A Higher-Order Indistinguishability Logic for Cryptographic Reasoning

David Baelde
Univ Rennes, CNRS, IRISA, France

Adrien Koutsos
Inria, France

Joseph Lallemand
Univ Rennes, CNRS, IRISA, France

Abstract—The field of cryptographic protocol verification in the computational model aims at obtaining formal security proofs of protocols. To facilitate writing such proofs, which are complex and hard to automate, Bana and Comon have proposed the Computationally Complete Symbolic Attacker (CCSA) approach, which is based on a first-order logic with a probabilistic computational semantics. Later, a *meta-logic* was built on top of the CCSA logic, to extend it with support for unbounded protocols and effective mechanisation. This meta-logic was then implemented in the SQUIRREL prover.

In this paper, we propose a careful re-design of the SQUIRREL logic, providing clean and robust foundations for its future development. We show in this way that the original meta-logic was both needlessly complex and too restrictive. Our new, higher-order logic avoids the indirect definition of the meta-logic on top of the CCSA logic, decouples the logic from the notion of protocol, and supports advanced generic reasoning and non-computable functions. We also equip it with generalised cryptographic rules to reason about corruption. This theoretical work justifies our extension of SQUIRREL with higher-order reasoning, which we illustrate on case studies.

I. INTRODUCTION

Cryptographic protocols are widely used to secure all sorts of communications: web browsing, payment, instant messaging, *etc.* Concretely, they are simple distributed programs that rely on cryptographic constructions (encryption, signatures...). As the security of many critical systems, and the privacy of many users, depend on these protocols, it is essential to ensure they do not have flaws or vulnerabilities. In order to formally establish this, one must take into account the presence or an arbitrary *adversary*, or *attacker*. As it turns out, considering adversarial behaviours renders the analysis particularly difficult. Formal security proofs therefore tend to be rather complex and error prone. All these reasons have motivated much work on developing and implementing mechanised formal methods for cryptographic protocols.

The first step before devising verification techniques is to formally define the model under study. In our case, we must notably define what the adversary can and cannot do. One important class of models, sometimes called *symbolic*, only considers idealised attackers, to enable highly-automated verification techniques. This line of work pioneered by Dolev and Yao [1] has lead to mature verification tools [2]–[4]. We are concerned in this paper with a more concrete model, the

so-called *computational model*, in which messages are bit-strings and adversaries are probabilistic polynomial-time Turing machines (PPTMs). In this model, security properties generally do not hold in an absolute sense: although polynomial time attackers cannot mount brute force attacks, they may still luckily guess a secret key and break the property. Hence, one considers that a property holds when it is true with *overwhelming probability* for any PPTM attacker. Here, the probability is evaluated as a function f of a *security parameter* η – typically the length of secret keys. We say that a function f is negligible when it is asymptotically smaller than the inverse of any polynomial η^{-k}, and that it is overwhelming when $1 - f$ is negligible. We write the former $f \in \mathsf{negl}(\eta)$, and the latter $f \in \mathsf{ow}(\eta)$. Proving security in this sense is typically done through a reduction, showing that breaking the security of the system (with good probability) entails violating a computational hardness assumption on some cryptographic primitive. The computational model is more realistic than symbolic ones, and thus provides stronger security guarantees. It is the standard for cryptographers. But this comes at a cost: automating proofs is more difficult, and existing tools either have a limited scope (*e.g.* [5]), or use low-level modelling with little support for automated reasoning, and may require users to write intricate proofs involving probabilistic arguments (*e.g.* [6], [7]).

Example 1. *We provide examples of an information-theoretic fact phrased in terms of negligible probability, and a simple cryptographic assumption.*

Consider two probabilistic machines that receive as input the security parameter η, and draw a bit-string of length η uniformly at random. If the two samplings are independent, the probability that the two machines return the same result is $2^{-\eta}$, which is negligible in η.

Cryptographic assumptions are also phrased in terms of negligible probabilities, but crucially rely on limitations of the computational resources of the attacker. For instance, assume a keyed hash function, used with some randomly sampled key k *of length η. Consider a probabilistic polynomial-time attacker with the ability to compute hashes using this key* k, *but without direct access to* k. *The* collision resistance under hidden key attacks *assumption states that there is a negligible probability (in η) that such an attacker finds two different messages with the same hash. The polynomial time assumption is crucial here, as unrestricted attackers clearly exist, because hash functions cannot be injective.*

1

A few years ago, Bana and Comon have proposed a symbolic framework [8] allowing to prove computational security, providing a way to obtain strong security guarantees while writing high-level proofs that can be more easily automated. Their approach, called the Computationally Complete Symbolic Attacker (CCSA), builds on first-order logic. First-order terms are used to model messages, using function symbols for cryptographic operations and adversarial computations, and special constants called *names* to model secrets, nonces, *etc.* as in symbolic models. Complex security properties can then be naturally expressed as first-order formulas, using a single predicate $\sim$ interpreted as *computational indistinguishability* – we will provide later a full definition of this notion but, for now, we only use $u \sim$ true with u a boolean, which means that u is true with overwhelming probability. The semantics is given by considering a specific class of first-order structures, where terms are interpreted as probabilistic bit-strings. Function symbols representing adversarial computations are interpreted as PPTMs. Those representing cryptographic primitives are interpreted as deterministic Turing machines[1] satisfying the relevant security assumptions . Names are interpreted as independent uniform random samplings of bit-strings of length η. A proof system is finally associated to this logic, with rules that encode computational security assumptions on the primitives, whose soundness justifies the ability of the CCSA logic to carry security proofs in the computational model. This proof system, and the accompanying methodology, have been successfully used on various protocols [9]–[11].

Example 2. *When* n *and* n$'$ *are distinct name symbols, the boolean term* $n \neq n'$ *is true with probability* $1 - 2^{-\eta}$ *in any CCSA model (equality is not a predicate in this logic, but is only used as a function symbol). Hence, the atomic formula* $(n \neq n') \sim$ true *is valid.*

Further, if $h(\_,\_)$ *is interpreted as a collision resistant keyed hash function (under hidden key attacks), and* u, v *are two closed terms in which* k *only occurs as second argument of* h, *then* (if $h(u, k) = h(v, k)$ then $u = v$ else true) $\sim$ true *is valid: this is because* u *and* v *can be seen as probabilistic polynomial time computations without access to* k, *except for the ability to compute hashes with that key.*

In many cases though, we are interested in properties of the *execution* of protocols, such as "in any execution, no attacker can make it so that two honest agents disagree on the value of a supposedly shared key". To handle these, a *meta-logic* was developed in [12] on top of the CCSA logic, called *base logic* in this context, which internalises the notion of execution of a protocol. In addition to messages, meta-logic terms can describe timestamps (positions in an implicit execution trace) and indices (used to model unbounded collections). Special terms, called *macros*, are used to represent the attacker's knowledge and the inputs and outputs of the protocol at a particular timestamp. Meta-logic formulas are first interpreted

---

[1]Randomised primitives, *e.g.* encryption, are modelled as deterministic algorithms inputting the randomness explicitly as an additional argument.

in a *trace model* encoding one specific execution. They then devolve into boolean terms of the base logic, that can be interpreted as before. The proof rules on the base level are lifted to the meta level, yielding a proof system that can be used to prove properties of all protocol executions. That system has been implemented in the SQUIRREL proof assistant [13], and successfully used to study several protocols [12], [14], [15].

Example 3. *In the Hash Lock protocol [16], a RFID tag owns a key* k. *At session* i, *it inputs a message* x *and outputs in return a pair* $\langle n_i, h(\langle x, n_i \rangle, k) \rangle$ *where* n_i *is freshly generated. This protocol ensures that the second components of the outputs of distinct tag sessions are always distinct, provided that the hash function* h *is collision resistant.*

To verify this using the base logic, we have to model the successive inputs and outputs of the protocol. For each new session i, *the input is an arbitrary attacker computation from the past outputs, which yields the following definitions:*
- $out_i = \langle n_i, h(\langle in_i, n_i \rangle, k) \rangle$;
- $in_0 = att_0()$ *and* $in_{i+1} = att_{i+1}(out_0, \ldots, out_i)$.

Then we have to check, for all $i, j \in \mathbb{N}$, *the validity of:*

$$\big(h(\langle in_i, n_i \rangle, k) \neq h(\langle in_j, n_j \rangle, k) \big) \sim \text{true} \quad \text{when } i \neq j$$

Indeed, all of these formulas can be derived from axioms notably expressing the collision resistance of h.

The meta-logic allows to capture all of the above base logic formulas as the following meta-logic formula (with minor details elided), which is proved just as easily in the meta-logic as each of the previous base logic formulas was:

$$\tilde{\forall} i, j. \, [i \neq j]_{\mathcal{P}} \, \tilde{\Rightarrow} \, [snd(output@T(i)) \neq snd(output@T(j))]_{\mathcal{P}}$$

Here, $\mathcal{P}$ *represents our protocol. In more details, it is formed of actions* $T(i)$, *representing the interaction with session* i *of the tag. In this context, a trace model specifies in which finite domain* $\mathcal{D}_I$ *the* indices i *are interpreted, and in which order the actions* $(T(i))_{i \in \mathcal{D}_I}$ *are scheduled. In such a trace model, a term of sort timestamp is interpreted into some* $T(i)$ *for* $i \in \mathcal{D}_I$, *or into the special initial timestamp* init *that precedes all actions.*

The meta-logic allows macro *terms such as* $output@T(i)$, *which represents the protocol output at a given timestamp. In this case, using the ability to write indexed names to model unbounded collections we have:*

$$output@T(i) = \langle n(i), h(\langle input@T(i), n(i) \rangle, k) \rangle$$

The input at a given timestamp (other than init*) is then defined as the result of an unspecified adversarial computation on the* frame *at this timestamp, i.e. (roughly) the sequence of past outputs. The semantics of the* output, input *and* frame *macros are thus defined in a mutually recursive fashion, once the relative execution order of actions is fixed by a trace model, depending of course on the protocol specification* $\mathcal{P}$.

In the meta-logic, a sub-formula ϕ *in square brackets is called* local, *and is interpreted as a boolean value. Global meta-logic formulas are built on top of local ones: e.g.* [ϕ] *asserts*

that ϕ is overwhelmingly true. For clarity, we distinguish the two sorts of meta-logic connectives (e.g. local $\forall$ vs. global $\tilde{\forall}$).

Contributions: In this paper, we present a new logic which both simplifies and generalises the meta-logic of [12], [14]. This new development is beneficial in several ways.

i) Disentangling the logic from the protocol. The meta-logic approach inextricably intertwines the definition of the logic and the notion of protocol. The logic is thus tied to both a class of protocols and an execution model, which is detrimental to the readability and extensibility of the approach. This is visible in [14] which introduces stateful protocols along with a more flexible execution model (only a subset of actions is scheduled for execution): this forces a redefinition of the logic, potentially hiding the fact that most results and proof rules for the old logic actually carry over unchanged to the new one.

Our new logic is independent from any notion of protocol. Instead, it comes with a generic mechanism for forming mutually recursive definitions. This mechanism can be used to re-express the former macros encoding the notions of protocol and execution model. It could also be used to encode new execution models, with private channels, side channels, distance bounding, partially passive adversaries, *etc.*

ii) An expressive, self-contained logic. Rather than building a meta-logic, our new logic is self-contained. With this more direct presentation, it is no longer possible to justify cryptographic rules by lifting known CCSA rules. This is a liberation rather than a limitation: while the CCSA logic comes with strict overall assumptions on interpretations, we design our logic with minimal assumptions, adding constraints only when necessary. For instance, CCSA terms only model PPTMs, but we allow non-polynomial and even non-computable terms, which lets us considerably enrich the language of local formulas into a higher-order logic.

Despite this added expressivity, we show that existing proof rules can be adapted; when needed, we recover the necessary assumptions through usual logical devices. We also provide new proof rules for the generalised constructs of our new logic – *e.g.* our logic supports unrestricted quantification in local formulas, which is handled by usual-looking sequent calculus rules, thanks to non-trivial observations on the probabilistic semantics. As for cryptographic rules, we significantly improve over the former meta-logic by taking corruptions into account, which was beyond the reach of all previous (base or meta) CCSA rules. We argue that the soundness arguments for our new rules are no more complex than the former lifting-based arguments, and less error-prone.

iii) A more pleasant framework for proofs. The generalisation of our logic to a higher-order language, and the generic recursive definition mechanism, provide a convenient language for the user to structure proofs in an abstract and modular way.

As a second, distinct contribution, we extended SQUIRREL to make it consistent with our higher-order semantics, and to implement our new proof rules.

We use this implementation to verify that all past case studies carried in the former meta-logic can be adapted to our new system, without any significant burden on the user. We also showcase the added expressivity on new case studies. First, taking advantage of the usual benefits of a higher-order language, we show that generic cryptographic reasoning techniques can be expressed in our system, in the form of a hybrid proof argument. Second, we show that our new cryptographic rules can be used to carry out protocol analyses where agents can be dynamically corrupted by the attacker.

Given the improved clarity and generality of our theoretical framework, and the experimental proof of its effectiveness to analyse both simple and advanced protocols, we believe that our new logic constitutes appropriate foundations for the SQUIRREL prover, and will facilitate its future development.

Related Work: As related work on verification of cryptographic designs has been discussed above, we now focus on other related lines of work in the literature.

There is a large body of work on logics with measure-theoretic quantifiers $\forall x.\phi$ asserting that ϕ holds in a probabilistic sense, with different meanings: "ϕ is true for almost all x" [17]; "ϕ is true for most x" [18]; or more quantitative measures (*e.g.* counting logics [19] or uncertainty logics [20], [21]). These works tackled a large panel of problems, *e.g.* the Curry-Howard correspondence between a counting logic and probabilistic functional programs [22]; establishing probability bounds using some knowledge with uncertainty logic [23]; or characterising complexity classes [17].

In this work, we are interested in a different kind of statements, of the form "ϕ is overwhelmingly true for all (families of) random variables x", which is a *qualitative* property on the asymptotic behaviour of ϕ. Moreover, our goal is to mechanise cryptographic reasoning, which explains our focus on building an implementable and usable proof system.

Another line of work [24], [25] consists in building the semantic foundations of statistical programming languages (*e.g.* [26]), which are languages mixing higher-order functions, continuous probabilities (*e.g.* on real numbers), and soft constraints. Accounting for these features requires complex domain-theoretic constructions, and building proof systems on top of such systems seems challenging (*e.g.* establishing that independent assignments commute is a non-trivial contribution of [24]). Our logic avoids this complexity by considering a fixed and finite probability space, which allows us to define our semantics and design a proof system for our logic with higher-order features.

Outline: The rest of the paper is structured as follows. We present the syntax and semantics of the logic in Section II, the core proof system in Section III. Information theoretic and cryptographic rules are presented in Section IV. We discuss our case studies and implementation in Section V. In Appendices A, B, and C we provide further details and definitions on typing rules, recursive definitions, and global formulas. We discuss in Appendix D how the former meta-logic can be encoded into our higher-order logic. We show the soundness of our proof system in Appendix E. Finally, we introduce and prove sound additional information theoretic and cryptographic rules in Appendices F and G.

II. HIGHER-ORDER CCSA LOGIC

We define a first-order logic over higher-order terms, where terms of sort bool will serve as local formulas. In order to recover the key features of the CCSA (meta-)logic, we equip our logic with a recursive definition mechanism, a specifically crafted probabilistic semantics, and we use tags (on types) and predicates (on terms) to constrain their interpretations where necessary. We introduce our terms with recursive definitions in Section II-A, explain in Section II-B how we equip them to obtain a suitable language of local formulas, before moving on to the global logic in Section II-C.

A. Terms

The terms of our logic are simply-typed λ-terms with a specific probabilistic semantics. We equip them with a strong notion of occurrence that helps justifying recursive definitions, and will be useful for cryptographic reasoning.

We assume a set of variables $\mathcal{X}$, and a set of base types $\mathbb{B}$ containing at least bool. Simple types, generated from base types $\tau_{\mathsf{b}} \in \mathbb{B}$ using the arrow constructor, will be denoted by the letter τ. Terms are then simply-typed λ-terms with booleans, with the following syntax, where $x \in \mathcal{X}$:

$$t ::= x \mid (t \; t') \mid \lambda(x : \tau).\, t \mid \text{if } t \text{ then } t' \text{ else } t'' \mid \text{true} \mid \text{false}$$

As usual, λ-terms are considered modulo alpha-renaming. We use $\equiv$ to denote syntactic equality of terms up to α-renaming, and let $\mathsf{fv}(t)$ denote the free variables of t. The typing rules are as expected. They are given in Fig. 6. We omit types when they are irrelevant or can be inferred.

We define propositional connectives over terms using conditionals and boolean constants. For instance, $t \wedge t'$ stands for (if t then t' else false). We call *local formulas* the terms of type bool, which can be constructed, *e.g.*, from boolean connectives and variables of return type bool.

An environment $\mathcal{E}$ is a finite set of *declarations* of the form $x : \tau$ and *definitions* of the form $x : \tau = t$. Both constructs bind x. We require that variables are bound at most once in $\mathcal{E}$. We let $\mathcal{E}(x) = \tau$ be the type of the declared or defined variable x in $\mathcal{E}$.

Given some $\mathcal{E}$, we define $\mathcal{E}^*$ as the environment obtained from $\mathcal{E}$ by replacing each definition $x : \tau = t$ with the declaration $x : \tau$. In other words, $\mathcal{E}^*$ is a usual typing environment. We require that an environment $\mathcal{E}$ is well-formed in the sense that for each definition $x : \tau = t$ in $\mathcal{E}$, we must have $\mathcal{E}^* \vdash t : \tau$. Note that this allows mutually recursive definitions, as illustrated next.

Example 4. $(a : \tau_1)$, $(b : \tau_2 = c \, a)$, $(c : \tau_1 \rightarrow \tau_2 = \lambda x.\, b)$ *is a (well-formed) environment.*

In the end, we will only be interested in environments whose recursive definitions are well-founded, and hence uniquely

$$
\begin{aligned}
\mathcal{ST}_{\mathcal{E}}(x) &\overset{\mathsf{def}}{=} \{(\epsilon, \mathsf{true}, x)\} && \text{when } (x{:}\tau) \in \mathcal{E} \text{ or } x \notin \mathcal{E} \\
\mathcal{ST}_{\mathcal{E}}(x) &\overset{\mathsf{def}}{=} \mathcal{ST}_{\mathcal{E}}(t) && \text{when } (x{:}\tau{=}t) \in \mathcal{E} \\
\mathcal{ST}_{\mathcal{E}}(t \; t') &\overset{\mathsf{def}}{=} \begin{cases} \mathcal{ST}_{\mathcal{E}}(t_0\{y \mapsto t'\}) & \text{when } t \equiv x \\ & \text{and } (x{:}\tau{=}\lambda y.\, t_0) \in \mathcal{E} \\ \{(\epsilon, \mathsf{true}, (t \; t'))\} \\ \cup\, \mathcal{ST}_{\mathcal{E}}(t) \cup \mathcal{ST}_{\mathcal{E}}(t') & \text{otherwise} \end{cases} \\
\mathcal{ST}_{\mathcal{E}}(\lambda(x : \tau).t) &\overset{\mathsf{def}}{=} \{(\epsilon, \mathsf{true}, \lambda(x : \tau).t)\} \\
&\quad\; \cup (x : \tau).\mathcal{ST}_{\mathcal{E}}(t) && \text{where } x \text{ is taken fresh} \\
\mathcal{ST}_{\mathcal{E}}(\text{if } \phi \text{ then } t_1 \text{ else } t_0) &\overset{\mathsf{def}}{=} \{(\epsilon, \mathsf{true}, \text{if } \phi \text{ then } t_1 \text{ else } t_0)\} \\
&\quad\; \cup\, \mathcal{ST}_{\mathcal{E}}(\phi) \\
&\quad\; \cup [\phi]\mathcal{ST}_{\mathcal{E}}(t_1) \cup [\neg\phi]\mathcal{ST}_{\mathcal{E}}(t_0)
\end{aligned}
$$

Figure 1. Generalised subterms.

defined. However, we will use a semantic notion of well-foundedness: this requires that we first introduce tools to analyse recursive occurrences, and a semantics.

