rceil^m) = A \quad (1)$$

and want to prove that

$$\mathbf{v}\vec{x}_1.(\langle \Gamma_1 \rangle \parallel Q_1) \sqsubseteq_{\text{kn}} \Phi_{k+1, L_2}^{n, m}(\lceil L_2 \rceil^m) = B. \quad (2)$$

Intuitively, we have to find a way to preserve the knowledge embedding from (1) when removing some branches in the holes of the context to get from A to B . Let us show what A and B look like explicitly with their context:

$$\begin{aligned} A &\equiv \mathbf{v}\vec{c}.(\langle \Gamma_c \rangle \parallel Q_c \parallel D) && \text{with } D = (\lceil F_1 \rceil^m)^m \parallel \dots \parallel (\lceil F_j \rceil^m)^m \\ B &\equiv \mathbf{v}\vec{c}.(\langle \Gamma_c \rangle \parallel Q_c \parallel E) && \text{with } E = (\lceil F_1 \rceil^m)^n \parallel \dots \parallel (\lceil F_j \rceil^m)^n \end{aligned}$$

We call Γ_c the knowledge of the context.

Our goal is to show that reducing the number of iterations in each of D 's parallel components does not affect the knowledge embedding described in (i) and thereby obtain (ii). Let us first consider names and process calls. We can split $\lceil L_1 \rceil^0$ in the following way: $\lceil L_1 \rceil^0 = \mathbf{v}\vec{x}_1.(\langle \Gamma_1 \rangle \parallel Q_1) = \mathbf{v}\vec{y}. \mathbf{v}\vec{z}.(\langle \Gamma_1 \rangle \parallel Q_y \parallel Q_z)$ where $\vec{y} \subseteq \vec{c}$ and $Q_y \subseteq Q_c$. The intention is to distinguish names and process calls that are already matched in the context. This is why we do not require all process calls that are only using names from \vec{y} to be in Q_y but some of them might be in Q_z . The goal follows with the following claim immediately.

Claim I: In every hole of context $C[-, \dots, -]$, we can reduce the number of branches to at most n , i.e. for every $1 \leq l \leq j$ we can have a grounding $\lceil F_l \rceil^n$ instead of $\lceil F_j \rceil^m$.

Proof of Claim I. Towards a contradiction, assume that there is a hole in which we cannot remove $m - n$ branches. W.l.o.g. let $[L_l]^m$ for $1 \leq l \leq j$ be the sublimit in this hole. There might be three reasons for this: names, process calls and knowledge.

Considering the names and process calls, we know that $|x_1| + |Q_1| < n$ by definition and hence $|\vec{z}| + |Q_z| < n$. Now, we investigate the components that \vec{z} and Q_z are matched to. In the worst case, all of them are mapped to this hole but we still delete $m - n$ branches that are not used to match the names \vec{z} . For the process calls in Q_z , we have to distinguish two cases. First, if a process call uses any name from \vec{z} , it is fine as we will leave them anyway. Second, if a process call does not use any name from \vec{z} , it is fine to delete this branch as there will be enough copies in the remaining branches to cover this process call.

It remains to reason about knowledge. We make the knowledge of D , i.e. the one having budget k , explicit: $\text{sf}(D) = \mathbf{v}\vec{a}.(\Gamma_m \parallel \dots)$ and factor out the knowledge from sublimit $[F_l]^m$: $\Gamma_m = \Gamma'_m, \Gamma_{l,m}$ so that $\Gamma_{l,m}$ was the knowledge obtained through $[F_l]^m$. For knowledge, we have to prove that $\Gamma_1 \leq_{\text{kn}} \Gamma_c, \Gamma'_m, \Gamma_{l,n}$. By Proposition 2, we need to prove that for every message $M \in \Gamma_1, \Gamma_c, \Gamma'_m, \Gamma_{l,n} \vdash M$ given that $\Gamma_c, \Gamma'_m, \Gamma_{l,m} \vdash M$. The idea now is to reduce the knowledge from $\Gamma_{l,m}$ to $\Gamma_{l,n}$ by Corollary 48, which is a corollary of the absorbing intruder. Let us define $\Gamma'_{c,m} = \Gamma_c, \Gamma'_m$ indicating the context of the hole we are considering. As m might be bigger than $2n$, we have to iterate the process of reducing the number of branches. Hence, we generalise the notation of $\Gamma_{l,m}$ and $\Gamma_{l,n}$ in the following way: $([F_l]^m)^i \equiv \mathbf{v}\vec{a}_i.(\langle \Gamma_{l,i} \parallel \dots \rangle)$.

Claim II: $\forall m > n, \Gamma_{l,m} \vdash M \implies \Gamma_{l,m-1} \vdash M$.

Proof of Claim II. For convenience, we rename $\Gamma_{l,i}$ to Λ_i . The main observation is that we can split the knowledge Λ_m into Λ_{m-1} and a remainder Λ' . We can choose a branch which does not use names from \vec{z} to contribute to Λ' . Since we know that $n \geq 1$, we know that $m > 1$ by assumption. Therefore, we can split Λ_{m-1} again and obtain the knowledge stemming from one branch which we call Λ'' . Let us recall the assumption and goal after these rewriting steps: Given that

$$\Gamma'_{c,m}, \Lambda_{m-2}, \Lambda', \Lambda'' \vdash M \quad (3)$$

holds, we want to prove that $\Gamma'_{c,m}, \Lambda_{m-2}, \Lambda'' \vdash M$. Let \vec{w}' and \vec{w}'' be the names only used in Λ' and Λ'' respectively so that:

$$\vec{w}' \cap \vec{w}'' = \emptyset \text{ and } \text{names}(\Gamma'_{c,m}, \Lambda_{m-2}) \cap \vec{w}' = \emptyset = \text{names}(\Gamma'_{c,m}, \Lambda_{m-2}) \cap \vec{w}'' \quad (4)$$

Λ' and Λ'' have been obtained from a branch of the same sublimit, so we can infer that

$$\Lambda'' = \Lambda'[\vec{w}''/\vec{w}']. \quad (5)$$

Notice that $\vec{w}' \cap \vec{z} = \emptyset$ by the fact how we have chosen the branch for Λ' . Furthermore, $\vec{w}' \cap \vec{y} = \emptyset$ by $\vec{y} \subseteq \vec{c}$. Combining these observations, we get that $x_1 \cap \vec{w}' = \emptyset$. By the fact that $M \in \Gamma_1$, we have $\text{names}(M) \subseteq \text{names}(\Gamma_1) \subseteq x_1$. Therefore, $\text{names}(M) \cap \vec{w}' = \emptyset$ which implies that

$$\text{names}(M) \subseteq \text{names}(\Gamma'_{c,m}, \Lambda_{m-2}, \Lambda'') \quad (6)$$

Facts (3) to (6) fulfil the conditions for Corollary 48, resulting in $\Gamma'_{c,m}, \Lambda_{m-2}, \Lambda'' \vdash M$ which reads $\Gamma'_{c,m}, \Lambda_{m-1} \vdash M$ when folding back which is the goal of Claim II. By this, we have shown that we can remove $m - n$ branches which leads to a contradiction which is why Claim I holds. As $[F_l]^m$ was chosen arbitrarily, we have shown that we can remove $m - n$ branches in every hole of the context $C[-, \dots, -]$ which concludes this proof. \blacktriangleleft

► **Corollary 56.** *Let L_1, L_2 be two limits with $\text{sf}(L_1) = \mathbf{v}\vec{x}_1.\langle \Gamma_1 \rangle \parallel Q_1 \parallel R_1$ and $\llbracket L_1 \rrbracket \subseteq \llbracket L_2 \rrbracket$. Then, $\text{sf}(L_2 \otimes n) \stackrel{\alpha}{=} \mathbf{v}\vec{x}_1, \vec{x}_2.\langle \Gamma_2 \rangle \parallel Q_1 \parallel Q_2 \parallel R_2$ and $\Gamma_1 \leq_{\text{kn}} \Gamma_2$. for $n = |\vec{x}_1| + |Q_1| + 1$.*

Proof. First, we show that $\lceil \text{sf}(L_1) \rceil^0 \sqsubseteq_{\text{kn}} \lceil L_2 \rceil^n$. By $\llbracket L_1 \rrbracket \subseteq \llbracket L_2 \rrbracket$, we know that there is an m so that $\lceil \text{sf}(L_1) \rceil^0 \sqsubseteq_{\text{kn}} \lceil L_2 \rceil^m$ by Lemma 36. From Lemma 55, we obtain that that $\lceil \text{sf}(L_1) \rceil^0 \sqsubseteq_{\text{kn}} \Phi_{k, L_2}^{n, m}(\lceil L_2 \rceil^m)$ for every k . Substituting $\omega\text{-height}(L_2)$ for k leads to $\Phi_{\omega\text{-height}, L_2}^{n, m}(\lceil L_2 \rceil^m)$. This is $\lceil L_2 \rceil^n$ by Lemma 52. Using this knowledge embedding, we get $\lceil L_2 \rceil^n \stackrel{\alpha}{=} \mathbf{v}\vec{x}_1, \vec{x}_2.\langle \Gamma_2 \rangle \parallel Q_1 \parallel Q_2$ with $\Gamma_1 \leq_{\text{kn}} \Gamma_2$. With Lemma 38, we observe that $\lceil L_2 \rceil^n = \lceil L_2 \otimes n \rceil^0$. By this, the claim follows as $\lceil - \rceil^0$ just omits the iterated parts, i.e. R_2 , from $\text{sf}(L_2 \otimes n)$. ◀

► **Lemma 57 (Necessary Conditions for Inclusion).** *Let L_1 and L_2 be two limits, with $\text{sf}(L_1) \stackrel{\alpha}{=} \mathbf{v}\vec{x}_1.\langle \Gamma_1 \rangle \parallel Q_1 \parallel R_1$ with $R_1 = \prod_{i \in I} B_i^\omega$, and let $n = |\vec{x}_1| + |Q_1| + 1$. Given that the inclusion $\llbracket L_1 \rrbracket \subseteq \llbracket L_2 \rrbracket$, both conditions (A) and (B) hold.*

Proof. For (A), the claim follows by Corollary 56 which implicitly gives a renaming for L_2 . For (B), we want to show that $\llbracket \langle \Gamma_1 \rangle \parallel \prod_{i \in I} B_i \rrbracket \subseteq \llbracket \langle \Gamma_2 \rangle \parallel R_2 \rrbracket$. Let $N_i = \langle \Gamma_i \rangle \parallel R_i$. It is straightforward to see that $\llbracket \langle \Gamma_1 \rangle \parallel \prod_{i \in I} B_i \rrbracket \subseteq \llbracket N_1 \rrbracket$. This is why it is enough to show that $\llbracket N_1 \rrbracket \subseteq \llbracket N_2 \rrbracket$ by transitivity.

Towards a contradiction, we assume that for all possible renaming so that (A) is satisfied, $\llbracket N_1 \rrbracket \not\subseteq \llbracket N_2 \rrbracket$. By Lemma 36,

$$\exists m'_1, \forall m_1 \geq m'_1, \forall m_2. \lceil N_1 \rceil^{m_1} \not\sqsubseteq_{\text{kn}} \lceil N_2 \rceil^{m_2}. \quad (7)$$

First, knowledge could break the embedding. Let us define

$$\text{sf}(\lceil N_i \rceil^{m_i}) = \mathbf{v}y_i.\langle \Gamma_i \rangle \parallel \Gamma'_i \parallel Q'_i.$$

Notice that $\text{names}(\Gamma_i) \cap \vec{y}_i = \emptyset$. There might be two reasons why the knowledge embedding does not hold.

First, the embedding breaks because of knowledge. This is impossible as Γ_1, Γ'_1 and Γ_2, Γ'_2 represent the knowledge of groundings of the two limits L_1 and L_2 as their top level knowledge is replicated in (B). Therefore, the inclusion $\llbracket L_1 \rrbracket \subseteq \llbracket L_2 \rrbracket$ would also break which is a contradiction.