1) Generalised subterms: We now define the set $\mathcal{ST}_{\mathcal{E}}(t)$ of the generalised subterms of t with respect to $\mathcal{E}$. Our goal is to capture not only syntactic subterms, but also subterms that may need to be evaluated to effectively compute t. The difficulty is that the evaluation of t depends on the random tape and on the model. To cover all intermediate terms that may need to be computed (for any random tape and model), our notion captures all terms that may appear through the expansion of definitions of $\mathcal{E}$. This will in general yield an infinite set of subterms. In order to determine which are relevant, we decorate each subterm t' with the surrounding conditionals ϕ. Finally, we must keep track of variables $\vec{\alpha}$ bound inside t that may occur in t'. Formally, $\mathcal{ST}_{\mathcal{E}}(t)$ will thus be a set of triples $(\vec{\alpha}, \phi, t')$, called *occurrences*, where $\vec{\alpha}$ is a sequence of typed variables, ϕ is a term of type bool in $\mathcal{E}, \vec{\alpha}$, and t' is a term well-typed in $\mathcal{E}, \vec{\alpha}$. We require that $\mathcal{E}, \vec{\alpha}$ is an environment, *i.e.* variables bound in $\vec{\alpha}$ are unbound in $\mathcal{E}$. For a set S of occurrences, we define:

$$
\begin{aligned}
[\phi]S &\overset{\mathsf{def}}{=} \{(\vec{\alpha}, \psi \wedge \phi, t) \mid (\vec{\alpha}, \psi, t) \in S\} \\
(x : \tau).S &\overset{\mathsf{def}}{=} \{((\vec{\alpha}, x : \tau), \psi, t) \mid (\vec{\alpha}, \psi, t) \in S\}
\end{aligned}
$$

Subterms $\mathcal{ST}_{\mathcal{E}}(t)$ are then defined as the smallest set satisfying the equations in Fig. 1.

This notion of generalised subterm will be useful to define well-founded recursive definitions but also, more importantly, to analyse the contents of recursive definitions when reasoning about freshness or cryptographic primitives.

2) Semantics of types: A *type structure* $\mathbb{M}$ associates to each base type $\tau_{\mathsf{b}} \in \mathbb{B}$ and each $\eta \in \mathbb{N}$ a non-empty interpretation domain $\mathbb{M}_{\tau_{\mathsf{b}}}(\eta)$, together with an injective mapping from this domain to bit-strings. We require that $\mathbb{M}_{\mathsf{bool}}(\eta) = \{0, 1\}$ for all $\mathbb{M}$ and η. While this base type is interpreted uniformly over η, that will not always be the case: a type of cryptographic keys will crucially have to be interpreted as sets of bit-strings of length (polynomial in) η.

The semantics $\llbracket \tau \rrbracket_{\mathbb{M}}^{\eta}$ of type τ in the type structure $\mathbb{M}$ at security parameter η is then given by:

$$\llbracket \tau_{\mathsf{b}} \rrbracket_{\mathbb{M}}^{\eta} \overset{\text{def}}{=} \mathbb{M}_{\tau_{\mathsf{b}}}(\eta) \qquad \llbracket \tau_1 \to \tau_2 \rrbracket_{\mathbb{M}}^{\eta} \overset{\text{def}}{=} \llbracket \tau_1 \rrbracket_{\mathbb{M}}^{\eta} \to \llbracket \tau_2 \rrbracket_{\mathbb{M}}^{\eta}$$

3) Semantics of terms: As in the CCSA logic, we endow terms with a probabilistic semantics, where probabilistic samplings depend on the security parameter η. Unlike in the CCSA logic, however, we consider random variables that are not necessarily computable.

A *term structure* $\mathbb{M}$ w.r.t. environment $\mathcal{E}$ (noted $\mathbb{M} : \mathcal{E}$) extends a type structure with the following elements:

- for each $\eta \in \mathbb{N}$, a set $\mathbb{T}_{\mathbb{M},\eta} = \mathbb{T}_{\mathbb{M},\eta}^{\mathsf{a}} \times \mathbb{T}_{\mathbb{M},\eta}^{\mathsf{h}}$, where $\mathbb{T}_{\mathbb{M},\eta}^{\mathsf{a}}$ (resp. $\mathbb{T}_{\mathbb{M},\eta}^{\mathsf{h}}$) is a *finite* set of bit-strings of the same length called *random tapes* – $\mathbb{T}_{\mathbb{M},\eta}^{\mathsf{a}}$ are random tapes for the adversary, while $\mathbb{T}_{\mathbb{M},\eta}^{\mathsf{h}}$ will be used for honest samplings;
- a partial mapping $\sigma_{\mathbb{M}}$ associating to each variable $\mathcal{E} \vdash x : \tau$ a family $(X_{\eta})_{\eta \in \mathbb{N}}$ of functions $X_{\eta} : \mathbb{T}_{\mathbb{M},\eta} \to \llbracket \tau \rrbracket_{\mathbb{M}}^{\eta}$. We let $\mathbb{M}(x)$ denote $\sigma_{\mathbb{M}}(x)$.

Given a term structure $\mathbb{M}$, $\mathbb{RV}_{\mathbb{M}}(\tau)$ denotes the set of all η-indexed families of random variables, *i.e.* functions from $\mathbb{T}_{\mathbb{M},\eta}$ to $\llbracket \tau \rrbracket_{\mathbb{M}}^{\eta}$. We use standard probability and measure theory definitions, which we recall in Appendix E-A.

Given a term structure $\mathbb{M} : \mathcal{E}$, x and τ such that $\mathcal{E} \vdash x : \tau$, and $X \in \mathbb{RV}_{\mathbb{M}}(\tau)$, we let $\mathbb{M}[x \mapsto X]$ be the structure which coincides with $\mathbb{M}$, except on x, which it maps to X.

For any $\mathcal{E} \vdash t : \tau$, for any $\mathbb{M} : \mathcal{E}$, and for any $\eta \in \mathbb{N}$ and $\rho \in \mathbb{T}_{\mathbb{M},\eta}$, we define the semantics $\llbracket t \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} \in \llbracket \tau \rrbracket_{\mathbb{M}}^{\eta}$ as follows:

$$\llbracket \text{if } t \text{ then } t_1 \text{ else } t_0 \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} \overset{\text{def}}{=} \llbracket t_i \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} \text{ when } \llbracket t \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = i$$

$$\llbracket \text{true} \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} \overset{\text{def}}{=} 1 \text{ and } \llbracket \text{false} \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} \overset{\text{def}}{=} 0$$

$$\llbracket x \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} \overset{\text{def}}{=} \mathbb{M}(x)(\eta)(\rho)$$

$$\llbracket t \, t' \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} \overset{\text{def}}{=} \llbracket t \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho}(\llbracket t' \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho})$$

$$\llbracket \lambda(x : \tau). t \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} \overset{\text{def}}{=} (a \in \llbracket \tau \rrbracket_{\mathbb{M}}^{\eta} \mapsto \llbracket t \rrbracket_{\mathbb{M}[x \mapsto \mathbb{1}_a^{\eta}]:(\mathcal{E},x:\tau)}^{\eta,\rho})$$

where $\mathbb{1}_a^{\eta}$ is the indexed family of functions such that:

- $\mathbb{1}_a^{\eta}(\eta)(\rho) = a$ for all $\rho \in \mathbb{T}_{\mathbb{M},\eta}$;
- $\mathbb{1}_a^{\eta}(\eta')(\rho')$ is some arbitrary value in $\llbracket \tau \rrbracket_{\mathbb{M}}^{\eta'}$ for all $\eta' \neq \eta$ and $\rho' \in \mathbb{T}_{\mathbb{M},\eta'}$.

We then define $\llbracket t \rrbracket_{\mathbb{M}:\mathcal{E}} \in \mathbb{RV}_{\mathbb{M}}(\tau)$ as $(\rho \mapsto \llbracket t \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho})_{\eta \in \mathbb{N}}$.

The reader may have noticed an artificial complexity in the semantics of abstractions: x is interpreted as an indexed family of random variables $\mathbb{1}_a^{\eta}$, but since η and ρ are fixed throughout the interpretation, only $\mathbb{1}_a^{\eta}(\eta)(\rho)$ matters. We chose this formulation for uniformity reasons: as we shall see in Section II-C, variables will truly be interpreted as families of random variables in the semantics of global formulas.

4) Recursive definitions: We can now define the notion of well-founded recursive definition, and prove the existence and uniqueness of their semantics. Given a term ϕ of the form $\wedge_i t_i$ and an environment $\mathcal{E}$, we define $\phi \setminus \{x_1, \ldots, x_n\}$ as the conjunction of all the t_i which do not have free variables in $\{x_1, \ldots, x_n\}$.

Definition 1. *We say that environment $\mathcal{E}$ is well-founded w.r.t. $\mathbb{M} : \mathcal{E}$ when:*

- *$\mathcal{E}$ is of the form $\mathcal{E}_0, \{x_i : \tau_i \to \tau_i' = \lambda x.t_i\}_{i \in [1;n]}$ where $\mathcal{E}_0$ contains only declarations;*
- *for each $i, j \in [1;n]$, x_i only occurs in $\lambda x.t_j$ in subterms of the form $(x_i \, t')$ with $\mathsf{fv}(t') \cap \{x_1, \ldots, x_n\} = \emptyset$;*
- *for each $\eta \in \mathbb{N}$, there exists a well-founded order $<$ over $\{(x_i, e) \mid i \in [1;n], e \in \llbracket \tau_i \rrbracket_{\mathbb{M}}^{\eta}\}$ such that, for any $i, j \in [1;n]$ and $(\vec{\alpha}, \phi, x_i \, t') \in \mathcal{ST}_{\mathcal{E}^*, x:\tau_j}(t_j)$, we have, for any $\mathbb{M}' : (\mathcal{E}, \vec{\alpha}, (x : \tau_j))$ extending $\mathbb{M}$ and any $\rho \in \mathbb{T}_{\mathbb{M},\eta}$ such that $\llbracket \phi \setminus \{x_1, \ldots, x_n\} \rrbracket_{\mathbb{M}':(\mathcal{E},\vec{\alpha},(x:\tau_j))}^{\eta,\rho} = 1$:*

$$(x_i, \llbracket t' \rrbracket_{\mathbb{M}':(\mathcal{E},\vec{\alpha},(x:\tau_j))}^{\eta,\rho}) < (x_j, \llbracket x \rrbracket_{\mathbb{M}':(\mathcal{E},\vec{\alpha},(x:\tau_j))}^{\eta,\rho}).$$

We require $<$ to be internalised as formulas, see Appendix B.

The definition above requires that recursive occurrences $(x_i \, t')$ are strictly smaller in the semantics. The notion is quite flexible thanks to its use of generalised occurrences, allowing it to take into account the conditionals that surround the recursive occurrence. For instance, assuming that the well-founded ordering is directly reflected as a function $(< : \tau_i \to \tau_i \to \text{bool}) \in \mathcal{E}_0$, we could write: $x_i = \lambda x.$ if $f \, x < x$ then $x_i \, (f \, x)$ else x. Note that, while we will later use our notion of generalised subterm taking recursive definitions into account, this feature is undesirable here, hence the use of $\mathcal{ST}_{\mathcal{E}^*, x:\tau_j}(t_j)$ with an environment without definitions.

Crucially, we constrain the free variables of t' to make sure that its interpretation does not rely on that of the x_i in $\mathbb{M}$: this interpretation is irrelevant as we intend to replace it by its recursively defined meaning – and we need the well-foundedness conditions to keep holding as we update the model. Similarly, we must make sure that the filtering on ϕ is insensitive to the interpretation of the x_i: rather than constraining the free variables, we only consider $\phi \setminus \{x_1, \ldots, x_n\}$ for expressivity reasons. This allows us, for instance, to write $x_i = \lambda x.$ if $f \, x < x \wedge x_j \, x$ then $x_i \, (f \, x)$ else x.

The following proposition formally justifies our notion of well-foundedness; it is proved in Appendix B.

Proposition 1. *Let $\mathcal{E}$ be an environment that is well-founded w.r.t. $\mathbb{M} : \mathcal{E}$. Let $\mathbb{M}_0$ be the restriction of $\mathbb{M}$ to the declarations of $\mathcal{E}$. There exists a unique term structure $\mathbb{M}_{\text{rec}} : \mathcal{E}$ extending $\mathbb{M}_0$ such that, for each $(x_i : \tau_i \to \tau_i' = \lambda x.t_i) \in \mathcal{E}$, $\llbracket x_i \rrbracket_{\mathbb{M}_{\text{rec}}:\mathcal{E}}^{\eta,\rho} = \llbracket \lambda x.t_i \rrbracket_{\mathbb{M}_{\text{rec}}:\mathcal{E}}^{\eta,\rho}$ for all η, ρ.*

Structures obtained from the previous result are called *models* of $\mathcal{E}$. An environment can only have models if it is well-founded.

B. Builtin Types and Function Symbols

In order to see our logic as an extension of the meta-logic of [12], [14], we need to make further assumptions on the models we consider, by assuming some built-ins, and restricting the interpretations of types and declared variables.

We first assume, for each type τ, that $\mathcal{E}$ contains two declarations for $\forall_{\tau}$ and $\exists_{\tau}$, both with type $(\tau \to \text{bool}) \to \text{bool}$. We require that any model $\mathbb{M} : \mathcal{E}$ interprets them with the

expected semantics, *e.g.* $[\![\forall_\tau]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho}(f) = 1$ when $f(a) = 1$ for all $a \in [\![\tau]\!]_\mathbb{M}^\eta$. We use quantifiers with the usual notation, *e.g.* $\forall(x : \tau). t$.

We also assume, for each type τ, a declaration $=_\tau: \tau \to \tau \to$ bool, written in infix, and require that $[\![t = t']\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = 1$ iff $[\![t]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = [\![t']\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho}$ in all models.

Finally, we assume for convenience that there exists a base type unit whose interpretation is always a singleton, and that $() :$ unit is declared in all environments.

1) Restricting the interpretations of types: We assume given, for each base type $\tau_\mathsf{b} \in \mathbb{B}$, a set of labels $\mathsf{label}(\tau_\mathsf{b})$ restricting τ_b's interpretation. For instance `finite` states that the type τ is finite; `polynomial` that it is of polynomial-size w.r.t. η; and `fixed` that $[\![\tau]\!]_\mathbb{M}^\eta = [\![\tau]\!]_\mathbb{M}^{\eta'}$ for any $\eta, \eta' \in \mathbb{N}$. The semantics for all labels are given in Appendix A. Labels are lifted to types in the natural way: for example, a type τ is finite if τ is a base type and `finite` $\in \mathsf{label}(\tau)$, or if $\tau = \tau_1 \to \tau_2$ where τ_1 and τ_2 are finite.

Example 5. *When modelling protocols, we will often use types* message, timestamp, index, *which are meant to represent respectively all bit-strings, time points along the protocol execution, and indices that identify protocol sessions. We will further assume that* $\mathsf{label}(\mathsf{index}) = \mathsf{label}(\mathsf{timestamp}) = \{\texttt{finite}, \texttt{fixed}\}$ *which is in line with the corresponding types in the former meta-logic.*

2) Restricting variable declarations: We consider a subset $\mathcal{N} \subseteq \mathcal{X}$ of variables which we call *names*. We only allow a name $\mathsf{n} \in \mathcal{N}$ to appear in a declaration with a type of the form $\tau_0 \to \tau_1$ where τ_0 is a finite type.

Models must interpret names in a precise way. Consider a model $\mathbb{M} : \mathcal{E}$ and a name $\mathsf{n} \in \mathcal{N}$ with $\mathsf{n} : \tau_0 \to \tau_1 \in \mathcal{E}$. We require that n is interpreted as a polynomial-time computable random sampling using random bits in ρ_h: there must exist a function $[\![\mathsf{n}]\!]_\mathbb{M}$ such that, for every $\eta \in \mathbb{N}$ and random tape $\rho = (\rho_\mathsf{a}, \rho_\mathsf{h}) \in \mathbb{T}_{\mathbb{M},\eta}$, $[\![\mathsf{n}\ t]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = [\![\mathsf{n}]\!]_\mathbb{M}(\eta, [\![t]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho})(\rho_\mathsf{h})$. Moreover, $[\![\mathsf{n}]\!]_\mathbb{M}$ must run in polynomial time (w.r.t. 1^η). We also require that the random variables

$$\rho_\mathsf{h} \mapsto [\![\mathsf{n}]\!]_\mathbb{M}(\eta, a)(\rho_\mathsf{h}) \quad \text{and} \quad \rho_\mathsf{h} \mapsto [\![\mathsf{n}']\!]_\mathbb{M}(\eta, a')(\rho_\mathsf{h})$$

be independent random samplings whenever $(\mathsf{n}, a) \neq (\mathsf{n}', a')$ – they must extract different random bits from ρ_h. Moreover they must extract the same number of bits and follow the same *distribution* when n and n' have the same output type (*i.e.* $\mathcal{E}(\mathsf{n}) = \_ \to \tau$ and $\mathcal{E}(\mathsf{n}') = \_ \to \tau$).

Note that an environment contains a finite number of name symbols, each taking an argument in a finite type. Moreover, a name value requires at most a finite number of bits from ρ_h. Thus, for any type model, there always exists a term model with large enough honest tapes for names to draw their randomness, while satisfying all of the above requirements – recall that tapes are finite bit-strings.

Example 6. *We replicate in our logic the meta-logic modelling of the Hash Lock protocol from Example 3, using the types of Example 5. The keyed hash function and pairing operation are modelled by declarations using a specific type for hashing keys:*

$$\mathsf{h}: \mathsf{message} \to \mathsf{key} \to \mathsf{message}$$
$$\langle\cdot,\cdot\rangle: \mathsf{message} \to \mathsf{message} \to \mathsf{message}$$

To model many sessions of the tag, we consider a declaration $\mathsf{T} : \mathsf{index} \to \mathsf{timestamp}$ *associating to each session index the time point when it is executed. The tag's key and each session's nonce will be modelled by names* $\mathsf{k} : \mathsf{unit} \to \mathsf{key}$ *and* $\mathsf{n} : \mathsf{index} \to \mathsf{message}$.