Second, names or process calls can hence be the only reasons why the knowledge embedding $\lceil N_1 \rceil^{m'_1} \sqsubseteq_{\text{kn}} \lceil N_2 \rceil^{m_2}$ does not hold for any m_2 . We will derive a contradiction by choosing $n_1 = 2 \cdot \max(|\vec{x}_2| + |Q_2| + 1, m'_1)$. We incorporate $\lceil R_1 \rceil^{n_1}$ into $\text{sf}(L_1)$: $\lceil \text{sf}(L_1) \rceil^{n_1} = \mathbf{v}\vec{x}_1.\langle \Gamma_1 \rangle \parallel Q_1 \parallel \lceil R_1 \rceil^{n_1}$. Recall that $\lceil \text{sf}(L_2 \otimes n) \rceil^{m_2} \stackrel{\alpha}{=} \mathbf{v}\vec{x}_1, \vec{x}_2.\langle \Gamma_2 \rangle \parallel Q_1 \parallel Q_2 \parallel \lceil R_2 \rceil^{m_2}$. By the size of n_1 , at least half of the names and process calls of $\lceil R_1 \rceil^{n_1}$ have to be covered by $\lceil R_2 \rceil^{m_2}$ as the non-iterated part \vec{x}_2 and Q_2 cannot do more than half. But by definition $\frac{n_1}{2} \geq m'_1$ so $\forall m_2, \lceil R_1 \rceil^{\frac{n_1}{2}} \not\sqsubseteq_{\text{kn}} \lceil R_2 \rceil^{m_2}$ as knowledge cannot be the reason for (7) to break. Altogether, this entails that there is a n_1 such that for all m_2 :

$$\begin{aligned} \lceil \text{sf}(L_1) \rceil^{n_1} &= \mathbf{v}\vec{x}_1.\langle \Gamma_1 \rangle \parallel Q_1 \parallel \lceil R_1 \rceil^{n_1} \\ &\not\sqsubseteq_{\text{kn}} \mathbf{v}\vec{x}_1, \vec{x}_2.\langle \Gamma_2 \rangle \parallel Q_1 \parallel Q_2 \parallel \lceil R_2 \rceil^{m_2} \stackrel{\alpha}{=} \lceil \text{sf}(L_2 \otimes n) \rceil^{m_2} \end{aligned}$$

Using Lemma 37, this implies that $\lceil L_1 \rceil^{n_1} \not\sqsubseteq_{\text{kn}} \lceil L_2 \rceil^{m_2}$ for every m_2 . In turn, this entails that $\llbracket L_1 \rrbracket \not\subseteq \llbracket L_2 \rrbracket$ by Lemma 36 which is a contradiction. ◀

G

 Correctness of $\widehat{\text{post}}$

► **Definition 58** ($\beta(\Delta)$ and b). Let Δ be a set of definitions. We define $\beta(\Delta)$ and $\gamma(\mathbb{I})$ as follows:

$$\begin{aligned}\beta(\Delta) &:= \max\{|\vec{x}| \mid (\mathbb{Q}[\vec{y}] := A + \mathbf{in}(\vec{x} : M).P + A') \in \Delta\} \\ \gamma(\mathbb{I}) &:= \max\{\text{ar}(\mathbf{f}) \mid \mathbf{f} \in \Sigma\} \text{ with } \mathbb{I} = (\Sigma, \vdash)\end{aligned}$$

then $b := \beta(\Delta) \cdot \gamma(\mathbb{I})^{s-1} + 1$.

Recall the definition of $\widehat{\text{post}}$ from Definition 25. Let $\text{sf}(L \otimes b) = \mathbf{v}\vec{x}.(\langle \Gamma \rangle \parallel Q \parallel R)$, then

$$\widehat{\text{post}}_{\Delta}^s(L) := \{ \mathbf{v}\vec{y}.(\langle \Gamma' \rangle \parallel Q' \parallel R) \mid \mathbf{v}\vec{x}.(\langle \Gamma \rangle \parallel Q) \rightarrow_{\Delta} \mathbf{v}\vec{y}.(\langle \Gamma' \rangle \parallel Q') \in \mathbb{S}_s \}$$

► **Corollary 59** (Post of expansion is enough). Let L be a limit and $P \in \llbracket L \rrbracket$. Then, for every $P_1 \in \text{post}(P)$, there is a $P_2 \in \text{post}(\llbracket L \rrbracket^n)$ for some $n \in \mathbb{N}$ such that $P_1 \sqsubseteq_{\text{kn}} P_2$.

Proof. As $P \in \llbracket L \rrbracket$, we know that there is a $n' \in \mathbb{N}$ such that $P \sqsubseteq_{\text{kn}} \llbracket L \rrbracket^{n'}$ by Lemma 34. We choose this n' to be n . We have that $P \rightarrow P_1$ and $P \sqsubseteq_{\text{kn}} \llbracket L \rrbracket^n$. As \sqsubseteq_{kn} is a simulation (Theorem 8), we know that there is a P_2 so that $\llbracket L \rrbracket^n \rightarrow P_2$ and $P_1 \sqsubseteq_{\text{kn}} P_2$. ◀

Let us recall the theorem we want to prove: for all $L \in \mathbb{L}_{s,k}$,

$$\widehat{\text{post}}_{\Delta}^s(L) = \{L_1, \dots, L_n\} \implies (\text{post}(\llbracket L \rrbracket) \cap \mathbb{S}_s) \downarrow = \llbracket L_1 \rrbracket \cup \dots \cup \llbracket L_n \rrbracket.$$

Proof of Theorem 26. We assume that $\widehat{\text{post}}_{\Delta}^s(L) = \{L_1, \dots, L_n\}$ and prove the set equality by inclusion in both directions.