We declare init : timestamp, *and let* pred : timestamp $\to$ timestamp *be the predecessor w.r.t. the scheduling order, and restrict our attention to models where* pred *induces a total order on timestamps, for which* init *is minimal. We assume that every time point is either* init *or some* $(\mathsf{T}\ i)$. *We finally make use of a construction* matchwith *to perform case distinctions in terms – this may be encoded, again, by means of appropriate declarations and model restrictions.*

The messages input and output at a particular time point, and the sequence of messages previously output (the frame), are modelled as three mutually recursive function definitions – all of type (timestamp $\to$ message) *– where* d *is an arbitrary default term and* att *would later be constrained to represent an adversarial computation as in Example 3:*

$$\mathsf{input} = \lambda t.\ \mathsf{match}\ t\ \mathsf{with}\ \mathsf{init} \to d$$
$$| \_ \to \mathsf{att}\ (\mathsf{frame}\ (\mathsf{pred}\ t))$$

$$\mathsf{output} = \lambda t.\ \mathsf{match}\ t\ \mathsf{with}\ \mathsf{init} \to d$$
$$| \mathsf{T}\ i \to \langle\mathsf{n}(i), \mathsf{h}(\langle\mathsf{input}\ (\mathsf{T}\ i), \mathsf{n}(i)\rangle, \mathsf{k}\ ())\rangle$$

$$\mathsf{frame} = \lambda t.\ \mathsf{match}\ t\ \mathsf{with}\ \mathsf{init} \to d$$
$$| \_ \to \langle\mathsf{frame}\ (\mathsf{pred}\ t), \mathsf{output}\ t\rangle$$

C. Global Formulas

Global formulas are first-order formulas[2], interpreted in first-order structures of domain $\mathbb{RV}_\mathbb{M}(\tau)$ for each τ. The syntax is standard, except for decorations that distinguish global connectives from local ones, using some predicate variables $p \in \mathcal{P}$:

$$F ::= F_1 \tilde{\wedge} F_2 \mid F_1 \tilde{\vee} F_2 \mid F_1 \tilde{\Rightarrow} F_2 \mid \tilde{\neg} F$$
$$\mid p(t_1, \ldots, t_n) \mid \tilde{\forall}(x : \tau).\ F \mid \tilde{\exists}(x : \tau).\ F$$

We interpret these formulas in structures that extend a term model $\mathbb{M} : \mathcal{E}$ by providing, for each predicate symbol p taking arguments of types $(\tau_i)_{i \in [1;n]}$, an interpretation $[\![p]\!]_{\mathbb{M}:\mathcal{E}} \subseteq \Pi_{i \in [1;n]}\mathbb{RV}_\mathbb{M}(\tau_i)$. The semantics of formulas is then derived in a standard way. For instance, $[\![\tilde{\forall}(x : \tau).F]\!]_{\mathbb{M}:\mathcal{E}} = 1$ iff $[\![F]\!]_{\mathbb{M}[x \mapsto a]:(\mathcal{E},x:\tau)} = 1$ is true for all $a \in \mathbb{RV}_\mathbb{M}(\tau)$.

We write $\mathbb{M} : \mathcal{E} \models F$ whenever $[\![F]\!]_{\mathbb{M}:\mathcal{E}} = 1$, which we shorten as $\mathbb{M} \models F$ when $\mathcal{E}$ is clear from context. We say that a formula F is valid (w.r.t. the environment $\mathcal{E}$) if $\mathbb{M} : \mathcal{E} \models F$ for every model $\mathbb{M}$ of $\mathcal{E}$.

In practice, we work with specific predicate symbols, with a fixed semantics. We shall consider the following atoms and

---

[2] There would be no difficulty in defining a higher-order global logic, but we choose first-order for simplicity. It does not limit us much in practice.

$$\llbracket [\phi] \rrbracket_{\mathsf{M}:\mathcal{E}} \stackrel{\text{def}}{=} \Pr_{\rho \in \mathbb{T}_{\mathsf{M},\eta}} \left(\llbracket \phi \rrbracket_{\mathsf{M}:\mathcal{E}}^{\eta,\rho} \right) \in \mathsf{ow}(\eta)$$

$$\llbracket \vec{t}_1 \sim \vec{t}_2 \rrbracket_{\mathsf{M}:\mathcal{E}} \stackrel{\text{def}}{=} \text{for all } \mathcal{A} \in \mathsf{PPTMs},$$
$$\mathsf{Adv}_{\mathsf{M}:\mathcal{E}}^{\eta}(\mathcal{A} : \vec{t}_1 \sim \vec{t}_2) \in \mathsf{negl}(\eta)$$

$$\llbracket \mathsf{const}(t) \rrbracket_{\mathsf{M}:\mathcal{E}} \stackrel{\text{def}}{=} \text{exists } c \in (\bigcap_{\eta \in \mathbb{N}} \llbracket \tau \rrbracket_{\mathsf{M}}^{\eta}),$$
$$\llbracket t \rrbracket_{\mathsf{M}:\mathcal{E}}^{\eta,\rho} = c \text{ for any } \eta \in \mathbb{N}, \rho \in \mathbb{T}_{\mathsf{M},\eta}$$

$$\llbracket \mathsf{adv}(t) \rrbracket_{\mathsf{M}:\mathcal{E}} \stackrel{\text{def}}{=} \text{exists } \mathcal{A} \in \mathsf{PPTMs} \text{ s.t.}$$
$$\text{for all } \eta \in \mathbb{N}, \rho = (\rho_{\mathsf{a}}, \rho_{\mathsf{h}}) \in \mathbb{T}_{\mathsf{M},\eta},$$
$$\llbracket t \rrbracket_{\mathsf{M}:\mathcal{E}}^{\eta,\rho} = A(1^{\eta}, \rho_{\mathsf{a}})$$

In the $\mathsf{const}(t)$ case, τ is the type of t in $\mathcal{E}$.

Figure 2. Semantics of global formulas.

intuitive semantics: an overwhelming truth atom $[\phi]$, where ϕ is a local formula, which states that ϕ holds overwhelmingly; an equivalence atom $\vec{t}_1 \sim \vec{t}_2$, where $\vec{t}_1$ and $\vec{t}_1$ are same-length vectors of local terms, which states that the distributions induced by $\vec{t}_1$ and $\vec{t}_2$ are indistinguishable by any PPTM adversary; an atom $\mathsf{const}(t)$ which states that t is a constant value; an atom $\mathsf{adv}(t)$ expressing the fact that t can be computed by a PPTM using only the adversarial part of the random tape. Figure 6 includes typing rules for global atoms and formulas. Note that this list of predicates is not exhaustive: our logic and proof rules do accommodate extra predicate symbols.

The formal semantics of these specific predicates is given in Fig. 2. We illustrate it on simple predicates, before explaining the semantics of equivalence atoms.

Example 7. *Given the fixed interpretation of $\forall_\tau$, the atom $\mathsf{adv}(\forall_\tau)$ is valid whenever τ is polynomial, and $\mathsf{const}(\forall_\tau)$ is valid whenever τ is fixed.*

In order to understand the semantics of the indistinguishability predicate, we first need to: i) define what is the class of PPTM adversaries against equivalence atoms $\vec{t}_1 \sim \vec{t}_2$; ii) quickly describe our execution model, in particular how higher-order terms are handled.

1) Adversaries: Let $\vec{t}_1 \sim \vec{t}_2$ be a well-typed equivalence in $\mathcal{E}$, where $\vec{t}_1, \vec{t}_2$ are two vectors of terms of length L. A machine $\mathcal{A} \in \mathsf{PPTMs}$ against game $\vec{t}_1 \sim \vec{t}_2$ is a Turing machine such that: i) $\mathcal{A}$ has L *input tapes*, and a special *randomness input tape* for the tape of random bits ρ_{a}; ii) $\mathcal{A}$ has a special *query tape* used to perform queries to functions for which it has a handle, where each query is composed of a function handle and the query arguments, adequately encoded as bit-strings; iii) $\mathcal{A}$'s randomness only comes from ρ_{a}, *i.e.* once the random tape ρ_{a} is fixed, $\mathcal{A}$ is a deterministic machine; and iv) $\mathcal{A}$ has a fixed but arbitrary number of working tapes, and runs in polynomial-time w.r.t. the length of its inputs, *ignoring* the length of the random tapes.

2) Execution model: We now sketch our execution model by quickly describing how the adversary is given access to higher-order terms (*i.e.* terms of types $\tau_1 \to \tau_2$).

In an equivalence between sequences of terms $\vec{t}_1 \sim \vec{t}_2$ with some higher-order terms, we allow the adversary to make queries to these terms using its query tape. Roughly, all functions available to the adversary have an associated short bit-string handle (an identifier). Then, the execution environment maintains a map from handles to functions, which it lets the adversary query on arbitrary inputs.

If f is a function whose return type is itself a function (*i.e.* f is of type $\tau_1 \to (\tau_2 \to \tau_3)$), then whenever $\mathcal{A}$ calls f, through its handle h_f, on some input a, the execution environment computes the function $g = f(a)$ of type $\tau_2 \to \tau_3$, binds g to the shortest fresh handle h_g available, and returns h_g to $\mathcal{A}$.

Let s_1 and s_2 be the i-th terms of, respectively, $\vec{t}_1$ and $\vec{t}_2$, and τ their type. Then, given the security parameter η and the random tape $\rho \in \mathbb{T}_{\mathsf{M},\eta}$, $\mathcal{A}$ receives on input tape i a bit-string which depends on τ:

- if τ is a base type τ_{b}, the bit-string encoding of $\llbracket s_i \rrbracket_{\mathsf{M}:\mathcal{E}}^{\eta,\rho}$ (recall that base-types come with an encoding);
- if τ is an arrow type $\tau_1 \to \tau_2$, a bit-string handle h to the function $\llbracket s_i \rrbracket_{\mathsf{M}:\mathcal{E}}^{\eta,\rho}$ – and the execution environment now binds the handle h to $\llbracket s_i \rrbracket_{\mathsf{M}:\mathcal{E}}^{\eta,\rho}$;

Note that handles are similar to oracles: the associated functions may be uncomputable. This is in contrast with *closures* that the machine may compute and return to encode functions that it can compute — see Appendix G-A for details.

3) Semantics: The advantage $\mathsf{Adv}_{\mathsf{M}:\mathcal{E}}^{\eta}(\mathcal{A} : \vec{t}_1 \sim \vec{t}_2)$ of $\mathcal{A} \in$ PPTMs against the *left-or-right* game $\vec{t}_1 \sim \vec{t}_2$ is

$$\left| \Pr_{\rho \in \mathbb{T}_{\mathsf{M},\eta}} \left(\mathcal{A}(1^{\eta}, \llbracket \vec{t}_1 \rrbracket_{\mathsf{M}:\mathcal{E}}^{\eta,\rho}, \rho_{\mathsf{a}}) \right) - \Pr_{\rho} \left(\mathcal{A}(1^{\eta}, \llbracket \vec{t}_2 \rrbracket_{\mathsf{M}:\mathcal{E}}^{\eta,\rho}, \rho_{\mathsf{a}}) \right) \right|$$

where ρ_{a} is the adversarial part of the pair of random tapes ρ. Here, the set of all tapes $\mathbb{T}_{\mathsf{M},\eta}$ is equipped with the discrete σ-algebra (*i.e.* all subsets are measurable) and the uniform probability measure (recall that $\mathbb{T}_{\mathsf{M},\eta}$ is always finite).

Since $\mathbb{T}_{\mathsf{M},\eta}$ is equipped with the discrete probability measure, any function from $\mathbb{T}_{\mathsf{M},\eta}$ to a measurable space is measurable. In particular, $(\rho \in \mathbb{T}_{\mathsf{M},\eta} \mapsto \llbracket \phi \rrbracket_{\mathsf{M}:\mathcal{E}}^{\eta,\rho})_{\eta \in \mathbb{N}}$ is a (family of) random variables. This is essential in order to define the semantics of overwhelming truth atoms (but also equivalence atoms). Without this, it may not be possible to interpret $[\forall x.\phi]$, which corresponds to a potentially uncountable intersection. The simplicity of this probabilistic setting is not a limitation for our cryptographic application: indeed, we ultimately only consider PTIME computations that draw a bounded amount of randomness from random tapes. The interested reader will find more details in Appendix C.

Example 8. *Assume $\mathcal{E}$ declares two names $\mathsf{n} : \mathsf{unit} \to \tau$ and $\mathsf{n}' : \mathsf{unit} \to \tau$. Given our constraints on the interpretation of names, the formula $(\mathsf{n} () \sim \mathsf{n}' ())$ is valid. Moreover, if we assume that τ is large, i.e. that the probability that name samplings in τ draw any given particular value is negligible, then $[\mathsf{n} () \neq \mathsf{n}' ()]$ is valid by independence of the two name samplings (see Appendix A for details).*

In addition, we have $\mathsf{n} () \sim \mathsf{n} ()$, as computational indistinguishability is an equivalence relation. Note however that while $\sim$ is an equivalence relation, it is not a congruence:

it is not stable by context application. Indeed, although n () ∼ n () *and* n () ∼ n′ () *are valid,* n (), n () ∼ n (), n′ () *is in general not. It is usually possible to distinguish the left from the right, since on the left the adversary is always provided with two identical bit-strings, while on the right,* $[\![$n $()]\!]$ *and* $[\![$n′ $()]\!]$ *will be different with a non-negligible probability (except in degenerate cases, where the names* n *and* n′ *put all their weight on a single value of* $[\![\tau]\!]$ *).*

III. PROOF SYSTEM

We have a two-level logic, with an outer global logic and an inner local logic. This structure is reflected in proof systems, which feature two kinds of judgements: a *global judgement* $\mathcal{E}; \Theta \vdash F$ comprises a set of global formulas Θ as hypotheses, and a global formula F as goal; and a *local judgement* $\mathcal{E}; \Theta; \Gamma \vdash \phi$ comprises a set of global hypotheses Θ, a set of local formulas Γ as local hypotheses, and a local formula ϕ as goal.

A global judgement $\mathcal{E}; \Theta \vdash F$ is valid iff $\tilde{\wedge}\Theta \tilde{\Rightarrow} F$ is satisfied by all models of $\mathcal{E}$. Similarly, a local judgement $\mathcal{E}; \Theta; \Gamma \vdash \phi$ is valid iff $\tilde{\wedge}\Theta \tilde{\Rightarrow} [\wedge \Gamma \Rightarrow \phi]$ is satisfied by all models of $\mathcal{E}$.

Standard rules: Global judgements are essentially standard first-order classical sequents, for which we consider only a restricted class of interpretation domains. We can thus equip them with the usual rules of classical sequent calculus.

More interestingly, local judgements can also be equipped with standard reasoning rules. Here, it must be noted that since local judgements have a global and a local set of hypotheses, all standard left rules have a global and local variant, *e.g.* we have two left "and" rules, one for $\wedge$ and one for $\tilde{\wedge}$. Our logic also features rules allowing to go back on forth between the two kinds of sequents. Our standard rules are described in Fig. 7.

Local vs global quantification: The reader may have noticed that while our logic features two kinds of universal quantifiers (local and global), our judgements do not make any distinctions between locally and globally universally quantified variables: both kinds are recorded in the same way in the environment $\mathcal{E}$.

This surprising feature is explained by the observation that quantifications "commute" with $[\cdot]$: we show next that $[\forall(x : \tau). \phi]$ and $\tilde{\forall}(x : \tau). [\phi]$ are logically equivalent; the same holds for existential quantifiers, with the same proof.

Proposition 2. *Let* $\mathcal{E}$ *be a well-formed environment and* ϕ *be a local formula such that* $\mathcal{E} \vdash \forall(x : \tau). \phi :$ bool. *For every model* $\mathbb{M}$ *of* $\mathcal{E}$, *we have:*

$$\mathbb{M} : \mathcal{E} \models [\forall(x : \tau). \phi] \qquad iff \qquad \mathbb{M} : \mathcal{E} \models \tilde{\forall}(x : \tau). [\phi]$$

Proof (sketch, see details in Appendix E-C). $\Rightarrow$ is clear. For $\Leftarrow$, we build a (sequence of) random variable(s) A such that

$$[\![\phi]\!]^{\eta,\rho}_{\mathbb{M}[x \mapsto A]:\mathcal{E},x:\tau} = [\![\forall(x : \tau). \phi]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}} \qquad (1)$$

We do this by having $A(\eta)(\rho)$ choose, if it exists, an arbitrary element $a \in [\![\tau]\!]^{\eta}_{\mathbb{M}}$ violating ϕ, *i.e.* such that

$$[\![\phi]\!]^{\eta,\rho}_{\mathbb{M}[x \mapsto \mathbb{1}^{\eta}_a]:\mathcal{E},x:\tau} = 0.$$

If no such element exists, $A(\eta)(\rho)$ takes an arbitrary value in $[\![\tau]\!]^{\eta}_{\mathbb{M}}$ We can check that A indeed verifies Eq. (1). We conclude using the fact that $\tilde{\forall}(x : \tau). [\phi]$ implies that

$$\mathrm{Pr}_{\rho \in \mathbb{T}_{\mathbb{M},\eta}} \left([\![\phi]\!]^{\eta,\rho}_{\mathbb{M}[x \mapsto A]:\mathcal{E},x:\tau} \right) \in \mathsf{ow}(\eta) \qquad \square$$

Remark 1. *A key point of the* $\Leftarrow$ *direction is that* A *is a sequence of random variables. This is made obvious by our semantics, since all functions from* $\mathbb{T}_{\mathbb{M},\eta}$ *to* $\{0, 1\}$ *are random variables w.r.t. the discrete* σ*-algebra on* $\mathbb{T}_{\mathbb{M},\eta}$ *(thanks to* $\mathbb{T}_{\mathbb{M},\eta}$*'s finiteness). This is why the finiteness assumption is a crucial ingredient of our framework.*

We can now give our right quantifiers rules, which have an identical treatment of local and global variables.

$$\begin{array}{cc}
\text{L.R-}\forall & \text{G.R-}\tilde{\forall} \\
\dfrac{\mathcal{E}, x : \tau; \Theta; \Gamma \vdash \phi}{\mathcal{E}; \Theta; \Gamma \vdash \forall(x : \tau). \phi} & \dfrac{\mathcal{E}, x : \tau; \Theta \vdash F}{\mathcal{E}; \Theta \vdash \tilde{\forall}(x : \tau). F}
\end{array}$$

This does not mean that global and local quantification are identical, only that they coincide when in front of an atom $[\phi]$. We show in Appendix E-C the following.

Proposition 3. *Rules* L.R-$\forall$ *and* G.R-$\tilde{\forall}$ *are sound.*

The issue about global and local quantification does not arise in earlier works on the meta-logic. Indeed, in [14], local quantification only applies to two specific types (timepoints and index parameters). Terms of these types are only built with a restricted syntax, and have a particular ad-hoc semantics forcing their constancy. We lift these restrictions, which limited the logic's expressiveness and modelling power.

Local vs global connectives: Our proof system has rules exploiting the links between the two logics. For example, $[\phi \wedge \psi]$ implies $[\phi] \tilde{\wedge} [\psi]$: indeed, if the conjunction of ϕ and ψ is overwhelmingly true, then ϕ must be overwhelmingly true, and ψ must be overwhelmingly true.

This kind of phenomenon does not occur for all connectives, *e.g.* $[\phi \vee \psi]$ does not generally imply $[\phi] \tilde{\vee} [\psi]$[3]. Indeed, take ϕ to be a coin toss, and $\psi := \neg\phi$: then $\phi \vee \psi$ is always true, but neither ϕ nor ψ are overwhelmingly true. But the property does hold when ϕ (or ψ) is constant.

To that end, [14] introduced the ad-hoc notion of *pure timestamp formula*, which is a side-condition exploiting syntactic restrictions, to guarantee that terms of some specific types are always constant. We replace the *pure trace formula* side-condition of [14] – which no longer makes sense, since we removed all ad-hoc syntactic restrictions – with a proof obligation. Constancy is established through the global logic predicate const($\cdot$), which can itself be derived using general proof rules.

Fig. 3 presents a selected set of left and right rules relating local and global connectives. These rules are proved sound in Appendix E-E.

---

[3]It may be surprising that $\exists$ and $\vee$ behave differently in that respect. We refer the reader to Appendix E-C for a discussion on that point.

$$\frac{\text{G.R-LOC:}\rightarrow}{\mathcal{E};\Theta,[\phi]\vdash[\psi] \quad \mathcal{E};\Theta\vdash\mathsf{const}(\phi)}{\mathcal{E};\Theta\vdash[\phi\Rightarrow\psi]}$$

$$\frac{\text{G.L-LOC:}\vee}{\mathcal{E};\Theta,[\phi]\vdash F \quad \mathcal{E};\Theta,[\psi]\vdash F \quad \mathcal{E};\Theta\vdash\mathsf{const}(\phi)\tilde{\vee}\mathsf{const}(\psi)}{\mathcal{E};\Theta,[\phi\vee\psi]\vdash F}$$

Figure 3. Selected rules relating local and global connectives.

$$\frac{\text{G.CONST:APP}}{\mathcal{E};\Theta\vdash\mathsf{const}(t) \quad \mathcal{E};\Theta\vdash\mathsf{const}(t')}{\mathcal{E};\Theta\vdash\mathsf{const}(t\ t')}$$

$$\frac{\text{G.CONST:QUANT}}{\mathcal{Q}\in\{\lambda,\forall,\exists\} \quad \texttt{fixed}\in\mathsf{label}(\tau) \quad \mathcal{E};\Theta\vdash\tilde{\forall}(x:\tau).\,\mathsf{const}(x)\tilde{\Rightarrow}\mathsf{const}(t)}{\mathcal{E};\Theta\vdash\mathsf{const}(\mathcal{Q}(x:\tau).\,t)}$$

Alternative right $\forall$-introduction rule:

$$\frac{\text{L.R-}\forall\text{-CONST}}{\{\texttt{fixed},\texttt{finite}\}\subseteq\mathsf{label}(\tau) \quad \mathcal{E},x:\tau;\Theta,\mathsf{const}(x);\Gamma\vdash\phi}{\mathcal{E};\Theta;\Gamma\vdash\forall(x:\tau).\phi}$$

Figure 4. Selected rules for $\mathsf{const}(\cdot)$.

A. Constant Terms and Restricting Types

We designed a simple set of rules to establish that a term represents a constant computation (Fig. 4). The first two are rather simple: G.CONST:APP states that $t\ t'$ is constant whenever t and t' are constant; and G.CONST:QUANT that a (local) binder $\mathcal{Q}(x:\tau).\,t$ over a fixed type τ is constant if its body t is whenever the bound variable x is constant.

The rule L.R-$\forall$-CONST is more interesting. This is a stronger version of the right $\forall$ rule L.R-$\forall$, which allows, when extending $\mathcal{E}$ with a new bound variable $(x:\tau)$, to add a global constant hypothesis $\mathsf{const}(x)$ at the same time. This rule applies only with the proviso that τ is *finite and fixed*, *i.e.* its interpretation $[\![\tau]\!]_{\mathbb{M}}^{\eta}$ is a finite set independent from η.

This allows us to prove a stronger property than Proposition 2: for fixed finite types, local quantification is equivalent to global quantification over *constant* values.

Proposition 4. *Let $\mathcal{E}$ be a well-formed environment and ϕ a local formula such that $\mathcal{E}\vdash\forall(x:\tau).\phi:\mathsf{bool}$ and $\{\texttt{fixed},\texttt{finite}\}\subseteq\mathsf{label}(\tau)$. For every model $\mathbb{M}:\mathcal{E}$:*

$$\mathbb{M}:\mathcal{E}\models[\forall(x:\tau).\phi] \quad \textit{iff} \quad \mathbb{M}:\mathcal{E}\models\tilde{\forall}(x:\tau).\,\mathsf{const}(x)\tilde{\Rightarrow}[\phi]$$

Proof (sketch, details in Appendix E-C) . Thanks to Proposition 2, we only have to prove that

$$\mathbb{M}:\mathcal{E}\models\tilde{\forall}(x:\tau).\,[\phi] \quad \textit{iff} \quad \mathbb{M}:\mathcal{E}\models\tilde{\forall}(x:\tau).\,\mathsf{const}(x)\tilde{\Rightarrow}[\phi]$$

The $\Rightarrow$ direction is obvious. For the $\Leftarrow$ direction, we use the fact that the conjunction $(\bigcap_{1\le j\le N}X_{\eta}^{j})_{\eta\in\mathbb{N}}$ of a collection of N (finite, independent from η) η-indexed families of boolean random variables $(X_{\eta}^{1})_{\eta\in\mathbb{N}},\dots,(X_{\eta}^{N})_{\eta\in\mathbb{N}}$ is overwhelmingly true whenever the random variables $(X_{\eta}^{1})_{\eta\in\mathbb{N}},\dots,(X_{\eta}^{N})_{\eta\in\mathbb{N}}$ are all overwhelmingly true.

By taking $N=|[\![\tau]\!]_{\mathbb{M}}^{\eta}|$ and one constant random variable $(X_{\eta}^{a})_{\eta\in\mathbb{N}}$ for each $a\in[\![\tau]\!]_{\mathbb{M}}^{\eta}$, we get that $\Pr([\![\phi]\!]_{\mathbb{M}[x\mapsto A]:(\mathcal{E},x:\tau)}^{\eta,\rho})\in\mathsf{ow}(\eta)$ for any family of random variables $A\in\mathbb{RV}_{\mathbb{M}}(\tau)$. $\qquad\square$

Proposition 5. *The rules in Fig. 4 are sound.*

$$\frac{\text{G.EQUIV:FA-ADV}}{\mathcal{E};\Theta\vdash\vec{u}_l\sim\vec{u}_r \quad \mathcal{E};\Theta\vdash\mathsf{adv}(t)}{\mathcal{E};\Theta\vdash\vec{u}_l,t\sim\vec{u}_r,t}$$

$$\frac{\text{G.EQUIV:FA-APP}}{\mathcal{E};\Theta\vdash\vec{u}_l,t_l,t_l'\sim\vec{u}_r,t_r,t_r'}{\mathcal{E};\Theta\vdash\vec{u}_l,(t_l\ t_l')\sim\vec{u}_r,(t_r\ t_r')}$$

$$\frac{\text{G.EQUIV:FA}_{\lambda}\text{-APP}}{\mathcal{E};\Theta\vdash\begin{array}{l}\vec{u}_l,(\lambda(x:\tau).\,t_l),(\lambda(x:\tau).\,t_l')\\\sim\ \vec{u}_r,(\lambda(x:\tau).\,t_r),(\lambda(x:\tau).\,t_r')\end{array}}{\mathcal{E};\Theta\vdash\vec{u}_l,\lambda(x:\tau).\,(t_l\ t_l')\sim\vec{u}_r,\lambda(x:\tau).\,(t_r\ t_r')}$$

Figure 5. Selected indistinguishability rules.

The proof is given in Appendix E-F.

These rules have been designed with pragmatic considerations in mind. Our goal is to be able to prove the constant proof obligations coming from our rules, not to build a logic fully capturing constant computations. We therefore did not investigate this aspect of the logic further: *e.g.* questions of completeness are out-of-scope of this work. Instead, we justify the usefulness of our rules through practice, by showing that they are sufficient for all existing and new case-studies conducted in SQUIRREL.

B. Indistinguishability Rules

We present a selected set of rules of our logic related to the indistinguishability predicate $\sim$ in Fig. 5. The soundness of these rules must be established through cryptographic reductions [27]. Roughly, given a PPTM adversary $\mathcal{A}$ against the conclusion of the rule, we must build a PPTM $\mathcal{B}$ against the premise, such that $\mathcal{A}$'s advantage is upper-bounded by $\mathcal{B}$'s advantage. Then, since all efficient adversaries against the premise have a negligible advantage, it follows that all efficient adversaries against the conclusion have a negligible advantage. Usually, the adversary $\mathcal{B}$ performs some computations and then calls $\mathcal{A}$ as a black-box. Note that if the conclusion contains higher-order terms (*e.g.* $\lambda(x:\tau).\,t$), then the adversary $\mathcal{B}$ must answer $\mathcal{A}$'s queries to the corresponding oracle (*e.g.* $t\{x\mapsto a\}$ for some arbitrary input $a\in[\![\tau]\!]$ from $\mathcal{A}$). Of course, $\mathcal{B}$ can use its own oracles to do so.

Rule G.EQUIV:FA-ADV on Fig. 5 states that a term t that can be computed by the adversary can be safely removed from an indistinguishability. G.EQUIV:FA-APP allows to decompose a function application, by letting the adversary compute it. G.EQUIV:FA$_{\lambda}$-APP generalises the previous rule by applying it under an abstraction – indeed, $\mathcal{B}$ only has to call $\mathcal{A}$, answering $\mathcal{A}$'s calls to its oracle using the two oracles available to $\mathcal{B}$.

These rules are all adaptations or extensions of existing rules from [8], [12], [14] to our higher-order logic. In particular, previous work did not allow higher-order terms, and consequently did not have to deal with oracle simulations. A more detailed set of rules, including some usual [8] structural rules, can be found in Fig. 8. We also present additional rules dedicated to oracles in Appendix E-G.

IV. ADVANCED RULES

We now present the more advanced rules of our logic, that rely on information theoretic arguments or capture cryptographic hardness assumptions.

A. Freshness in Equivalences

In this section, we present a rule exploiting the freshness of a name in an equivalence formula. Roughly, our rule states that $\vec{u}, \mathsf{n}\ t \sim \vec{u}, \mathsf{n}_{\mathsf{fresh}}()$ where $\mathsf{n}_{\mathsf{fresh}}$ is a fresh name. This rule is not unconditionally true: *e.g.* as seen in Example 8, $\mathsf{n}\ (), \mathsf{n}\ () \sim \mathsf{n}\ (), \mathsf{n}'\ ()$ is not valid. To be valid, we add a premise requiring that index t of name n is never read in $\vec{u}$, under which condition, $[\![\vec{u}]\!]$ and $[\![\mathsf{n}\ t]\!]$ are independent.

Before formalising the independence condition, we present a result on $\mathcal{ST}_{\mathcal{E}}(\cdot)$ stating that the semantics of a term t w.r.t. some model $\mathbb{M} : \mathcal{E}$ and two different random tapes ρ_1 and ρ_2 is identical, as long as the interpretation by $\mathbb{M}$ of any *declared* variables in $\mathcal{E}$ coincide on ρ_1 and ρ_2.

Proposition 6. *Let t be a term that is well-typed in $\mathcal{E}$, and in eta-long form. Let $\mathbb{M}$ be models of $\mathcal{E}$, $\eta \in \mathbb{N}$ and $\rho_1, \rho_2 \in \mathbb{T}_{\mathbb{M},\eta}$. We have $[\![t]\!]^{\eta,\rho_1}_{\mathbb{M}:\mathcal{E}} = [\![t]\!]^{\eta,\rho_2}_{\mathbb{M}:\mathcal{E}}$ provided*

$$\mathbb{M}(x)(\eta)(\rho_1)(a) = \mathbb{M}(x)(\eta)(\rho_2)(a) \quad \text{with } a \stackrel{\mathsf{def}}{=} [\![\vec{u}]\!]^{\eta,\rho_1}_{\mathbb{M}':\mathcal{E},\vec{\alpha}}$$

for all $(\vec{\alpha}, \phi, (x\ \vec{u})) \in \mathcal{ST}_{\mathcal{E}}(t)$ such that: x is a variable declaration bound in $\mathcal{E}$ (not in $\vec{\alpha}$); $\mathbb{M}'$ extends $\mathbb{M}$ into a model of $(\mathcal{E}, \vec{\alpha})$; and $[\![\phi]\!]^{\eta,\rho_1}_{\mathbb{M}':\mathcal{E},\vec{\alpha}} = 1$.

Example 9. *Assume a type int modelling integers, a constant $0 : \mathsf{int}$ and a predecessor function $\mathsf{pred} : \mathsf{int} \to \mathsf{int}$, restricted to have the expected interpretations, and let $\mathsf{n} : \mathsf{int} \to \mathsf{message}$ be a name. Consider the recursive function $\ell\ i$ computing the list of the first i values of n.*

$$\ell = \lambda(i : \mathsf{int}).\,\mathsf{if}\ i = 0\ \mathsf{then}\ \mathsf{empty}\ \mathsf{else}\ \langle \mathsf{n}\ i, \ell\ (\mathsf{pred}\ i)\rangle$$

Let $i : \mathsf{int}$ be a variable declaration. The only names appearing in occurrences in the (infinite) set of generalised subterms $\mathcal{ST}_{\mathcal{E}}(\ell\ i)$ are of the form $(\mathsf{n}\ (\mathsf{pred}^j\ i))$ for $j \in \mathbb{N}$ (pred^j denotes j applications of pred). Proposition 6 states that, as expected, the interpretation of $(\ell\ i)$ only depends on the section of the random tapes containing these names. In particular, it does not depend on sections of the tape containing any $(\mathsf{n}\ j)$ for $j > i$.

1) The rule: Let $\mathsf{n} : \tau_0 \to \tau$ and $\mathsf{n}_{\mathsf{fresh}} : \mathsf{unit} \to \tau$ be names. We assume that $\mathsf{n}_{\mathsf{fresh}}$ does not appear in $\mathcal{E}, \vec{u}$ and t (*i.e.* it is a fresh symbol), except in its declaration. Let Θ be a set of global hypotheses. We ask that all declared variables x of $\mathcal{E}$ occur only in eta-long form, and are either names or only depend on the adversarial tape (*i.e.* $\mathsf{adv}(x) \in \Theta$).

The following rule states that giving the value of a name symbol n at an index t which has not been involved in the computation of $\vec{u}$ is equivalent to sampling a value from the fresh name symbol $\mathsf{n}_{\mathsf{fresh}}$:

$$\mathsf{G.EQUIV.FRESH}$$
$$\frac{\mathcal{E}; \Theta \vdash [\phi^{\mathsf{n},t}_{\mathsf{fresh}}(\vec{u}, t)]}{\mathcal{E}; \Theta \vdash \vec{u}, \mathsf{n}\ t \sim \vec{u}, \mathsf{n}_{\mathsf{fresh}}\ ()}$$

where $\phi^{\mathsf{n},t}_{\mathsf{fresh}}(\vec{u}, t)$ is any well-typed formula in $\mathcal{E}$ implying the *freshness* of $\mathsf{n}\ t$. Formally, we require that for every model $\mathbb{M} : \mathcal{E}$ of Θ, for every $\eta \in \mathbb{N}$ and $\rho \in \mathbb{T}_{\mathbb{M},\eta}$:

$$[\![\phi^{\mathsf{n},t}_{\mathsf{fresh}}(\vec{u}, t)]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}} = 1 \text{ implies}$$
$$[\![\phi]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}} = 1 \text{ for every } \phi \in \mathcal{S}^{\mathsf{n},t}_{\mathsf{fresh}}(\mathcal{ST}_{\mathcal{E}}(\vec{u}, t)) \quad (2)$$

where, for any (possibly infinite) set of occurrences $\mathbb{S}$, $\mathcal{S}^{\mathsf{n},t}_{\mathsf{fresh}}(\mathbb{S})$ is a (possibly infinite) set of formulas expressing the fact that n is *not sampled* at index t in the set $\mathbb{S}$:

$$\mathcal{S}^{\mathsf{n},t}_{\mathsf{fresh}}(\mathbb{S}) \stackrel{\mathsf{def}}{=} \left\{(\forall \vec{\alpha}.\psi \Rightarrow t \neq t_0) \mid (\vec{\alpha}, \psi, \mathsf{n}\ t_0) \in \mathbb{S}\right\}$$

Proposition 7. *The rule* G.EQUIV.FRESH *is sound.*

A detailed proof is given in Appendix F-A.

2) Choosing $\phi^{\mathsf{n},t}_{\mathsf{fresh}}(\vec{u}, t)$: We left the choice of the formula $\phi^{\mathsf{n},t}_{\mathsf{fresh}}(\vec{u}, t)$ open in our rule. Formally, we do not have a single rule, but a rule schema containing one rule for every formula $\phi^{\mathsf{n},t}_{\mathsf{fresh}}(\vec{u}, t)$ satisfying the condition in Eq. (2).

As it is, our rule schema is not effective. In Appendix F-B, we present an effective method to build a suitable formula $\phi^{\mathsf{n},t}_{\mathsf{fresh}}(\vec{u}, t)$, making the rule *implementable*. To do this, we describe a technique to over-approximate the (possibly infinite) set of occurrences of $\mathcal{ST}_{\mathcal{E}}(\vec{u}, t)$ by a *finite* set of occurrences $\mathbb{S}_{\mathsf{approx}}$, and we show that it is sound to take for $\phi^{\mathsf{n},t}_{\mathsf{fresh}}(\vec{u}, t)$ the conjunction of all formulas in $\mathcal{S}^{\mathsf{n},t}_{\mathsf{fresh}}(\mathbb{S}_{\mathsf{approx}})$.

Example 10. *Continuing Example 9, consider $\mathcal{ST}_{\mathcal{E}}(\ell\ i)$, where $\ell\ i$ is the list of the i first instances of name n. All name occurrences in $\mathcal{ST}_{\mathcal{E}}(\ell\ i)$ are of the form*

$$(\epsilon, i \neq \mathsf{zero} \wedge \mathsf{pred}\ i \neq \mathsf{zero} \wedge \cdots \wedge \mathsf{pred}^j\ i \neq \mathsf{zero}, \mathsf{n}\ (\mathsf{pred}^j\ i))$$

for $j \in \mathbb{N}$. All of these are subsumed by the occurrence $(\mathsf{j} : \mathsf{int}, i > \mathsf{j}, \mathsf{n}\ \mathsf{j})$. Hence taking $\mathbb{S}_{\mathsf{approx}}$ to be this single occurrence soundly and finitely over-approximates $\mathcal{ST}_{\mathcal{E}}(\ell\ i)$.

Since this approximation must be finite, it cannot be perfect. The difficulty lies in building a formula precise enough to be useful (*e.g.* taking $\phi^{\mathsf{n},t}_{\mathsf{fresh}}(\vec{u}, t) = \mathsf{false}$ is sound, but useless), while keeping a high degree of automation (the approach we settled upon allows the computation of that formula to be fully automated). We justify the precision of the formulas generated by our approach through our concrete case studies (see Section V).

This two-step approach is modular: we prove the soundness of the rule once and for all, but allow for future improvements in our implementation.

B. Cryptographic Rules

Our proof system features several *cryptographic rules* capturing standard security assumptions. We describe below the rule for unforgeable signatures. Other rules are described in Appendix G.