First, we show that $(\text{post}(\llbracket L \rrbracket) \cap \mathbb{S}_s) \downarrow \supseteq \llbracket L_1 \rrbracket \cup \dots \cup \llbracket L_n \rrbracket$. It suffices to show that $\llbracket L_j \rrbracket \subseteq (\text{post}(\llbracket L \rrbracket) \cap \mathbb{S}_s) \downarrow$ for every $j \in \{1, \dots, n\}$. We choose j arbitrarily and prove the latter statement. By definition, we know that L_j stems from at least one transition in the non-iterated part of $\text{sf}(L \otimes b) = \mathbf{v}\vec{x}.(\langle \Gamma \rangle \parallel Q \parallel R)$. Wlog let $P_1 := \mathbf{v}\vec{x}.(\langle \Gamma \rangle \parallel Q) \rightarrow_{\Delta} \mathbf{v}\vec{y}.(\langle \Gamma' \rangle \parallel Q') =: P_2$ be this transition in \mathbb{S}_s . Hence, $L_j = \mathbf{v}\vec{y}.(\langle \Gamma' \rangle \parallel Q' \parallel R)$. By Lemma 37, $\llbracket L \rrbracket = \llbracket L \otimes b \rrbracket$ and therefore $\llbracket \mathbf{v}\vec{x}.(\langle \Gamma \rangle \parallel Q \parallel R) \rrbracket^n = \mathbf{v}\vec{x}.(\langle \Gamma \rangle \parallel Q \parallel \llbracket R \rrbracket^n) \in \llbracket L \rrbracket$ for every n . Because of $P_1 \rightarrow P_2$, we know that $\llbracket \mathbf{v}\vec{x}.(\langle \Gamma \rangle \parallel Q \parallel R) \rrbracket^n \rightarrow \llbracket \mathbf{v}\vec{y}.(\langle \Gamma' \rangle \parallel Q' \parallel R) \rrbracket^n$. This is why $\llbracket L_j \rrbracket^n = \llbracket \mathbf{v}\vec{y}.(\langle \Gamma' \rangle \parallel Q' \parallel R) \rrbracket^n \in \text{post}(\llbracket L \rrbracket)$ for every n . With Lemma 34, we know that for every $P \in \llbracket L_j \rrbracket$, there is a m so that $P \sqsubseteq_{\text{kn}} \llbracket L_j \rrbracket^m$. By downward-closure, we infer that $P \in ((\text{post}(L) \cap \mathbb{S}_s) \downarrow)$.

Second, we show that $(\text{post}(\llbracket L \rrbracket) \cap \mathbb{S}_s) \downarrow \subseteq \llbracket L_1 \rrbracket \cup \dots \cup \llbracket L_n \rrbracket$. Let $\llbracket L \rrbracket^b$ has standard form: $\text{sf}(\llbracket L \rrbracket^b) \stackrel{\alpha}{=} \mathbf{v}\vec{x}_1.(\langle \Gamma_1 \rangle \parallel \mathbb{Q}[\vec{M}] \parallel C_1)$. By Corollary 59, it suffices to consider only successors of groundings of L . Therefore, let $b < m \in \mathbb{N}$ with $\text{sf}(\llbracket L \rrbracket^m) \stackrel{\alpha}{=} \mathbf{v}\vec{x}_1, \vec{x}_2.(\langle \Gamma_1 \rangle \parallel \langle \Gamma_2 \rangle \parallel \mathbb{Q}[\vec{M}] \parallel C_2)$. We consider the three different reduction rules that can be fired starting from this grounding:

- a principal is waiting for some message on a private channel which is derivable and the pattern can be matched by Γ
- a principal would like to send a message on some private channel which can be derived from the environment
- one principal sends a message on a private channel to another principal

We split the parts into three paragraphs.

Public private channel, message input We have $Q[\vec{M}] := a(\vec{p} : N).P_1 + A$ for some private channel a and action A . Wlog we derive the message which is matched using some fresh intruder names \vec{c} : $\Gamma_1, \vec{c} \vdash N[\vec{M}'/\vec{x}]$ and the channel name a can be derived by assumption: $\Gamma_1, \Gamma_2 \vdash a$. Overall, we get the following transition:

$$[L]^m \xrightarrow[\vec{p} \rightarrow \vec{M}']{Q[\vec{M}] = a(\vec{p}:N).P_1 + A} \underbrace{\mathbf{v}\vec{x}_1, \vec{x}_2, \vec{c}.(\langle \Gamma_1 \rangle \parallel \langle \Gamma_2 \rangle \parallel \langle \vec{c} \rangle \parallel P_1[\vec{M}'/\vec{p}'] \parallel C_2)}_{Q' \in \text{post}^s(L)}$$

where the annotations explicitly state which process call and action was used with which substitution. We want to show that we can have the same reduction in $[L]^b$. We do so by using the Φ function as in Lemma 55 to relate the difference between iterating k times and $k + 1$ times.

Claim I: For every $k \in \mathbb{N}$, $\exists \vec{y}_k, \Delta_k, D_k$ such that

$$\Phi_{k,L}^{b,n}([L]^n) \xrightarrow[\vec{p} \rightarrow \vec{M}']{Q[\vec{M}] = a(\vec{p}:N).P_1 + A} \mathbf{v}\vec{y}_k, \vec{c}.(\langle \Delta_k \rangle \parallel \langle \vec{c} \rangle \parallel P_1[\vec{M}'/\vec{p}'] \parallel D_k) \quad (8)$$

with $\Delta_k, \vec{c} \vdash N[\vec{M}'/\vec{p}']$ and $\Delta_k, \vec{c} \vdash a$.

Proof of Claim I by induction on k : For the base case in which $k = 0$, the claim holds by assumption. For the induction step, we assume that (8) holds for k and we prove it for $k + 1$. Similar to the proof of Lemma 55, we use a multi-hole context $C[-, \dots, -]$ to distinguish between having budget k or $k + 1$. Let F_1, \dots, F_j be j limits and $C[-, \dots, -]$ a multi-hole context so that:

$$\begin{aligned} \Phi_{k,L}^{b,n}([L]^n) &\equiv C[\Phi_{0,F_1^\omega}^{b,n}(F_1^\omega), \dots, \Phi_{0,F_j^\omega}^{b,n}(F_j^\omega)] = C[[F_1^\omega]^n, \dots, [F_j^\omega]^n] \\ &= C[[[F_1]^n]^n, \dots, ([F_j]^n)^n] = A \\ \Phi_{k+1,L}^{b,n}([L]^n) &\equiv C[\Phi_{1,F_1^\omega}^{b,n}(F_1^\omega), \dots, \Phi_{1,F_j^\omega}^{b,n}(F_j^\omega)] \\ &= C[[[F_1]^n]^b, \dots, ([F_j]^n)^b] = B \end{aligned}$$

We want to show that it suffices to have b copies of F_l for any $1 \leq l \leq j$. Since $Q[\vec{M}]$ also occurs in $\text{sf}([L]^b)$, it is trivial to keep the process call which is reduced. It remains to argue that the same redex is enabled in B. Towards a contradiction: Assume that there is a hole in which b copies are not sufficient. Wlog let L_l be the limit in this hole.

We did not explicitly single out the message $N[\vec{M}'/\vec{p}']$ but only the substitution $[\vec{M}'/\vec{p}']$ for the continuation since the substitution is solely determining the successor.

We know that A 's knowledge is Δ_k and factor out the knowledge from sublimit $[F_l]^m$: $\Delta_k = \Delta'_k, \Delta_{l,m}$ so that $\Delta_{l,m}$ was the knowledge obtained through $[F_l]^m$. We want to prove that $\Delta'_k, \Delta_{l,m}, \vec{c} \vdash N[\vec{M}'/\vec{p}'] \implies \Delta'_k, \Delta_{l,b}, \vec{c} \vdash N[\vec{M}'/\vec{p}']$ where $\Delta_{l,b}$ denotes the knowledge obtained through $[F_l]^b$ respectively.

Now, we consider the different names used in $N[\vec{M}'/\vec{p}']$. It is straightforward to see that $\text{names}(N) \subseteq \vec{x}_1$. Recall the definition of b :

$$b := \beta(\Delta) \cdot \gamma(\mathbb{I})^{s-1} + 1$$

By this, we know that $\beta(\Delta) \geq |\vec{p}'|$ and hence $b \geq |\vec{p}'| \cdot \gamma(\mathbb{I})^{s-1} + 1$. Intuitively, this ensures that there are enough distinct names for every single parameter in the parameter list as well as for channel a as the size determines the maximum depth of the syntax tree of a message. As $\text{names}(\vec{p}) < b$, we can remove at least $m - b$ branches without loosing names used in \vec{p}' or channel name a . Therefore, we can assume that $\text{names}(N[\vec{M}'/\vec{p}']) \cup \{a\} \subseteq \vec{x}_1$. The idea

is to reduce the knowledge from $\Delta_{l,m}$ to $\Delta_{l,b}$ by Corollary 48, which is a corollary of the absorbing intruder. As m might be bigger than $2b$, we have to iterate the process of reducing the number of branches. Hence, we generalise the notation of $\Delta_{l,m}$ and $\Delta_{l,b}$ in the obvious way: $([F_l]^m)^i \equiv \mathbf{v}\vec{a}_i.(\Delta_{l,i} \parallel \dots)$.

Claim II: For all $m > b$,

$$(\Delta_{l,m} \vdash N[\vec{M}/\vec{p}] \wedge \Delta_{l,m} \vdash a) \implies (\Delta_{l,m-1} \vdash N[\vec{M}/\vec{p}] \wedge \Delta_{l,m-1} \vdash a).$$

Proof of Claim II. For convenience, we rename $\Delta_{l,i}$ to Λ_i . The main observation is that we can split the knowledge Λ_m into Λ_{m-1} and a remainder Λ' . We can choose a branch which does not use names from \vec{x}_1 to contribute to Λ' . Since we know that $n \geq 1$, we know that $m > 1$ by assumption. Therefore, we can split Λ_{m-1} again and obtain the knowledge stemming from one branch which we call Λ'' . Let us recall the assumption and goal after these rewriting steps: Given that

$$\Delta'_k, \Lambda_{m-2}, \Lambda', \Lambda'' \vdash N[\vec{M}/\vec{p}] \quad \text{and} \quad \Delta'_k, \Lambda_{m-2}, \Lambda', \Lambda'' \vdash a \quad (9)$$

holds, we want to prove that $\Delta'_k, \Lambda_{m-2}, \Lambda'' \vdash N[\vec{M}/\vec{p}]$ and $\Delta'_k, \Lambda_{m-2}, \Lambda'' \vdash a$. Let \vec{w}' and \vec{w}'' be the names only used in Λ' and Λ'' respectively so that:

$$\vec{w}' \cap \vec{w}'' = \emptyset \quad \text{and} \quad \text{names}(\Gamma'_{c,m}, \Lambda_{m-2}) \cap \vec{w}' = \emptyset = \text{names}(\Gamma'_{c,m}, \Lambda_{m-2}) \cap \vec{w}'' \quad (10)$$

Λ' and Λ'' have been obtained from a branch of the same sublimit, so we can infer that

$$\Lambda'' = \Lambda'[\vec{w}''/\vec{w}']. \quad (11)$$

Notice that $\vec{w}' \cap \vec{x}_1 = \emptyset$ by the fact how we have chosen the the branch for Λ' . Because of $\text{names}(N[\vec{M}/\vec{p}]) \cup \{a\} \subseteq \vec{x}_1$, we can infer that $(\text{names}(N[\vec{M}/\vec{p}]) \cup \{a\}) \cap \vec{w}' = \emptyset$ which implies that

$$\text{names}(N[\vec{M}/\vec{p}]) \cup \{a\} \subseteq \text{names}(\Delta'_k, \Lambda_{m-2}, \Lambda'') \quad (12)$$

Facts (9) to (12) fulfil the conditions for Corollary 48 resulting in $\Delta'_k, \Lambda_{m-2}, \Lambda'' \vdash N[\vec{M}/\vec{p}]$ and $\Delta'_k, \Lambda_{m-2}, \Lambda'' \vdash a$ which reads $\Delta'_k, \Lambda_{m-1} \vdash N[\vec{M}/\vec{p}]$ and $\Delta'_k, \Lambda_{m-1} \vdash a$ respectively when folding back which is the goal of Claim II.