Assume $\mathcal{E}$ contains the symbols $\mathsf{pk}, \mathsf{sign}$ and verify for, resp., the public key associated to a private key, the signature of a

message with a private key, and the verification that a signature matches a message and a public key:

$$\mathsf{pk} : \mathsf{key} \to \mathsf{pubkey} \qquad \mathsf{sign} : \mathsf{message} \to \mathsf{key} \to \mathsf{signature}$$
$$\mathsf{verify} : \mathsf{signature} \to \mathsf{message} \to \mathsf{pubkey} \to \mathsf{bool}$$

The expected behaviour of these functions on correct signatures is described by the following axiom:

$$[\forall m : \mathsf{message}, k : \mathsf{key}.\ \mathsf{verify}\ (\mathsf{sign}\ m\ k)\ m\ (\mathsf{pk}\ k)]$$

The *Existential Unforgeability under Chosen Message Attacks* (EUF-CMA) assumption [28] is a standard cryptographic hypothesis stating that an adversary cannot construct valid signatures without knowing the secrecy signing key. More precisely, for a randomly sampled key k, an adversary with access to $(\mathsf{pk}\ \mathsf{k})$ and an a signing oracle $\mathsf{sign}(\cdot, \mathsf{k})$, cannot produce a signature s and a message m such that $(\mathsf{verify}\ s\ m\ (\mathsf{pk}\ \mathsf{k}))$ holds, without having called the oracle on m. The assumption is captured by the following rule:

$$
\text{L.EUF}
$$
$$
\dfrac{\mathcal{E};\Theta;\Gamma, \neg(\phi_{\mathsf{key}}^{\mathsf{k}}(s,m,t) \wedge \phi_{\mathsf{sign}}^{\mathsf{k}}(s,m,t)) \vdash \phi \qquad \mathcal{E};\Theta;\emptyset \vdash_{\mathsf{pptm}} s,m,t}{\mathcal{E};\Theta, \mathsf{const}(t);\Gamma, \mathsf{verify}\ s\ m\ (\mathsf{pk}\ (\mathsf{k}\ t)) \vdash \phi}
$$

where $\mathcal{E};\Theta;\emptyset \vdash_{\mathsf{pptm}} s,m,t$ states that the semantics $\llbracket \cdot \rrbracket_{\mathbb{M},\mathcal{E}}^{\eta,\rho}$ actually corresponds to a PPTM machine when evaluating s, m and t (see Appendix G-A) $\phi_{\mathsf{key}}^{\mathsf{k}}(s,m,t)$ and $\phi_{\mathsf{sign}}^{\mathsf{k}}(s,m,t)$ are, similarly to how we proceeded for the freshness rule, any formulas well-typed in $\mathcal{E}$ that imply, respectively,

- the correct use of $(\mathsf{k}\ t)$: the attacker only has access to the public key and the signing oracle, *i.e.* $(\mathsf{k}\ t)$ can only appears in $\mathsf{pk}\ (\mathsf{k}\ t)$ or $\mathsf{sign}\ \_\ (\mathsf{k}\ t)$;
- that m is not given to the signing oracle, *i.e.* $\mathsf{sign}\ m\ (\mathsf{k}\ t)$ is never computed.

Essentially, to show some ϕ when a signature is known to verify, we may assume either the key was badly used or the message was submitted to the oracle. The rule finally requires that t is constant (actually, we only need t deterministic).

In order to specify $\phi_{\mathsf{key}}^{\mathsf{k}}(s,m,t)$ and $\phi_{\mathsf{sign}}^{\mathsf{k}}(s,m,t)$ formally, we adapt the notion of generalised subterm. For a term u, $\mathcal{ST}_{\mathcal{E}}(u)$ is, essentially, the set of all subterms an adversary needs to compute to obtain u. We need here a similar set $\mathcal{ST}_{\mathcal{E},\mathsf{k},t}^{\mathsf{euf}}(u)$, that is however aware of the fact that the public key and signing oracle are provided to the adversary. That is, any subterm of the form $\mathsf{k}\ u'$ used either under pk or as signing key only needs to be computed by the adversary when $u' \neq t$: indeed when $u' = t$ it will be provided by the oracle, and thus can safely be ignored. Formally, we define $\mathcal{ST}_{\mathcal{E},\mathsf{k},t}^{\mathsf{euf}}(u)$ recursively just like $\mathcal{ST}_{\mathcal{E}}(u)$ (Fig. 1), with two exceptions when u is a function application:

$$
\mathcal{ST}_{\mathcal{E},\mathsf{k},t}^{\mathsf{euf}}(\mathsf{pk}\ (\mathsf{k}\ t_1)) \overset{\mathsf{def}}{=}
$$
$$
\{(\epsilon, \mathsf{true}, \mathsf{pk}\ (\mathsf{k}\ t_1))\} \cup \{(\epsilon, \mathsf{true}, \mathsf{pk})\}
$$
$$
\cup\ \mathcal{ST}_{\mathcal{E},\mathsf{k},t}^{\mathsf{euf}}(t_1) \cup [t \neq t_1]\mathcal{ST}_{\mathcal{E},\mathsf{k},t}^{\mathsf{euf}}(\mathsf{k}\ t_1)
$$
$$
\mathcal{ST}_{\mathcal{E},\mathsf{k},t}^{\mathsf{euf}}(\mathsf{sign}\ t_1\ (\mathsf{k}\ t_2)) \overset{\mathsf{def}}{=}
$$
$$
\{(\epsilon, \mathsf{true}, \mathsf{sign}\ t_1\ (\mathsf{k}\ t_2))\} \cup \{(\epsilon, \mathsf{true}, \mathsf{sign})\}
$$
$$
\cup\ \mathcal{ST}_{\mathcal{E},\mathsf{k},t}^{\mathsf{euf}}(t_1) \cup \mathcal{ST}_{\mathcal{E},\mathsf{k},t}^{\mathsf{euf}}(t_2) \cup [t \neq t_2]\mathcal{ST}_{\mathcal{E},\mathsf{k},t}^{\mathsf{euf}}(\mathsf{k}\ t_2)
$$

We can now formally express the conditions on $\phi_{\mathsf{key}}^{\mathsf{k}}(s,m,t)$ and $\phi_{\mathsf{sign}}^{\mathsf{k}}(s,m,t)$. For every model $\mathbb{M} : \mathcal{E}$ of Θ, every $\eta \in \mathbb{N}$ and $\rho \in \mathbb{T}_{\mathbb{M},\eta}$, we require that:

1) If $\llbracket \phi_{\mathsf{key}}^{\mathsf{k}}(s,m,t) \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = 1$ then $\llbracket \forall \vec{\alpha}.\ \psi \Rightarrow t \neq t_0 \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = 1$ for every $(\vec{\alpha}, \psi, \mathsf{k}\ t_0) \in \mathcal{ST}_{\mathcal{E},\mathsf{k},t}^{\mathsf{euf}}(s,m,t)$.

2) If $\llbracket \phi_{\mathsf{sign}}^{\mathsf{k}}(s,m,t) \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = 1$ then

$$\llbracket \forall \vec{\alpha}.\ \psi \Rightarrow (t = t_0) \Rightarrow (m \neq\ m_0) \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = 1$$

for every $(\vec{\alpha}, \psi, \mathsf{sign}\ m_0\ (\mathsf{k}\ t_0)) \in \mathcal{ST}_{\mathcal{E},\mathsf{k},t}^{\mathsf{euf}}(s,m,t)$.

Proposition 8. *The rule* L.EUF *is sound.*

Proof (sketch, see Appendix G-B for details). Fix a model $\mathbb{M} : \mathcal{E}$ satisfying the premises. We construct a Turing machine $\mathcal{M}$ that computes $\llbracket s \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho}$ and $\llbracket m \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho}$, which is is doable in PTIME by assumption. More specifically, our machine uses oracles to compute the interpretations of subterms $\mathsf{pk}\ (\mathsf{k}\ t')$ and $(\mathsf{sign}\ m\ (\mathsf{k}\ t'))$ when $t' = t$. If k has to be evaluated on $\llbracket t \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho}$ outside of these oracle calls, the machine fails. Such failures cannot happen when $\llbracket \phi_{\mathsf{key}}^{\mathsf{k}}(s,m,t) \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = 1$. If, moreover, $\llbracket \phi_{\mathsf{sign}}^{\mathsf{k}}(s,m,t) \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = 1$, our adversary wins the EUF-CMA game which can only happen with negligible probability. We conclude that $\mathbb{M} \models [\mathsf{verify}\ s\ m\ (\mathsf{pk}\ (\mathsf{k}\ t) \Rightarrow \neg(\phi_{\mathsf{key}}^{\mathsf{k}}(s,m,t) \wedge \phi_{\mathsf{sign}}^{\mathsf{k}}(s,m,t))]$, hence the conclusion of the rule is satisfied. $\square$

V. IMPLEMENTATION AND CASE-STUDY

SQUIRREL [13] is a proof assistant dedicated to analysing cryptographic protocols, implementing a proof system for the CCSA meta-logic [12], [14]. Users describe protocols in a process algebra with mutable states (inspired by [29]), then specify and prove security properties. Proofs are built using *tactics* derived from proof rules: generic reasoning with COQ-inspired tactics (*e.g.* **apply**, **rewrite**); cryptographic and freshness reasoning with dedicated ones (*e.g.* **fresh**, **euf**).

We integrated into SQUIRREL most of the logic presented in this paper. The new version of the prover is consistent with the semantics of our logic, though it does not implement all of its features, as detailed next. It is open source and can be found at [13]. The changes represent roughly 15 kLoC.

First, we added support for higher-order terms – though not for user-defined *recursive* terms yet, which will require a significant re-work of the occurrence approximation mechanisms. Second, we lifted syntactic restrictions on built-in types (*e.g.* **timestamp**, **index**), which we replaced with $\mathsf{const}(\cdot)$ side-conditions. Finally, we implemented improved versions of several freshness and cryptographic tactics (IND-CCA, INT-CTXT, EUF-CMA), to support key corruption.

The rest of this section showcases these new features through two case-studies: a formalisation of the hybrid argument, which relies on higher-order reasoning, and an analysis of the signed Diffie-Hellman protocol with corruption. Also note that we ported to the new setting all existing case studies from previous work using SQUIRREL [12], [14], [15].

```
global goal hybrid [α] (N₁ : int) (tₗ, tᵣ : int → α) (z : α) :
  const(N₁) ⇒
  (∀̃ (N₀ : int),
     const(N₀) ⇒ [N₀ ≤ N₁] ⇒
     ( z, λ(i:int). if i < N₀ then tₗ i else z ∼
       z, λ(i:int). if i < N₀ then tᵣ i else z   ) ⇒
     ( z, (λ(i:int). if i < N₀ then tₗ i else z), tₗ N₀ ∼
       z, (λ(i:int). if i < N₀ then tᵣ i else z), tᵣ N₀   )) ⇒
  ( z, λ(i:int). if i ≤ N₁ then tₗ i else z ∼
    z, λ(i:int). if i ≤ N₁ then tᵣ i else z   ).
```

Listing 1. Hybrid argument in SQUIRREL (using polymorphism)

A. Case Study: Hybrid Argument

The hybrid argument is a standard technique to prove the indistinguishability of distributions [30], [31]. Using our new higher-order features, we formalised and proved a version of it in SQUIRREL[4], stated in Listing 1. Essentially, the hybrid argument states that to show $(t_l\ i)_{i \leq N_1} \sim (t_r\ i)_{i \leq N_1}$ for some constant N_1, it suffices to show, for all $N_0 \leq N_1$:

$$(t_l\ i)_{i < N_0} \sim (t_r\ i)_{i < N_0} \tilde{\Rightarrow} (t_l\ i)_{i < N_0}, t_l\ N_0 \sim (t_r\ i)_{i < N_0}, t_r\ N_0$$

At its core, it is just an induction principle. The difficulty when using it in paper proofs, is to correctly show that the advantages of the distinguishers involved are negligible. Here, this is handled by the logic, and is proved by a simple induction – the only subtlety lies in dealing with higher-order terms. Consequently, our SQUIRREL formalisation is quite short (≤100 lines). Moreover, we implemented some basic higher-order matching into SQUIRREL, so that instantiating this lemma is in some cases automatic, when the **apply** tactic manages to infer the higher-order arguments t_l and t_r.

Discussion. This is not the first mechanised formalisation of the hybrid argument. Notably, one has been written in EasyCrypt[5]. EasyCrypt [6] is a general-purpose proof assistant with special support for cryptographic reasoning. It uses a higher-order logic built on top of a probabilistic Relational Hoare Logic [33]. In EasyCrypt, protocols are encoded as programs – *i.e.* modules and functors, whose parameters encode oracles. Security properties are written as relational properties over pairs of programs. Contrary to SQUIRREL, probabilistic reasoning is exposed to the user. This framework is very precise and expressive, but that comes at a cost: writing EasyCrypt proofs tends to be long and arduous. One reason is that, to instantiate a lemma, the user must often manually define the programs it is being applied to. The hybrid argument formalisation in EasyCrypt is ∼650 lines, contains ∼15 different modules, and applying it usually requires to write additional modules – which is markedly more work than in SQUIRREL.

Although comparing the same developments in different tool is a hazardous task, we believe this illustrates the fact that our new logic provides an appealing trade-off between expressivity and usability. Precisely and thoroughly comparing SQUIRREL with other tools is left as future work.

B. Case Study: Forward Secrecy

The signed Diffie-Hellman protocol [34] (Signed-DH) is an authenticated key exchange protocol, between two agents A, B. Each agent starts with a long-term signing key, resp. k_A, k_B. By running the protocol, they wish to authenticate each other, and to derive a shared secret key which can then be used *e.g.* to exchange encrypted messages.

The protocol fixes a finite cyclic group $\mathcal{G}$, with a generator g. First, A samples a value a uniformly at random in $\{1, \ldots, |\mathcal{G}|\}$, and sends g^a to B. B answers with his own value g^b for b freshly sampled, together with a signature $\mathsf{sign}(\langle 1, A, g^b, g^a \rangle, k_B)$. After checking the signature, A sends back her own signature $\mathsf{sign}(\langle 2, B, g^a, g^b \rangle, k_A)$ for B to check. If the signature checks out, A also computes the key $k = H((g^b)^a) = H(g^{a \cdot b})$, where H is a Random Oracle [35]. B derives the same key using his own secret value b and the group element g^a received from A.

We analysed the Signed-DH protocol in our new version of SQUIRREL[6]. We proved mutual authentication for A and B, and Forward Secrecy [36], [37] of the key (from A's point of view, though a similar result should hold for B). Standard secrecy states that the key derived by A is indistinguishable from a fresh uniformly sampled key. Forward secrecy further mandates that the key remains secure going forward, *i.e.* even if the long-term keys (here k_A and k_B) are *later* leaked to the adversary. In practice, this allows past communication between A and B, encrypted with $g^{a \cdot b}$, to remain confidential even if an adversary later gains control of A or B's devices. We express it in SQUIRREL as follows:

```
global goal kA_fs (t, t₀ : timestamp, A,B:index) :
  const(t,t₀,A,B) ⇒ [t = A2(A,B)] ⇒
  [t ≤ t₀] ⇒ [not (corrupt B < t)] ⇒
  (frame@t₀, if cond@t then kA(A,B)@t ∼
   frame@t₀, if cond@t then n_fresh).
```

Listing 2. Forward Secrecy in SQUIRREL, simplified.

Lemma kA_fs states that the key kA(A,B)@t derived by A at time t = A2(A,B) is indistinguishable from a fresh name n_{fresh}, provided A correctly checked B's signature (condition cond@t). This holds at any later time $t_0 \geq t$, provided no corruption occurred *before* the key was established at t. Here, only B must not be corrupted. Hence the requirement that [not (corrupt B < t)]. We allow corruption after t: in particular, corruption can occur in the time-frame]t,t_0] without compromising the security of the key at time t_0.

We proved this property, as well as the authentication properties, in our new version of SQUIRREL, using our new cryptographic tactics supporting key-corruption, in particular the unforgeability of the signature (EUF-CMA).

VI. CONCLUSION

We have presented a new logic for reasoning about cryptographic protocols. Unifying the CCSA base logic and its meta-logic into a higher-order logic, we obtain a general framework that is more expressive and extensible. We extended the SQUIRREL prover to conform to this new logic, and

validated the extra expressiveness on new case studies. We believe this work constitutes a solid theoretical foundation for the SQUIRREL prover and its future extensions.

As future work, we intend to study how to leverage the new features of our logic – in particular, recursive definitions – to model different execution models, in order to study *e.g.* side-channel behaviours, distance bounding protocols, or alternative attackers who can for instance influence the scheduling of actions. It would also be desirable to achieve some forms of completeness in our core sequent calculus, either as a cut elimination result or by relating provability and validity in a suitable class of models; both questions are currently wide open.

REFERENCES

[1] D. Dolev and A. C. Yao, "On the security of public key protocols," *IEEE Trans. Information Theory*, vol. 29, no. 2, pp. 198–207, 1983.

[2] B. Blanchet, "An efficient cryptographic protocol verifier based on prolog rules," in *CSFW*. IEEE Computer Society, 2001, pp. 82–96.

[3] B. Blanchet, V. Cheval, and V. Cortier, "ProVerif with lemmas, induction, fast subsumption, and much more," in *IEEE Symposium on Security and Privacy*. IEEE, 2022, pp. 69–86.

[4] S. Meier, B. Schmidt, C. Cremers, and D. Basin, "The TAMARIN prover for the symbolic analysis of security protocols," in *25th International Conference on Computer Aided Verification, CAV'13*. Springer-Verlag, 2013, pp. 696–701.

[5] B. Blanchet, "A computationally sound mechanized prover for security protocols," *IEEE Trans. Dependable Secur. Comput.*, vol. 5, no. 4, pp. 193–207, 2008.

[6] G. Barthe, B. Grégoire, S. Heraud, and S. Z. Béguelin, "Computer-aided security proofs for the working cryptographer," in *CRYPTO*, ser. Lecture Notes in Computer Science, vol. 6841. Springer, 2011, pp. 71–90.

[7] D. A. Basin, A. Lochbihler, and S. R. Sefidgar, "CryptHOL: Game-based proofs in higher-order logic," *J. Cryptol.*, vol. 33, no. 2, pp. 494–566, 2020.

[8] G. Bana and H. Comon-Lundh, "A computationally complete symbolic attacker for equivalence properties," in *CCS*. ACM, 2014, pp. 609–620.

[9] H. Comon and A. Koutsos, "Formal computational unlinkability proofs of RFID protocols," in *CSF*. IEEE Computer Society, 2017, pp. 100–114.

[10] G. Bana, R. Chadha, and A. K. Eeralla, "Formal analysis of vote privacy using computationally complete symbolic attacker," in *ESORICS (2)*, ser. Lecture Notes in Computer Science, vol. 11099. Springer, 2018, pp. 350–372.

[11] A. Koutsos, "The 5G-AKA authentication protocol privacy," in *EuroS&P*. IEEE, 2019, pp. 464–479.

[12] D. Baelde, S. Delaune, C. Jacomme, A. Koutsos, and S. Moreau, "An interactive prover for protocol verification in the computational model," in *IEEE Symposium on Security and Privacy*. IEEE, 2021, pp. 537–554.

[13] The Squirrel Prover repository. https://github.com/squirrel-prover/squirrel-prover/.

[14] D. Baelde, S. Delaune, A. Koutsos, and S. Moreau, "Cracking the stateful nut: Computational proofs of stateful security protocols using the squirrel proof assistant," in *CSF*. IEEE, 2022, pp. 289–304.

[15] C. Cremers, C. Fontaine, and C. Jacomme, "A logic and an interactive prover for the computational post-quantum security of protocols," in *IEEE Symposium on Security and Privacy*. IEEE, 2022, pp. 125–141.