In turn, this concludes the proof of Claim I.

Instantiating the statement of Claim I with $k > \omega\text{-height}(L)$ shows that this transition is still enabled in $[L]^b$. We prove that the same holds for transitions enabled by the remaining reduction rules.

Public private channel, message output Recall that $\text{sf}([L]^m) \stackrel{\circ}{=} \mathbf{v}\vec{x}_1, \vec{x}_2.(\langle \Gamma_1 \rangle \parallel \langle \Gamma_2 \rangle \parallel \mathbf{Q}[\vec{M}] \parallel C_2)$. We have $\mathbf{Q}[\vec{M}] := \bar{a}\langle M \rangle.P_1 + A$ for some private channel a and action A . The channel name a can be derived by assumption: $\Gamma_1 \vdash a$. Overall, we get the following transition:

$$[L]^m \xrightarrow{\mathbf{Q}[\vec{M}] = \bar{a}\langle M \rangle.P_1 + A} \underbrace{\mathbf{v}\vec{x}_1, \vec{x}_2, \vec{c}.(\langle \Gamma_1 \rangle \parallel \langle \Gamma_2 \rangle \parallel \langle M \rangle \parallel P_1 \parallel C_1)}_{Q' \in \text{post}^s(L)}$$

and the channel name a can be derived by assumption: $\Gamma_1, \Gamma_2 \vdash a$. We want to show that we can have the same reduction in $[L]^b$. We sketch the proof that this transition is still enabled as it is analogous to the first proof. We start with an analogous claim.

Claim I: In every hole of context $C[-, \dots, -]$, we can reduce the number of branches to at most n , i.e. for every j we can have a grounding $\lceil F_j \rceil^n$ instead of $\lceil F_j \rceil^m$. for every $k \in \mathbb{N}$:

$$\exists \vec{y}_k, \Delta_k, D_k: \Phi_{k,L}^{b,n}(\lceil L \rceil^n) \xrightarrow{\mathbb{Q}[\vec{M}] = \bar{a}\langle M \rangle + A} \mathbf{v}\vec{y}_k.(\langle \Delta_k \rangle \parallel P_1 \parallel \langle M \rangle \parallel D_k) \\ \text{with } \Delta_k \vdash a$$

It is obvious to see that the side condition is subsumed by (8) which we have used in the first case and hence the proof could be proceeded in the same way.

Private channel, two participants For the last case, we consider:

$$\text{sf}(\lceil L \rceil^m) \triangleq \mathbf{v}\vec{x}_1, \vec{x}_2.(\langle \Gamma_1 \rangle \parallel \langle \Gamma_2 \rangle \parallel \mathbb{Q}_1[\vec{M}_1] \parallel \mathbb{Q}_2[\vec{M}_2] \parallel C_2) \text{ with} \\ \mathbb{Q}_1[\vec{M}_1] \triangleq \bar{c}\langle N[\vec{M}'/\vec{x}] \rangle.P_1 + A_1 \\ \mathbb{Q}_2[\vec{M}_2] \triangleq c(\vec{x} : N).P_2 + A_2$$

The resulting transition looks as follows:

$$\mathbf{v}\vec{x}_1, \mathbf{v}\vec{x}_2.(\langle \Gamma_1 \rangle \parallel \langle \Gamma_2 \rangle \parallel \mathbb{Q}_1[\vec{M}_1] \parallel \mathbb{Q}_2[\vec{M}_2] \parallel C_2) \\ \rightarrow_{\Delta} \mathbf{v}\vec{x}_1, \vec{x}_2.(\langle \Gamma_1 \rangle \parallel \langle \Gamma_2 \rangle \parallel P_1 \parallel P_2[\vec{M}'/\vec{x}] \parallel C_2)$$

There are no side conditions on derivability in this case so we only have to ensure that all possible combinations of $\bar{c}\langle N[\vec{M}'/\vec{x}] \rangle.P_1$ and $c(\vec{x} : N).P_2$ can occur. As both actions could stem from the same process call, it suffices to have an expansion factor greater or equal to 2. This only poses restrictions on β and γ . By definition of b , we need to ensure that $\beta(\Delta) \cdot \gamma(\mathbb{I})^{s-1} \geq 1$. As argued before, considering $s < 2$ is unreasonable since it would be impossible to have encryptions in this setting. Hence, even if $\gamma(\mathbb{I})$ was 0, $\gamma(\mathbb{I})^{s-1} \geq 1$ for reasonable s . If $\beta(\Delta) < 1$, it is 0 and hence the transition we consider would be impossible. We therefore can assume that $\beta(\Delta) > 0$. Hence, we know that $b \geq 2$ and this suffices to have a congruent transition in $\lceil L \rceil^b$.

We now turn back to our general goal and know that all kinds of transitions are enabled in $\lceil L \rceil^b$. Consider the extension $L \otimes b$ of limit L whose standard form we choose to resemble the correlation with $\lceil L \rceil^b$: $\text{sf}(L \otimes b) \triangleq \mathbf{v}\vec{x}_1.(\langle \Gamma_1 \rangle \parallel \mathbb{Q}[\vec{M}] \parallel C_1 \parallel R_1)$. By the definition of $\widehat{\text{post}}(L)$, we know that

$$L' := \mathbf{v}\vec{x}_1.(\langle \Gamma_1 \rangle \parallel P_1[\vec{M}/\vec{p}] \parallel C_1 \parallel R_1) \in \widehat{\text{post}}(L).$$

Recall that $Q' = \mathbf{v}\vec{x}_1, \vec{x}_2, \bar{c}.(\langle \Gamma_1 \rangle \parallel \langle \Gamma_2 \rangle \parallel \langle \bar{c} \rangle \parallel P_1[\vec{M}'/\vec{p}] \parallel C_1)$ was the successor of $\lceil L \rceil^m$ from the beginning. It remains to show that $Q' \in \llbracket L' \rrbracket$. As before for $L \otimes b$, we relate the standard form of $L \otimes m$ to $\lceil L \rceil^m$ and get the following:

$$\text{sf}(L \otimes m) \triangleq \mathbf{v}\vec{x}_1, \vec{x}_2.(\langle \Gamma_1 \rangle \parallel \langle \Gamma_2 \rangle \parallel \mathbb{Q}[\vec{M}] \parallel C_2 \parallel R_2)$$

By Lemma 37, we have that $\llbracket L \otimes m \rrbracket = \llbracket L \otimes b \rrbracket$ and taking one step hence leads to

$$K' := \llbracket \mathbf{v}\vec{x}_1, \vec{x}_2, \bar{c}.(\langle \Gamma_1 \rangle \parallel \langle \Gamma_2 \rangle \parallel \langle \bar{c} \rangle \parallel P_1[\vec{M}'/\vec{p}] \parallel C_2 \parallel R_2) \rrbracket \\ = \llbracket \mathbf{v}\vec{x}_1.(\langle \Gamma_1 \rangle \parallel \langle \bar{c} \rangle \parallel P_1[\vec{M}'/\vec{p}] \parallel C_1 \parallel R_1) \rrbracket =: L'$$

As $Q' \in K'$, it also holds that $Q' \in L'$ which concludes this proof. \blacktriangleleft

H 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 [33] 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:

$$M_1 \oplus (M_2 \oplus M_3) \cong (M_1 \oplus M_2) \oplus M_3 \qquad M_1 \oplus M_2 \cong M_2 \oplus M_1 \qquad (13)$$

$$M \oplus 0 \cong M \qquad M \oplus M \cong 0 \qquad (14)$$

The results of [1, 12] establish that the laws (14) 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_{AC} defined by laws (13). Then the relation $\rightsquigarrow_{AC} := (\cong_{AC} \circ \rightsquigarrow)$ is terminating, and confluent modulo \cong_{AC} . The set of normal forms $M\Downarrow := \{N \mid M \rightsquigarrow_{AC}^* N \not\rightsquigarrow_{AC}\}$ 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

$$\frac{}{\Gamma \vdash 0} \text{XOR}_0 \qquad \frac{\Gamma \vdash M_1 \quad \Gamma \vdash M_2}{\Gamma \vdash M_1 \oplus M_2} \text{XOR}_R \qquad \frac{\Gamma, M_1, M_2, M_1 \oplus M_2 \vdash N}{\Gamma, M_1, M_2 \vdash N} \text{XOR}_L$$

$$\frac{\Gamma, M, M\Downarrow \vdash N}{\Gamma, M \vdash N} \Downarrow_L \qquad \frac{\Gamma \vdash N \quad N \in M\Downarrow}{\Gamma \vdash M} \Downarrow_R$$

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 losing 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}_{en} . The proofs of (Relevancy) and absorption make use of the fact that if $N \in M\Downarrow$ then $\text{fn}(N) \subseteq \text{fn}(M)$.

We conjecture Diffie-Hellman exponentiation (following the model of e.g. [31]) 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.

I 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 $\text{ty}: \mathcal{N} \rightarrow \mathbb{N}$, assigning to each pattern variable a maximum size for the messages it can match. A typing induces a typed transition relation $\rightarrow_{\Delta, \text{ty}}$ which only matches patterns with substitutions respecting ty ; $\text{reach}_{\Delta}^{\text{ty}}(P)$ collects all the terms reachable from P through $\rightarrow_{\Delta, \text{ty}}$. A typing ty of P , Δ is s -bounding if $\text{reach}_{\Delta}^{\text{ty}}(P) \subseteq \mathbb{S}_s$. One can check if ty is s -bounding on-the-fly while computing $\widehat{\text{post}}$.

I.1 Generalised Limits

Recall the “encryption oracle” of Example 15: $E[k] = \mathbf{in}(x : x).(\langle e(x)_k \rangle \parallel E[k])$. 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 $\text{ty} = [x \mapsto 1]$, to obtain the limit $E[k] \parallel (\mathbf{vm}.\langle e(m)_k \rangle)^\omega$ which is inductive (wrt $\rightarrow_{\Delta, \text{ty}}$) and contains the initial state $E[k]$. 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^* .

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

► **Definition 60** (Wildcard semantics). *Given $P \in \mathbb{P}^*$, its wildcard semantics is defined as*

$$\mathcal{W}[[P]] := \left\{ P' \left| \begin{array}{l} \text{sf}(P) = \mathbf{v}\vec{x}.(\langle \Gamma \rangle \parallel Q), P' \equiv_{\text{kn}} \mathbf{v}\vec{x}, \vec{c}.(\langle \langle \Gamma \rangle \parallel Q \rangle [\vec{M}/\vec{y}^*]) \\ \vec{y}^* = \text{names}(P) \cap \mathcal{N}^*, \text{names}(\vec{M}) \subseteq \vec{x} \cup \vec{c} \end{array} \right. \right\}$$

For $L \in \mathbb{L}_{s,k}^*$, 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^* . Then we can check that the wildcards generalise the limit enough to make it inductive wrt \rightarrow_{Δ} (i.e. without restricting the message size):

$$\forall P \in \mathcal{W}[[L^*]], \forall Q: P \rightarrow_{\Delta} Q \implies Q \in \mathcal{W}[[L^*]] \quad (15)$$