[16] A. Juels and S. A. Weis, "Defining strong privacy for RFID," *ACM Transactions on Information and System Security (TISSEC)*, vol. 13, no. 1, p. 7, 2009.

[17] C. F. Morgenstern, "The measure quantifier," *J. Symb. Log.*, vol. 44, no. 1, pp. 103–108, 1979.

[18] S. Zachos, "Probabilistic quantifiers and games," *J. Comput. Syst. Sci.*, vol. 36, no. 3, pp. 433–451, 1988.

[19] M. Antonelli, U. D. Lago, and P. Pistone, "On counting propositional logic and wagner's hierarchy," in *ICTCS*, ser. CEUR Workshop Proceedings, vol. 3072. CEUR-WS.org, 2021, pp. 107–121.

[20] J. Y. Halpern, "An analysis of first-order logics of probability," *Artif. Intell.*, vol. 46, no. 3, pp. 311–350, 1990.

[21] ——, *Reasoning about uncertainty*. MIT Press, 2005.

[22] M. Antonelli, U. D. Lago, and P. Pistone, "Curry and howard meet borel," in *LICS*. ACM, 2022, pp. 45:1–45:13.

[23] N. J. Nilsson, "Probabilistic logic revisited," *Artif. Intell.*, vol. 59, no. 1-2, pp. 39–42, 1993.

[24] M. Vákár, O. Kammar, and S. Staton, "A domain theory for statistical probabilistic programming," *Proc. ACM Program. Lang.*, vol. 3, no. POPL, pp. 36:1–36:29, 2019.

[25] J. Goubault-Larrecq, X. Jia, and C. Théron, "A domain-theoretic approach to statistical programming languages," *CoRR*, vol. abs/2106.16190, 2021.

[26] N. D. Goodman, V. K. Mansinghka, D. M. Roy, K. A. Bonawitz, and J. B. Tenenbaum, "Church: a language for generative models," in *UAI*. AUAI Press, 2008, pp. 220–229.

[27] V. Shoup, "Sequences of games: a tool for taming complexity in security proofs," *IACR Cryptol. ePrint Arch.*, p. 332, 2004.

[28] J. Katz and Y. Lindell, *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014.

[29] M. Abadi, B. Blanchet, and C. Fournet, "The applied pi calculus: Mobile values, new names, and secure communication," *J. ACM*, vol. 65, no. 1, pp. 1:1–1:41, 2018.

[30] M. Fischlin and A. Mittelbach, "An overview of the hybrid argument," *IACR Cryptol. ePrint Arch.*, p. 88, 2021.

[31] M. Bellare, A. Boldyreva, and S. Micali, "Public-key encryption in a multi-user setting: Security proofs and improvements," in *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, ser. Lecture Notes in Computer Science, B. Preneel, Ed., vol. 1807. Springer, 2000, pp. 259–274. [Online]. Available: https://doi.org/10.1007/3-540-45539-6_18

[32] the EasyCrypt development team, "Source code of EasyCrypt," January 2022, https://github.com/EasyCrypt/easycrypt.

[33] G. Barthe, C. Fournet, B. Grégoire, P. Strub, N. Swamy, and S. Z. Béguelin, "Probabilistic relational verification for cryptographic implementations," in *POPL*. ACM, 2014, pp. 193–206.

[34] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[35] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *CCS*. ACM, 1993, pp. 62–73.

[36] A. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1996.

[37] C. Cremers and M. Feltz, "Beyond eCK: perfect forward secrecy under actor compromise and ephemeral-key reveal," *Des. Codes Cryptogr.*, vol. 74, no. 1, pp. 183–218, 2015.

[38] S. J. Bellantoni and S. A. Cook, "A new recursion-theoretic characterization of the polytime functions," *Comput. Complex.*, vol. 2, pp. 97–110, 1992.

[39] M. Barbosa, G. Barthe, B. Grégoire, A. Koutsos, and P. Strub, "Mechanized proofs of adversarial complexity and application to universal composability," in *CCS*. ACM, 2021, pp. 2541–2563.

[40] D. Leivant and J. Marion, "Lambda calculus characterizations of polytime," *Fundam. Informaticae*, vol. 19, no. 1/2, pp. 167–184, 1993.

[41] O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," *Journal of the ACM (JACM)*, vol. 33, no. 4, pp. 792–807, 1986.

APPENDIX A
TYPING RULES AND TYPES RESTRICTIONS

We give in this section the typing rules, and some additional details on the logic which we omitted from the body.

The typing rules of our logic are given in Fig. 6.

Type restrictions: We now present the semantics of type restrictions. Let τ be a type, and $\mathsf{label}(\tau)$ its associated set of labels. We give the semantics of each label by describing in what way it restricts its potential model $\mathbb{M} : \mathcal{E}$:

- if $\mathtt{fixed} \in \mathsf{label}(\tau)$ then we must have, for all $\eta, \eta' \in \mathbb{N}$, $[\![\tau]\!]_{\mathbb{M}}^{\eta} = [\![\tau]\!]_{\mathbb{M}}^{\eta'}$
- if $\mathtt{finite} \in \mathsf{label}(\tau)$ then we must have, for all $\eta \in \mathbb{N}$,

$$\left| [\![\tau]\!]_{\mathbb{M}}^{\eta} \right| < +\infty$$

Term typing judgements

$$\frac{}{\mathcal{E} \vdash x : \mathcal{E}(x)} \text{TY.DECL}$$

$$\frac{\mathcal{E} \vdash t_1 : \tau_0 \to \tau_1 \quad \mathcal{E} \vdash t_2 : \tau_0}{\mathcal{E} \vdash t_1\, t_2 : \tau_1} \text{TY.FUN-APP}$$

$$\frac{\mathcal{E}, x : \tau_0 \vdash t : \tau_1}{\mathcal{E} \vdash \lambda(x : \tau_0).\, t : \tau_0 \to \tau_1} \text{TY.LAMBDA}$$

$$\frac{}{\mathcal{E} \vdash \mathsf{true} : \mathsf{bool}} \text{TY.TRUE}$$

$$\frac{}{\mathcal{E} \vdash \mathsf{false} : \mathsf{bool}} \text{TY.FALSE}$$

$$\frac{\mathcal{E} \vdash t_i : \tau \text{ for } i \in \{1,2\} \quad \mathcal{E} \vdash t : \mathsf{bool}}{\mathcal{E} \vdash \mathsf{if}\ t\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 : \tau} \text{TY.IF-THEN-ELSE}$$

$$\frac{\mathcal{E} \vdash t_i : \tau \text{ for } i \in \{1,2\}}{\mathcal{E} \vdash t_1 = t_2 : \mathsf{bool}} \text{TY.EQ}$$

Global formula typing judgements

$$\frac{\mathcal{E} \vdash t : \mathsf{bool}}{\mathcal{E} \vdash [t]} \text{GTY.REACH}$$

$$\frac{\exists \tau_1, \ldots, \tau_n \text{ s.t.} \quad \forall i.\, \mathcal{E} \vdash t_i : \tau_i \text{ and } \mathcal{E} \vdash s_i : \tau_i}{\mathcal{E} \vdash t_1, \ldots, t_n \sim s_1, \ldots, s_n} \text{GTY.EQUIV}$$

$$\frac{\diamond \in \{\tilde{\vee}, \tilde{\wedge}\} \quad \mathcal{E} \vdash F_1 \quad \mathcal{E} \vdash F_2}{\mathcal{E} \vdash F_1 \diamond F_2} \text{GTY.}\diamond$$

$$\frac{\mathcal{E} \vdash F}{\mathcal{E} \vdash \neg F} \text{GTY.NEG}$$

$$\frac{\mathcal{E}, x : \tau \vdash F \quad \mathcal{Q} \in \{\tilde{\forall}, \tilde{\exists}\}}{\mathcal{E} \vdash \mathcal{Q}(x : \tau), F} \text{GTY.QUANT}$$

Convention: we require that $\mathcal{E}$ is only extended with fresh symbols (w.r.t. $\mathcal{E}$), which is always possible by alpha-renaming.

Figure 6. Typing rules for local terms and global formulas.

- if $\mathtt{polynomial} \in \mathsf{label}(\tau)$ then there must exists a polynomial $Q[X]$ such that, for all $\eta \in \mathbb{N}$,

$$\left| \llbracket \tau \rrbracket_{\mathbb{M}}^{\eta} \right| < Q[\eta]$$

Furthermore, there must exists a polynomial-time machine $\mathcal{A}$ such that $\mathcal{A}$ can enumerate all elements in $\llbracket \tau \rrbracket_{\mathbb{M}}^{\eta}$.

- if $\mathtt{large} \in \mathsf{label}(\tau)$ then, for all name symbol[7] $\mathsf{n} : \tau_0 \to \tau$ in $\mathcal{E}$, for all $\eta \in \mathbb{N}$, $a \in \llbracket \tau_0 \rrbracket_{\mathbb{M}}^{\eta}$ and $b \in \llbracket \tau \rrbracket_{\mathbb{M}}^{\eta}$,

$$\Pr_{\rho_{\mathsf{h}} \in \mathbb{T}_{\mathbb{M},\eta}^{\mathsf{h}}} \left(\llbracket \mathsf{n} \rrbracket_{\mathbb{M}}(\eta, a)(\rho_{\mathsf{h}}) = b \right) \leq \frac{1}{2^{c_{\tau} \cdot \eta}}$$

where $c_{\tau} > 0$ is a positive real number, which can optionally depend on the type τ.

- if $\mathtt{well\text{-}founded} \in \mathsf{label}(\tau)$ then we require that the distinguished symbol $<\, : \tau \to \tau \to \mathsf{bool}$ (used with infix notation) is interpreted as a well-founded order in $\mathbb{M}$. More precisely, for every $\eta \in \mathbb{N}$, there exists $\dot{<}$ such that $\llbracket < \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = \dot{<}$ for every tape $\rho \in \mathbb{T}_{\mathbb{M},\eta}$, and $(\llbracket \tau \rrbracket_{\mathbb{M}}^{\eta}, \dot{<})$ is a well-founded set.

APPENDIX B
WELL-FOUNDED DEFINITIONS

We first present elided details from Definition 1, and then prove Proposition 1.

---

[7]Note that we must define simultaneously the type restriction $\mathtt{large}$ and the declaration restriction $\mathtt{name}$. For the sack of simplicity, the semantics of both restrictions are given separately.

Internal order of a model: For an environment

$$\mathcal{E} = \mathcal{E}_0, \{x_i : \tau_i \to \tau_i' = \lambda x.t_i\}_{i \in [1;n]} \tag{3}$$

where $\mathcal{E}_0$ contains only declarations to be well-founded w.r.t. some model $\mathbb{M} : \mathcal{E}$, Definition 1 requires that there exists a well-founded order $<$ over $\{(x_i, e) \mid i \in [1;n], e \in \llbracket \tau_i \rrbracket_{\mathbb{M}}^{\eta}\}$ such that all recursive calls to defined variables are decreasing w.r.t. $<$. We need this well-founded order to be representable in the logic itself, to be able to use it in the rules of our proof system. This is captured by the requirements that $<$ is the order represented by the *internal order formulas* of $\mathcal{E}$, which we define below.

First, we require that any well-typed environment $\mathcal{E}$ comes with a finite set of predicates (terms well-typed in $\mathcal{E}_0$, not $\mathcal{E}$, to ensure that they do not depend on the *defined* variables in $\mathcal{E}$) called its internal order formulas

$$(<_{x_i,x_j})_{i,j \in [1;n]} \quad \text{s.t.} \quad \mathcal{E}_0 \vdash <_{x_i,x_j} : \tau_i \to \tau_j \to \mathsf{bool}$$

where the $(x_i)_{i \in [1;n]}$ are the defined variables of $\mathcal{E}$, and $(\tau_i)_{i \in [1;n]}$ the type of their arguments (as in Eq. (3)). We stress the fact that these formulas are independent of the model: to make this formal, we modify the definition of environment, and require that an environment is a pair of a list of declarations and definitions (as in the body), and an *additional* finite set of formulas which is its internal order formulas (as we only need to assume the existence of the latter, we usually omit them from environment descriptions).

Then, we say that an order $<$ is the *internal order* of $\mathcal{E}$ w.r.t. a term structure $\mathbb{M}$ when for every $\eta \in \mathbb{N}$ and

$$(x_i, a_i), (x_j, a_j) \in \{(x_i, e) \mid i \in [1;n], e \in \llbracket \tau_i \rrbracket_{\mathbb{M}}^{\eta}\}$$

we have $(x_i, a_i) < (x_j, a_j)$ iff for every $\rho \in \mathbb{T}_{\mathbb{M},\eta}$

$$\llbracket y_i <_{x_i,x_j} y_j \rrbracket_{\mathbb{M}[y_i \mapsto \mathbb{1}_{a_i}^{\eta}, y_j \mapsto \mathbb{1}_{a_j}^{\eta}]:(\mathcal{E}, y_i : \tau_i, y_j : \tau_j)}^{\eta,\rho}$$

Moreover, we let $\leq_{x_i,x_j}$ be the non-strict version of $<_{x_i,x_j}$

$$\leq_{x_i,x_j} \overset{\mathsf{def}}{=} \lambda y, y'.\, y = y' \vee y <_{x_i,x_j} y'$$

Remark 2. *Consider a* term structure $\mathbb{M}$ *for environment* $\mathcal{E}$. *If the internal order relation* $<$ *associated to the internal order formulas of* $\mathcal{E}$ *w.r.t.* $\mathbb{M}$ *is not a well-founded ordering, then* $\mathcal{E}$ *is not well-founded w.r.t.* $\mathbb{M}$ *(per Definition 1). Consequently,* $\mathbb{M}$ *cannot be extended into a* model *of our logic (as models are build from well-founded term structures).*

Concretely, this restricts term structures from which models can be constructed to those such that $<$ *is a well-founded ordering represented by* $\mathcal{E}$'s *internal order formulas.*

Well-foundedness: We recall and prove Proposition 1.

Proposition 1. *Let* $\mathcal{E}$ *be an environment that is well-founded w.r.t.* $\mathbb{M} : \mathcal{E}$. *Let* $\mathbb{M}_0$ *be the restriction of* $\mathbb{M}$ *to the declarations of* $\mathcal{E}$. *There exists a unique term structure* $\mathbb{M}_{\mathsf{rec}} : \mathcal{E}$ *extending* $\mathbb{M}_0$ *such that, for each* $(x_i : \tau_i \to \tau_i' = \lambda x.t_i) \in \mathcal{E}$, $\llbracket x_i \rrbracket_{\mathbb{M}_{\mathsf{rec}}:\mathcal{E}}^{\eta,\rho} = \llbracket \lambda x.t_i \rrbracket_{\mathbb{M}_{\mathsf{rec}}:\mathcal{E}}^{\eta,\rho}$ *for all* η, ρ.

Proof. Let $\mathcal{E}$ be a well-founded environment w.r.t. $\mathbb{M} : \mathcal{E}$. Let $\mathbb{M}_0$ be the restriction of $\mathbb{M}$ to the declared variables of $\mathcal{E}$.

Fix some value of η, and consider the well-founded order $<$ over the pairs (x_i, e) with $e \in \llbracket \tau_i \rrbracket_\mathbb{M}^\eta$ given by Definition 1. We then define, for all i, the functions $X_{i,\eta} : \mathbb{T}_{\mathbb{M},\eta} \to \llbracket \tau_i \rrbracket_\mathbb{M}^\eta \to \llbracket \tau_i' \rrbracket_\mathbb{M}^\eta$, by well-founded induction. In other words, we define, for any j and $e \in \llbracket \tau_j \rrbracket_\mathbb{M}^\eta$, the value of $X_{j,\eta}$ when given e as second argument, assuming that all functions $X_{i,\eta}$ are defined when their second argument e' is such that $(x_i, e') < (x_j, e)$. We can do this by taking $X_{j,\eta}(\rho)(e) = \llbracket t_j \rrbracket_{\mathbb{M}_{\mathsf{rec}} : \mathcal{E}, x : \tau_j}^{\eta, \rho}$ where $\mathbb{M}_{\mathsf{rec}}$ extends $\mathbb{M}_0$ with $\mathbb{M}_{\mathsf{rec}}(x_i) = X_i$ for all i and $\mathbb{M}_{\mathsf{rec}}(x) = e$ (even though our $X_{i,\eta}$ functions are partial at this point, as we justify next). Indeed, if some x_i occurs in t_j, it will give rise to an occurrence $(\vec{\alpha}, \phi, x_i\ t') \in \mathcal{ST}_{\mathcal{E}^*, x : \tau_j}(t_j)$. Now, if this occurrence is used in computing $\llbracket t_j \rrbracket_{\mathbb{M}_{\mathsf{rec}} : \mathcal{E}, x : \tau_j}^{\eta, \rho}$, this means that we have some extension $\mathbb{M}'$ of $\mathbb{M}_{\mathsf{rec}}$ such that $\llbracket \phi \rrbracket_{\mathbb{M}' : \mathcal{E}, (x : \tau_j), \vec{\alpha}}^{\eta, \rho} = 1$ — otherwise the interpretation would not rely on this branch of nested conditionals. Because of the condition on the free variables of t', we have

$$\llbracket t' \rrbracket_{\mathbb{M}' : \mathcal{E}, (x : \tau_j), \vec{\alpha}}^{\eta, \rho} = \llbracket t' \rrbracket_{\mathbb{M}'' : \mathcal{E}, (x : \tau_j), \vec{\alpha}}^{\eta, \rho}$$

where $\mathbb{M}''$ coincides with $\mathbb{M}$ on the defined variables x_i and otherwise coincides with $\mathbb{M}'$ (which is an extension of $\mathbb{M}_{\mathsf{rec}}$, not $\mathbb{M}$). Moreover, by definition of $\phi \setminus \{x_1, \dots, x_n\}$ we also have:

$$\llbracket \phi \setminus \{x_1, \dots, x_n\} \rrbracket_{\mathbb{M}'' : \mathcal{E}, (x : \tau_j), \vec{\alpha}}^{\eta, \rho} =$$
$$\llbracket \phi \setminus \{x_1, \dots, x_n\} \rrbracket_{\mathbb{M}' : \mathcal{E}, (x : \tau_j), \vec{\alpha}}^{\eta, \rho} = 1$$

We can thus conclude, by Definition 1, that $(x_i, \llbracket t' \rrbracket_{\mathbb{M}' : \mathcal{E}, (x : \tau_j), \vec{\alpha}}^{\eta, \rho}) < (x_j, e)$, thus $X_{i,\eta}$ is well-defined on $\llbracket t' \rrbracket_{\mathbb{M}' : \mathcal{E}, (x : \tau_j), \vec{\alpha}}^{\eta, \rho}$, *i.e.* $\llbracket x_i\ t' \rrbracket_{\mathbb{M}' : \mathcal{E}, (x : \tau_j), \vec{\alpha}}^{\eta, \rho}$ is well-defined.

Once the functions $X_{i,\eta}$ are totally defined, for all i and η, we can form $\mathbb{M}_{\mathsf{rec}}$ which extends $\mathbb{M}_0$ with $\mathbb{M}_{\mathsf{rec}}(x_i) = X_i$. This new model satisfies the required fixed point equations by construction. Finally, to justify our uniqueness claim, one can easily verify that if two models $\mathbb{M}_{\mathsf{rec}}^1$ and $\mathbb{M}_{\mathsf{rec}}^2$ satisfied our requirements, they would have to provide equal interpretations for all x_i; this is done, as expected, by well-founded induction. $\square$

APPENDIX C
GLOBAL FORMULAS

We make more detailed remarks regarding the probabilistic semantics of our global formula atoms.