For our encryption oracle example, the annotated limit $E[k] \parallel (\nu m. \langle e(m^*)_k \rangle)^\omega$ 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 $\mathbf{ty} = [n_a \mapsto 1]$. To remove the size bound we only need to annotate the occurrence of n_a in L_2 obtaining the limit L^* :

$$\begin{aligned} L^* &= \nu a, b, k_{as}, k_{bs}. (\langle a, b \rangle \parallel A_1[a, b, k_{as}]^\omega \parallel B_1[a, b, k_{bs}]^\omega \parallel S[a, b, k_{as}, k_{bs}]^\omega \parallel L_1^\omega) \\ L_1 &= \nu n_a. (\langle n_a \rangle \parallel A_2[a, b, k_{as}, n_a] \parallel L_2^\omega) \\ L_2 &= \nu k. (\langle e(k)_{(n_a^*, k_{as})} \rangle \parallel \langle e(k)_{k_{bs}} \rangle \parallel \text{Secret}[k]^\omega \parallel A_3[a, b, k_{as}, k]^\omega \parallel L_3^\omega) \\ L_3 &= \nu n_b. (\langle e(n_b)_{(k, k)} \rangle \parallel \langle e(n_b)_k \rangle \parallel B_2[a, b, k_{bs}, n_b, k]) \end{aligned}$$

Indeed this generalised limit is inductive: the intruder can send any message (M, b) to the server $S[a, b, k_{as}, k_{bs}] := \mathbf{in}(x : (x, b)). (\dots \langle e(k)_{(x, k_{as})} \rangle \dots)$, 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 $e(k)_{(M, k_{as})}$ 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 so that the generalised limits satisfy condition (15):

- ARPC: There is only one annotation reasonable: $(n_x^*, (k, b))$. This is still inductive even though it can be fed back as we can have a different substitutions for n_x^* .
- KSL: We annotate $B_2[a, b, k_{bb}, k_{bs}, n_x^*, n_y]$ in L_2 and $e(nz, (b, n_x^*))_{k_{xy}}$ 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 $e(m_x^*, m_y)_{k_{xy}}$ and $B_5[a, b, k_{bb}, k_{bs}, k_{ab}, t_y, m_x^*, m_y]$ in L_7 . The new message cannot flow into $B_3[-]$ or $D_2[-]$ 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 , $B_4[-]$ can produce $B_5[-]$ ’s input message. This is still covered and inductive but does not comply with an normal reauthentication run.
- NHS: We annotate $e(n_x^*, (b, e(k, a)_{k_{bs}}))_{k_{as}}$ 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 $e(one, s^*)_k$ in L_3 and hence have to annotate the s in $\text{Secret}[s^*]$ as well as $\text{Leak}[s^*]$. This covers all the enabled transitions and is hence inductive.
- NHSs: The same annotation as in NHS works.
- OR/ORl: We annotate $e(n_y, (m^*, (a, b)))_{k_{bs}}$ 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 $e(a, (n_a^*, n_b))_{k_{bs}}$ in L_3 . This can be fed back to $B_2[-]$ and hence we also annotate $e(n_b)_{(n_a^*, n_b)}$ in L_5 . This is still inductive.
- 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 $e(one, s^*)_k$ 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.

I.2 Towards Automating Generalisation

Automating the check of condition (15) 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 $\widehat{\text{post}}_{\Delta, \star}^{\text{ty}}$, a “symbolic” version of $\widehat{\text{post}}$, and \Subset satisfying:

$$L_1^* \Subset L_2^* \implies \mathcal{W}[L_1^*] \subseteq \mathcal{W}[L_2^*] \quad (16)$$

$$\text{post}(\mathcal{W}[L^*]) \subseteq \mathcal{W}[\widehat{\text{post}}_{\Delta, \star}^{\text{ty}}(L^*)] \quad (17)$$

With these components, one can check (15) by checking $\widehat{\text{post}}_{\Delta, \star}^{\text{ty}}(L^*) \Subset L^*$. Note that (16) is an implication, and (17) 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_{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 $x^* \leq_{\text{kn}} x^*$ and $M \leq_{\text{kn}} x^*$.

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

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

and use it in $\widehat{\text{post}}_{\Delta, \star}^{\text{ty}}$ 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.

J Benchmarks

The Needham-Schröder protocol [27] 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 [28]. 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 [24] (we model $\text{succ}(-)$ with pairs, i.e. $\text{succ}((\text{zero}, -))$ is $(\text{one}, -)$). We modelled the Kehne-Schönwälder-Landendörfer protocol [24], 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.

K Example of Incorporation

Consider the limit $L = \mathbf{v}y. \left((\mathbf{v}x. (A[x] \parallel B[y, x]^\omega))^\omega \right)$. To compute $\widehat{\text{post}}$ we consider the limit $L \otimes 1 = \mathbf{v}y. \left((\mathbf{v}x. (A[x] \parallel B[y, x] \parallel B[y, x]^\omega)) \parallel (\mathbf{v}x. (A[x] \parallel B[y, x]^\omega))^\omega \right)$ and assume

$$A[x] \rightarrow \mathbf{v}z. (A[z] \parallel B[y, z]) \parallel B[y, x] = P.$$

Then, $\widehat{\text{post}}$ will contain the limit $L' = C[P]$ for $C[\bullet] = \mathbf{v}y. \left((\mathbf{v}x. (\bullet \parallel B[y, x] \parallel B[y, x]^\omega)) \parallel (\mathbf{v}x. (A[x] \parallel B[y, x]^\omega))^\omega \right)$. Since we are trying to prove inclusion between L and $L' = C[P]$, and L is equivalent to $L \otimes 1 = C[A[x]]$, 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:

$$\mathbf{v}y. \left((\mathbf{v}x. (A[x] \parallel B[y, x] \parallel B[y, x]^\omega)) \parallel (\mathbf{v}x. (A[x] \parallel B[y, x]^\omega))^\omega \right)$$

$$\mathbf{v}y. \left((\mathbf{v}x. (\mathbf{v}z. (A[z] \parallel B[y, z]) \parallel B[y, x] \parallel B[y, x]^\omega)) \parallel (\mathbf{v}x. (A[x] \parallel B[y, x]^\omega))^\omega \right)$$

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 $\forall L' \in \widehat{\text{post}}_\Delta^s(L): \llbracket L' \rrbracket \subseteq \llbracket L \rrbracket$. By construction, $L' = C[P]$ for some context C such that $L \otimes b = C[\mathbf{Q}[\vec{M}]]$. Then, if we can find P in the fixed part of $\text{sf}(L \otimes n)$, conditions (A) and (B) of Theorem 22 are automatically satisfied because of the shared context C .