Remark 3. *The finiteness of $\mathbb{T}_{\mathbb{M},\eta}$ immediately gives us that $\llbracket t \rrbracket_{\mathbb{M} : \mathcal{E}}^\eta$ is a family of random variables, but keeping this property without the requirement that $\mathbb{T}_{\mathbb{M},\eta}$ be finite would be non-trivial. Indeed, consider the semantics of $\forall (x : \tau).t$, which is for a given η*

$$X_{\forall x.t} \overset{\mathsf{def}}{=} \rho \mapsto \llbracket \forall (x : \tau).t \rrbracket_{\mathbb{M} : \mathcal{E}}^{\rho, \eta}$$

and let, for every $a \in \llbracket \tau \rrbracket_\mathbb{M}^\eta$, $X_{t\ a}$ be the function

$$X_{t\ a} \overset{\mathsf{def}}{=} \rho \mapsto \llbracket t \rrbracket_{\mathbb{M}[x \mapsto \mathbb{1}_a^\eta] : (\mathcal{E}, x : \tau)}^{\rho, \eta}$$

which we assume are random variables. Then, to show that $X_{\forall x.t}$ is a random variable, we must show (among other things) that

$$X_{\forall x.t}^{-1}(\{1\}) = \bigcap_{a \in \llbracket \tau \rrbracket_\mathbb{M}^\eta} X_{t\ a}^{-1}(\{1\})$$

is a measurable set of $\mathbb{T}_{\mathbb{M},\eta}$. If $\llbracket \tau \rrbracket_\mathbb{M}^\eta$ is at most countable, then this immediately follows from the fact that each $X_{t\ a}^{-1}(\{1\})$ is measurable, and that measurable sets are closed under countable intersection. But if $\llbracket \tau \rrbracket_\mathbb{M}^\eta$ is uncountable (which is the case, e.g., if $\tau = \mathsf{message} \to \mathsf{bool}$), then this arguments no longer applies.

Remark 4. *Since $\mathbb{T}_{\mathbb{M},\eta}$ must be finite, we can only represent random samplings with a finite amount of randomness in our logic. This puts some limits on what we can directly model in our logic using name symbols; e.g. we cannot model a random oracle [35] from $\mathsf{message}$ to $\mathsf{message}$ directly as a name symbol: indeed, we would need to use a $\mathsf{message}$ indexed name symbol, which itself requires an infinite number of random bits. Note that this limitation does not preclude us from modelling random oracles (see e.g. [14]), only from direct modelling using names.*

APPENDIX D
RELATIONSHIP TO META-LOGIC

In this section, we discuss how our new logic generalises the meta-logic of [14]. This discussion targets readers already familiar with the meta-logic and its underlying base logic [8], who would like to understand how they can work with our new framework instead.

It should be clear that several features of the new logic are out of the reach of the meta-logic approach, but it is less obvious why all features of the former meta-logic can be recovered in our new framework. Indeed, the technical points that allow it are scattered throughout the exposition of our logic. We summarise and detail these points below. Ideally, this discussion should ultimately be formalised as a conservativity result: a local meta-logic formula should be valid iff its suitable translation in the new logic is valid. As we shall see, this may not be quite the case, and there might be cases where a meta-logic formula is valid while its translation in the new logic is not. These cases are not necessarily problematic, though: from a practical perspective, it is good enough (and even preferable) if *provable* meta-logic formulas translate to provable higher-order logic formulas. We do not provide theoretical arguments concerning this question — though the proximity of proof systems gives hope that this would be possible. Note, however, that our implementation and case studies provides positive empirical evidence in this direction.

a) Base logic: We first explain how our higher-order logic generalises the base CCSA logic [8]. The base logic is a first-order logic whose function symbols are either names, honest function symbols, or adversarial function symbols. It features a single predicate $\sim$ interpreted as computational indistinguishability, which is also immediately available in our logic. Thus the only difficulty lies in the interpretation of

Mixed judgements rules

L.ByGlob
$$\frac{\mathcal{E};\Theta \vdash [\phi]}{\mathcal{E};\Theta;\Gamma \vdash \phi}$$

G.ByLoc
$$\frac{\mathcal{E};\Theta;\emptyset \vdash \phi}{\mathcal{E};\Theta \vdash [\phi]}$$

L.Localise
$$\frac{\mathcal{E};\Theta;\Gamma,\phi \vdash \psi}{\mathcal{E};\Theta,[\phi];\Gamma \vdash \psi}$$

L.Rewrite-Equiv
$$\frac{\mathcal{E};\Theta;\Gamma_1 \vdash \phi_1 \qquad \mathcal{E};\Theta \vdash (\Gamma_0 \Rightarrow \phi_0) \sim (\Gamma_1 \Rightarrow \phi_1)}{\mathcal{E};\Theta;\Gamma_0 \vdash \phi_0}$$

Local judgements: right rules

L.R-$\forall$
$$\frac{\mathcal{E},x:\tau;\Theta;\Gamma \vdash \phi}{\mathcal{E};\Theta;\Gamma \vdash \forall(x:\tau).\phi}$$

L.R-$\forall$-const
$$\frac{\{\texttt{fixed},\texttt{finite}\} \subseteq \mathsf{label}(\tau) \qquad \mathcal{E},x:\tau;\Theta,\mathsf{const}(x);\Gamma \vdash \phi}{\mathcal{E};\Theta;\Gamma \vdash \forall(x:\tau).\phi}$$

L.R-$\exists$
$$\frac{\mathcal{E};\Theta;\Gamma \vdash \phi\{x \mapsto t\} \qquad \mathcal{E} \vdash t:\tau}{\mathcal{E};\Theta;\Gamma \vdash \exists(x:\tau).\phi}$$

L.R-$\Rightarrow$
$$\frac{\mathcal{E};\Theta;\Gamma,\phi \vdash \psi}{\mathcal{E};\Theta;\Gamma \vdash \phi \Rightarrow \psi}$$

L.R-$\wedge$
$$\frac{\mathcal{E};\Theta;\Gamma \vdash \phi \qquad \mathcal{E};\Theta;\Gamma \vdash \psi}{\mathcal{E};\Theta;\Gamma \vdash \phi \wedge \psi}$$

L.R1-$\vee$
$$\frac{\mathcal{E};\Theta;\Gamma \vdash \phi}{\mathcal{E};\Theta;\Gamma \vdash \phi \vee \psi}$$

L.R2-$\vee$
$$\frac{\mathcal{E};\Theta;\Gamma \vdash \psi}{\mathcal{E};\Theta;\Gamma \vdash \phi \vee \psi}$$

L.R-true
$$\frac{}{\mathcal{E};\Theta;\Gamma \vdash \mathsf{true}}$$

Local judgements: left local rules

L.L-$\forall$
$$\frac{\mathcal{E};\Theta;\Gamma,\psi\{x \mapsto t\} \vdash \phi \qquad \mathcal{E} \vdash t:\tau}{\mathcal{E};\Theta;\Gamma,\forall(x:\tau).\psi \vdash \phi}$$

L.L-$\exists$
$$\frac{\mathcal{E},x:\tau;\Theta;\Gamma,\phi \vdash \psi}{\mathcal{E};\Theta;\Gamma,\exists(x:\tau).\phi \vdash \psi}$$

L.L-$\Rightarrow$
$$\frac{\mathcal{E};\Theta;\Gamma \vdash \phi_0 \qquad \mathcal{E};\Theta;\Gamma,\phi_1 \vdash \psi}{\mathcal{E};\Theta;\Gamma,\phi_0 \Rightarrow \phi_1 \vdash \psi}$$

L.L-$\wedge$
$$\frac{\mathcal{E};\Theta;\Gamma,\phi_0,\phi_1 \vdash \psi}{\mathcal{E};\Theta;\Gamma,\phi_0 \wedge \phi_1 \vdash \psi}$$

L.L-$\vee$
$$\frac{\mathcal{E};\Theta;\Gamma,\phi_0 \vdash \psi \qquad \mathcal{E};\Theta;\Gamma,\phi_1 \vdash \psi}{\mathcal{E};\Theta;\Gamma,\phi_0 \vee \phi_1 \vdash \psi}$$

L.L-false
$$\frac{}{\mathcal{E};\Theta;\Gamma,\mathsf{false} \vdash \phi}$$

Local judgements: left global rules

L.L-$\tilde{\forall}$
$$\frac{\mathcal{E};\Theta,F\{x \mapsto t\};\Gamma \vdash \phi \qquad \mathcal{E} \vdash t:\tau}{\mathcal{E};\Theta,\tilde{\forall}(x:\tau).F;\Gamma \vdash \phi}$$

L.L-$\tilde{\exists}$
$$\frac{\mathcal{E},x:\tau;\Theta,F;\Gamma \vdash \psi}{\mathcal{E};\Theta,\tilde{\exists}(x:\tau).F;\Gamma \vdash \psi}$$

L.L-$\tilde{\Rightarrow}$
$$\frac{\mathcal{E};\Theta \vdash F_0 \qquad \mathcal{E};\Theta,F;\Gamma \vdash \psi}{\mathcal{E};\Theta,F_0 \tilde{\Rightarrow} F;\Gamma \vdash \psi}$$

L.L-$\tilde{\wedge}$
$$\frac{\mathcal{E};\Theta,F_0,F_1;\Gamma \vdash \psi}{\mathcal{E};\Theta,F_0 \tilde{\wedge} F_1;\Gamma \vdash \psi}$$

L.L-$\tilde{\vee}$
$$\frac{\mathcal{E};\Theta,F_0;\Gamma \vdash \psi \qquad \mathcal{E};\Theta,F_1;\Gamma \vdash \psi}{\mathcal{E};\Theta,F_0 \tilde{\vee} F_1;\Gamma \vdash \psi}$$

L.L-$\tilde{\bot}$
$$\frac{}{\mathcal{E};\Theta,\tilde{\bot};\Gamma \vdash \phi}$$

Local judgements: other rules

L.Rewrite
$$\frac{\mathcal{E};\Theta;\Gamma \vdash \phi[s] \qquad \mathcal{E};\Theta;\Gamma \vdash s = t}{\mathcal{E};\Theta;\Gamma \vdash \phi[t]}$$

L.Axiom
$$\frac{}{\mathcal{E};\Theta;\Gamma,\phi \vdash \phi}$$

L.Absurd
$$\frac{\mathcal{E};\Theta;\Gamma,\phi \Rightarrow \mathsf{false} \vdash \mathsf{false}}{\mathcal{E};\Theta;\Gamma \vdash \mathsf{false}}$$

L.Cut-Loc
$$\frac{\mathcal{E};\Theta;\Gamma \vdash \phi \qquad \mathcal{E};\Theta;\Gamma,\phi \vdash \psi}{\mathcal{E};\Theta;\Gamma \vdash \psi}$$

L.Cut-Glob
$$\frac{\mathcal{E};\Theta \vdash F \qquad \mathcal{E};\Theta,F;\Gamma \vdash \psi}{\mathcal{E};\Theta;\Gamma \vdash \psi}$$

L.Weak
$$\frac{\mathcal{E};\Theta_0;\Gamma_0 \vdash \psi}{\mathcal{E};\Theta_0,\Theta_1;\Gamma_0,\Gamma_1 \vdash \psi}$$

Global judgements: right rules

G.R-$\tilde{\forall}$
$$\frac{\mathcal{E},x:\tau;\Theta \vdash F}{\mathcal{E};\Theta \vdash \tilde{\forall}(x:\tau).F}$$

G.R-$\tilde{\exists}$
$$\frac{\mathcal{E};\Theta \vdash F\{x \mapsto t\} \qquad \mathcal{E} \vdash t:\tau}{\mathcal{E};\Theta \vdash \tilde{\exists}(x:\tau).F}$$

G.R-$\tilde{\Rightarrow}$
$$\frac{\mathcal{E};\Theta,F \vdash F'}{\mathcal{E};\Theta \vdash F \tilde{\Rightarrow} F'}$$

G.L-$\tilde{\Rightarrow}$
$$\frac{\mathcal{E};\Theta \vdash F_0 \qquad \mathcal{E};\Theta,F_1 \vdash F}{\mathcal{E};\Theta,F_0 \tilde{\Rightarrow} F_1 \vdash F}$$

G.R-$\tilde{\wedge}$
$$\frac{\mathcal{E};\Theta \vdash F \qquad \mathcal{E};\Theta \vdash F'}{\mathcal{E};\Theta \vdash F \tilde{\wedge} F'}$$

G.R1-$\tilde{\vee}$
$$\frac{\mathcal{E};\Theta \vdash F}{\mathcal{E};\Theta \vdash F \tilde{\vee} F'}$$

G.R2-$\tilde{\vee}$
$$\frac{\mathcal{E};\Theta \vdash F'}{\mathcal{E};\Theta \vdash F \tilde{\vee} F'}$$

G.R-$\tilde{\top}$
$$\frac{}{\mathcal{E};\Theta \vdash \tilde{\top}}$$

Global judgements: left rules

G.L-$\tilde{\forall}$
$$\frac{\mathcal{E};\Theta,F'\{x \mapsto t\} \vdash F \qquad \mathcal{E} \vdash t:\tau}{\mathcal{E};\Theta,\tilde{\forall}(x:\tau).F' \vdash F}$$

G.L-$\tilde{\exists}$
$$\frac{\mathcal{E},x:\tau;\Theta,F \vdash F'}{\mathcal{E};\Theta,\tilde{\exists}(x:\tau).F \vdash F'}$$

G.L-$\tilde{\wedge}$
$$\frac{\mathcal{E};\Theta,F_0,F_1 \vdash F}{\mathcal{E};\Theta,F_0 \tilde{\wedge} F_1 \vdash F}$$

G.L-$\tilde{\vee}$
$$\frac{\mathcal{E};\Theta,F_0 \vdash F \qquad \mathcal{E};\Theta,F_1 \vdash F}{\mathcal{E};\Theta,F_0 \tilde{\vee} F_1 \vdash F}$$

G.L-$\tilde{\bot}$
$$\frac{}{\mathcal{E};\Theta,\tilde{\bot} \vdash F}$$

Global judgements: other rules

G.Rewrite
$$\frac{\mathcal{E};\Theta \vdash F[s] \qquad \mathcal{E};\Theta \vdash [s = t] \qquad []\text{ does not appear below a } \mathsf{const}(\cdot) \text{ or } \mathsf{adv}(\cdot) \text{ in } F[]}{\mathcal{E};\Theta \vdash F[t]}$$

G.Axiom
$$\frac{}{\mathcal{E};\Theta,F;\Gamma \vdash F}$$

G.Absurd
$$\frac{\mathcal{E};\Theta,F \tilde{\Rightarrow} \tilde{\bot} \vdash \tilde{\bot}}{\mathcal{E};\Theta \vdash F}$$

G.Cut
$$\frac{\mathcal{E};\Theta \vdash F' \qquad \mathcal{E};\Theta,F' \vdash F}{\mathcal{E};\Theta \vdash F}$$

G.Weak
$$\frac{\mathcal{E};\Theta_0 \vdash F}{\mathcal{E};\Theta_0,\Theta_1 \vdash F}$$

Convention: Right rules for universal quantification assume that the introduce variable x is not bound in $\mathcal{E}$. The rewrite rules cannot rewrite at a position that captures free variables in t or s.

Figure 7. Proof system.

Equivalence rules

G.EQUIV:FA-ADV
$$\frac{\mathcal{E};\Theta \vdash \vec{u}_l \sim \vec{u}_r \qquad \mathcal{E};\Theta \vdash \mathsf{adv}(t)}{\mathcal{E};\Theta \vdash \vec{u}_l, t \sim \vec{u}_r, t}$$

G.EQUIV:FA-APP
$$\frac{\mathcal{E};\Theta \vdash \vec{u}_l, t_l, t_l' \sim \vec{u}_r, t_r, t_r'}{\mathcal{E};\Theta \vdash \vec{u}_l, (t_l\ t_l') \sim \vec{u}_r, (t_r\ t_r')}$$

G.EQUIV:FA$_\lambda$-APP
$$\frac{\mathcal{E};\Theta \vdash \begin{array}{l}\vec{u}_l, (\lambda(x:\tau).t_l), (\lambda(x:\tau).t_l') \\ \sim\ \vec{u}_r, (\lambda(x:\tau).t_r), (\lambda(x:\tau).t_r')\end{array}}{\mathcal{E};\Theta \vdash \vec{u}_l, \lambda(x:\tau).(t_l\ t_l') \sim \vec{u}_r, \lambda(x:\tau).(t_r\ t_r')}$$

G.EQUIV:FA-$\mathcal{Q}$
$$\frac{\mathtt{polynomial} \in \mathsf{label}(\tau) \qquad \mathcal{Q} \in \{\forall, \exists\} \\ \mathcal{E};\Theta \vdash \vec{u}_l, \lambda(x:\tau).t_l \sim \vec{u}_r, \lambda(x:\tau).t_r}{\mathcal{E};\Theta \vdash \vec{u}_l, \mathcal{Q}(x:\tau).t_l \sim \vec{u}_r, \mathcal{Q}(x:\tau).t_r}$$

G.EQUIV:CONST-λ
$$\frac{x \notin \mathsf{fv}(t_l) \cup \mathsf{fv}(t_r) \\ \mathcal{E};\Theta \vdash \vec{u}_l, t_l \sim \vec{u}_r, t_r}{\mathcal{E};\Theta \vdash \vec{u}_l, \lambda(x:\tau).t_l \sim \vec{u}_r, \lambda(x:\tau).t_r}$$

Structural rules

G.EQUIV:REFL
$$\frac{}{\mathcal{E};\Theta \vdash \vec{u} \sim \vec{u}}$$

G.EQUIV:SYM
$$\frac{\mathcal{E};\Theta \vdash \vec{v} \sim \vec{u}}{\mathcal{E};\Theta \vdash \vec{u} \sim \vec{v}}$$

G.EQUIV:TRANS
$$\frac{\mathcal{E};\Theta \vdash \vec{u} \sim \vec{w} \qquad \mathcal{E};\Theta \vdash \vec{w} \sim \vec{v}}{\mathcal{E};\Theta \vdash \vec{u} \sim \vec{v}}$$

G.EQUIV:WEAK
$$\frac{\mathcal{E};\Theta \vdash \vec{u}_l, \vec{v}_l \sim \vec{u}_r, \vec{v}_r}{\mathcal{E};\Theta \vdash \vec{u}_l \sim \vec{u}_r}$$

G.EQUIV:DUP
$$\frac{\mathcal{E};\Theta \vdash \vec{u}_l, t_l \sim \vec{u}_r, t_r}{\mathcal{E};\Theta \vdash \vec{u}_l, t_l, t_l \sim \vec{u}_r, t_r, t_r}$$

G.EQUIV:PERM
$$\frac{\pi \text{ permutation of } \{1, \ldots, n\} \\ \mathcal{E};\Theta \vdash u_l^{\pi(1)}, \ldots, u_l^{\pi(n)} \sim u_r^{\pi(1)}, \ldots, u_r^{\pi(n)}}{\mathcal{E};\Theta \vdash u_l^1, \ldots, u_l^n \sim u_r^1, \ldots, u_r^n}$$

G.CASE-STUDY:ITE
$$\frac{\mathcal{E};\Theta \vdash \vec{u}_l, b_l, s_l \sim \vec{u}_r, b_r, s_r \\ \mathcal{E};\Theta \vdash \vec{u}_l, b_l, t_l \sim \vec{u}_r, b_r, t_r}{\mathcal{E};\Theta \vdash \vec{u}_l, \mathsf{if}\ b_l\ \mathsf{then}\ s_l\ \mathsf{else}\ t_l \sim \vec{u}_r, \mathsf{if}\ b_r\ \mathsf{then}\ s_r\ \mathsf{else}\ t_r}$$

Reduction rules *(local version of these rules can be obtained using* L.LOCALISE*)*

G.β
$$\frac{}{\mathcal{E};\Theta \vdash [(\lambda(x:\tau).t)\ t_0 = t\{x \mapsto t_0\}]}$$

G.δ
$$\frac{}{\mathcal{E}, x:\tau = t;\Theta \vdash [x = t]}$$

G.ζ
$$\frac{1 \leq i \leq n}{\mathcal{E};\Theta \vdash [(t_1, \ldots, t_n)\#i = t_i]}$$

Induction rules

L.INDUCTION
$$\frac{\mathtt{well\text{-}founded} \in \mathsf{label}(\tau) \\ \mathcal{E};\Theta;\Gamma \vdash \forall(x:\tau).\,(\forall(x':\tau).\,x' < x \Rightarrow \phi\{x \mapsto x'\}) \Rightarrow \phi}{\mathcal{E};\Theta;\Gamma \vdash \forall(x:\tau).\,\phi}$$

G.INDUCTION
$$\frac{\mathtt{well\text{-}founded} \in \mathsf{label}(\tau) \\ \mathcal{E};\Theta \vdash \tilde{\forall}(x:\tau).\,(\tilde{\forall}(x':\tau).\,[x' < x] \tilde{\Rightarrow} F\{x \mapsto x'\}) \tilde{\Rightarrow} F}{\mathcal{E};\Theta \vdash \tilde{\forall}(x:\tau).\,F}$$

Figure 8. Indistinguishability rules.

terms. In the base logic, terms are interpreted in *computational models* [8], which provide a PPTM for each function symbol (with some constraints discussed below). The interpretation has the same structure as our semantics $[\![t]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho}$ for terms, parameterised by η and random tapes ρ. A striking difference is that random tapes are infinite bitstrings in the base logic, while we consider only finite tapes in the higher-order logic. This is, however, not restrictive: since terms of the base logic are interpreted as machines that run in polynomial time in η, they can only access a finite amount of randomness for each η. We can thus construct from any computational model of the base logic a model in our sense with long enough finite tapes (for each η) such that the semantics of a base logic term is the same in both models.

We have argued so far why a base logic formula that is satisfied in all of our models, is also satisfied in all computational models, *i.e.* valid in the base logic. However, our models allow much more interpretations than the computational models of the base logic, hence the converse is not true[8]. We

argue now under which conditions this can be recovered. Since each honest function symbol f of the base logic can only be interpreted by a deterministic PTIME machine in computational models, we must similarly restrict their interpretation in our models. This can be achieved by adding a global predicate $\mathsf{det}(\cdot)$ expressing that a term has a deterministic interpretation: formally, $[\![\mathsf{det}(t)]\!]_{\mathbb{M}:\mathcal{E}}$ holds when, for any η and for any $\rho, \rho' \in \mathbb{T}_{\mathbb{M},\eta}$, we have $[\![t]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = [\![t]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho'}$. Then, for each honest function symbol f, we must ensure that they are deterministic and computable in polynomial time. This can be expressed by adding an axiom $\mathsf{det}(f) \tilde{\wedge} \mathsf{adv}(f)$. Similarly, we would add $\mathsf{adv}(g)$ for every adversarial function symbol g. To be complete, similar restrictions should be put in place for free variables and quantifiers: we do not detail this step, which is actually not necessary since the meta-logic only makes use of closed and quantifier-free base logic formulas.

We finally discuss the treatment of names, which is the most complex issue. In the base logic, names are interpreted as independent uniform random samplings in $\{0,1\}^\eta$. Names in our logic are not necessarily sampled uniformly, and the probability that two names are equal is not necessarily negligible. The latter problem can be fixed by taking names over the type message with the assumption that this type is `large` (as in Example 8). The former problem may be fixed

---

[8] Assuming that EQ is interpreted as equality, and that c and d are two function symbols of arity 0, the formula $c \sim d \Rightarrow \mathsf{EQ}(c,d) \sim \mathsf{true}$ is valid in the base logic because function symbols are deterministic PTIME machines in the base logic. The same formula is not valid in our logic, unless one postulates a similar condition for c and d.

by further restricting the meaning of names in message. This would be necessary to obtain a conservativity result at the level of validity, however it is likely to be unnecessary to obtain a result at the level of provability, since proof systems for the base logic (and the meta-logic) actually never rely on the uniformity of distributions.

b) Indices and timestamps: We now discuss how the meta-logic can be embedded in our higher-order logic. The construction of the meta-logic on top of the base logic involves the notion of *trace model*. A trace model $\mathbb{T}$ notably fixes the domains of interpretation for the index and timestamp types. Both domains must be finite. Once this is set, a meta-interpretation function translates meta-level terms and local formulas to base-level terms. The meta-interpretation of a meta-logic term $(t)^{\mathbb{T}}$ of sort message replaces indexed names $n\ i$ by base logic names n_v (where v is the value of i in $\mathbb{T}$) in a larger set of base logic names (which depends on $\mathbb{T}$). The meta-interpretation of a local meta-logic formula $(\phi)^{\mathbb{T}}$ (which is a base logic boolean term) replaces quantifications over timestamps and indices by finite boolean combinations, and replaces comparisons of timestamps and indices by their values (boolean constants). Once base logic terms are obtained by these meta-interpretations, they are further interpreted as PPTM according to a computational model.

In our new logic, models must play both the role of the trace and computational models of the former approach. We must thus constrain the interpretation of the types index and timestamp, but also the interpretation of the associated terms: in the meta-logic, terms of these types are interpreted at the meta-level and are thus independent of η and ρ. Hence, we work in the new logic with types index and timestamp tagged as both `finite` and `fixed`, and we restrict global quantifications over timestamps and indices to constant values[9]: $\tilde{\forall}(x:\tau).F$ becomes $\tilde{\forall}(x:\tau). \mathsf{const}(x) \stackrel{\sim}{\Rightarrow} F$, and dually for $\tilde{\exists}$. Such a transformation is not needed for local quantifications, since they quantify over random variables $\mathbb{1}_a^\eta$ in our semantics.

Indices in the meta-logic are unstructured: the only terms of sort index are variables, and they can only be compared for equality. In contrast, timestamps can be built using indexed constants (representing protocol actions) and a predecessor function, and they can be compared using an order. Moreover, the interpretation of this order in trace models must respect a partial order indicating sequential dependencies between protocol actions. All these constraints can be axiomatised in the new logic.

We finally address the *find* construct of meta-logic terms. Roughly, the term $(\mathsf{find}\ \vec{i}\ \mathsf{suchthat}\ \phi\ \mathsf{in}\ t\ \mathsf{else}\ t')$ attempts to find values for $\vec{i}$ such that ϕ holds, evaluates as t (which may rely on $\vec{i}$) upon success, and t' otherwise. Although this is not immediately available in the meta-logic, axiomatising the behaviour of this construct is quite natural. In fact, it is beneficial to do so by decomposing it using a *choice* operator (essentially Hilbert's epsilon operator) which associates to

any predicate a value that makes it true, if it exists. We have introduced this operator, and associated axioms, in our standard library, enabling easier and sometimes more precise reasoning about quantifiers and *find* constructs.

c) Protocols and macros: The syntax and semantics of the meta-logic are parameterised by protocols, which must all share the same set of partially-ordered actions. These actions give rise to the language of actions on which timestamps are built, and the partial-order constrains trace models; we have seen that these aspects can be handled axiomatically. Next, macros are used in the meta-logic to reflect the semantics of protocols: for each action, the protocol specifies an executability condition, some state updates, and an output message. All these are represented in the logic by built-in macros of the form $m@T$ where $m \in \{\mathsf{cond}, \mathsf{output}, \ldots\}$ and T is a timestamp. The semantics of macros is given as part of the meta-interpretation: once a trace model $\mathbb{T}$ is fixed we can define the meaning of macros by induction on timestamps.

In our new logic, this is replaced by recursive definitions. For each macro m and each protocol of interest $\mathcal{P}$, we define the meaning of m w.r.t. $\mathcal{P}$ at a timestamp T as a recursive definition $m_\mathcal{P}$ which takes a timestamp as argument: this is illustrated in Example 6. Obviously, this approach decouples the logic from the specific notion of protocol: we could use the logic to model a different kind of protocol or a different attacker model. Less obviously, this encoding of macros also changes the granularity of protocol annotations: in the meta-logic, it is not possible to mix, in the same term, macros w.r.t. several protocols. This is made possible in the style proposed here, and has practical applications[10].

APPENDIX E
SOUNDNESS OF THE PROOF SYSTEM

A. Measure Theory: Standard Definitions

We recall some standard measure theory definitions.

For any set $\mathbb{S}$, we let $\mathcal{P}(\mathbb{S})$ be the power set of $\mathbb{S}$. A *σ-algebra* Σ over a set $\mathbb{S}$ is a non-empty subset of $\mathcal{P}(\mathbb{S})$ closed under countable unions and intersections. For any set $\mathbb{S}$, the *discrete σ-algebra* on $\mathbb{S}$ is the power set of $\mathbb{S}$. This is the finest σ-algebra on $\mathbb{S}$. A set $\mathbb{S}$ equipped with a σ-algebra Σ on $\mathbb{S}$ is called a *measurable space*. A *measurable* function X from a measurable space $(\mathbb{S}_1, \Sigma_1)$ to a measurable $(\mathbb{S}_2, \Sigma_2)$ is a function $X : \mathbb{S}_1 \to \mathbb{S}_2$ such that $\forall A \in \Sigma_2. X^{-1}(A) \in \Sigma_1$. Notice that if Σ_1 is the discrete σ-algebra, then any function from $(\mathbb{S}_1, \Sigma_1)$ to any measurable space is measurable.

A *measure space* $(\mathbb{S}, \Sigma, \mu)$ is a measurable space $(\mathbb{S}, \Sigma)$ equipped with a *measure* μ, where a measure is a function from Σ to $\mathbb{R}$ such that: i) μ is non-negative, *i.e.* $\forall A \in \Sigma, \mu(A) \geq 0$; ii) $\mu(\emptyset) = 0$; and iii), μ is σ-additive, *i.e.* for all pairwise disjoint countable family $(A_i)_{i \in \mathbb{N}}$ of elements in Σ, $\mu(\bigcup_{i \in \mathbb{N}} A_i) = \sum_{i \in \mathbb{N}} \mu(A_i)$.

---

[9] In our tool, these $\mathsf{const}(x)$ constraints are handled as constraints attached to variables, which allowed for a smooth transition from the meta-logic to the new logic, and also avoids problems with substitution inside this predicate.

[10] To take a simple example, one might wish to declare the meta-logic equivalence $(u \sim v)_{\mathcal{P}, \mathcal{Q}}$ as being true by reflexivity when the local formula $u_\mathcal{P} = v_\mathcal{Q}$ is valid, but this requires to be able to consider the subterms u and v w.r.t. distinct protocols.

A *probability space* $(\mathbb{S}, \Sigma, \mu)$ is a measure space such that $\mu(\mathbb{S}) = 1$. A measurable function X from a probability space $(\mathbb{S}_1, \Sigma_1, \mu_1)$ to $(\mathbb{S}_2, \Sigma_2)$ induces a probability measure $\mu \circ X^{-1}$ on $\mathbb{S}_2$, for which we use the standard notation

$$\Pr_{s \in \mathbb{S}_1} (X(s) \in A) \overset{\text{def}}{=} \mu_1(X^{-1}(A)) \qquad (\forall A \in \Sigma_2)$$

B. Preliminaries

For any η-indexed families of reals A and B, we write $A =_{\mathsf{ow}} B$ whenever A and B are overwhelmingly equal, *i.e.* if $(A_\eta - B_\eta)_{\eta \in \mathbb{N}} \in \mathsf{negl}(\eta)$.

We lift this to events as follows: let E and E' be two η-indexed families events on the same probability space Ω. We say that E and E' have overwhelmingly equal probabilities, written $E =_{\mathsf{ow}} E'$, if

$$(\Pr_\Omega(E_\eta))_{\eta \in \mathbb{N}} =_{\mathsf{ow}} (\Pr_\Omega(E'_\eta))_{\eta \in \mathbb{N}}$$

C. Soundness of the Core Rules

Proposition 9. *Let ϕ be a local formula such that:*

$$\mathcal{E} \vdash \forall(x : \tau). \phi : \mathsf{bool}$$

Let $\mathbb{M} : \mathcal{E}$ be a model and $A, B \in \mathbb{RV}_{\mathbb{M}}(\tau)$. Then for any $\eta \in \mathbb{N}$ and $\rho \in \mathbb{T}_{\mathbb{M}, \eta}$, if A and B coincide on η, ρ (i.e. $A(\eta)(\rho) = B(\eta)(\rho)$) then

$$\llbracket \phi \rrbracket_{\mathbb{M}[x \mapsto A]:(\mathcal{E}, x:\tau)}^{\eta,\rho} = \llbracket \phi \rrbracket_{\mathbb{M}[x \mapsto B]:(\mathcal{E}, x:\tau)}^{\eta,\rho}$$

Furthermore, $B = \mathbb{1}_{A(\eta)(\rho)}^{\eta}$ coincides with A for any $A \in \mathbb{RV}_{\mathbb{M}}(\tau)$.

Proof. By an easy induction over ϕ. $\qquad \square$

We now recall and prove Proposition 2.

Proposition 2. *Let $\mathcal{E}$ be a well-formed environment and ϕ be a local formula such that $\mathcal{E} \vdash \forall(x : \tau). \phi : \mathsf{bool}$. For every model $\mathbb{M}$ of $\mathcal{E}$, we have:*

$$\mathbb{M} : \mathcal{E} \models [\forall(x : \tau). \phi] \qquad \textit{iff} \qquad \mathbb{M} : \mathcal{E} \models \tilde{\forall}(x : \tau). [\phi]$$

Proof. We prove both directions separately.
$\Rightarrow$ **case.** Assume the following:

$$\mathbb{M} : \mathcal{E} \models [\forall(x : \tau). \phi] \qquad (4)$$

Let $A \in \mathbb{RV}_{\mathbb{M}}(\tau)$ be a η-indexed sequence of random variables. We need to show that

$$\Pr \left(\llbracket \phi \rrbracket_{\mathbb{M}[x \mapsto A]:(\mathcal{E}, x:\tau)}^{\eta,\rho} \right) \in \mathsf{ow}(\eta) \qquad (5)$$

where the probability is over $\rho \in \mathbb{T}_{\mathbb{M}, \eta}$.

$$\begin{aligned}
&\Pr \left(\llbracket \phi \rrbracket_{\mathbb{M}[x \mapsto A]:(\mathcal{E}, x:\tau)}^{\eta,\rho} \right) \\
&= \Pr \left(\llbracket \phi \rrbracket_{\mathbb{M}[x \mapsto \mathbb{1}_{A(\eta)(\rho)}^{\eta}]:(\mathcal{E}, x:\tau)}^{\eta,\rho} \right) \\
&\qquad\qquad\qquad\qquad \text{(By Proposition 9)} \\
&\geq \Pr \left(\bigcap_{a \in \llbracket \tau \rrbracket_{\mathbb{M}}^{\eta}} \llbracket \phi \rrbracket_{\mathbb{M}[x \mapsto \mathbb{1}_a^{\eta}]:(\mathcal{E}, x:\tau)}^{\eta,\rho} \right) \\
&= \Pr \left(\llbracket \forall(x : \tau). \phi \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} \right) \\
&\in \mathsf{ow}(\eta) \qquad\qquad\qquad\qquad \text{(using Eq. (4))}
\end{aligned}$$

This concludes the proof of Eq. (5).

$\Leftarrow$ **case.** Assume that

$$\mathbb{M} : \mathcal{E} \models \tilde{\forall}(x : \tau). [\phi] \qquad (6)$$

We need to show that

$$\Pr \left(\llbracket \forall(x : \tau). \phi \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} \right) \in \mathsf{ow}(\eta) \qquad (7)$$

where the probability is over $\rho \in \mathbb{T}_{\mathbb{M}, \eta}$.

Let A be the η-indexed family of functions choosing, for any η and ρ, a value $a \in \llbracket \tau \rrbracket_{\mathbb{M}}^{\eta}$ making ϕ false when evaluated on tape ρ

$$A(\eta)(\rho) \overset{\text{def}}{=}$$
$$\begin{cases} \mathsf{choose}\{a \in \llbracket \tau \rrbracket_{\mathbb{M}}^{\eta} \mid \llbracket \neg\phi \rrbracket_{\mathbb{M}[x \mapsto \mathbb{1}_a^{\eta}]:(\mathcal{E}, x:\tau)}^{\eta,\rho}\} & \text{if non-empty} \\ a_{\mathsf{witness}} & \text{otherwise} \end{cases}$$

where a_{witness} is an arbitrary value in $\llbracket \tau \rrbracket_{\mathbb{M}}^{\eta}$ (recall that our semantics requires that $\llbracket \tau \rrbracket_{\mathbb{M}}^{\eta} \neq \emptyset$), and $\mathsf{choose}(\mathbb{S})$ is an arbitrary choice function for set $\mathbb{S}$.

Since all functions from $\mathbb{T}_{\mathbb{M}, \eta}$ to $\{0; 1\}$ are random variables w.r.t. the discrete σ-algebra on $\mathbb{T}_{\mathbb{M}, \eta}$ (thanks to $\mathbb{T}_{\mathbb{M}, \eta}$'s finiteness), we get that, by applying Eq. (6) to A

$$\Pr \left(\llbracket \phi \rrbracket_{\mathbb{M}[x \mapsto A]:(\mathcal{E}, x:\tau)}^{\eta,\rho} \right) \in \mathsf{ow}(\eta) \qquad (8)$$

Then

$$\begin{aligned}
&\Pr \left(\llbracket \phi \rrbracket_{\mathbb{M}[x \mapsto A]:(\mathcal{E}, x:\tau)}^{\eta,\rho} \right) \\
&= \Pr \left(\llbracket \phi \rrbracket_{\mathbb{M}[x \mapsto \mathbb{1}_{A(\eta)(\rho)}^{\eta}]:(\mathcal{E}, x:\tau)}^{\eta,\rho} \right) \quad \text{(By Proposition 9)} \\
&= \Pr \left(\bigcap_{a \in \llbracket \tau \rrbracket_{\mathbb{M}}^{\eta}} \llbracket \phi \rrbracket_{\mathbb{M}[x \mapsto \mathbb{1}_a^{\eta}]:(\mathcal{E}, x:\tau)}^{\eta,\rho} \right) \\
&= \Pr \left(\llbracket \forall(x : \tau). \phi \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} \right) \\
&\in \mathsf{ow}(\eta) \qquad\qquad\qquad\qquad \text{(using Eq. (8))}
\end{aligned}$$

which concludes the proof of Eq. (7) $\qquad \square$

Remark 5. *Proposition 2 establishes that universal quantification commutes with $[\cdot]$. The same result can be shown, with the same proof, for existential quantification.*

This may seem surprising, since, on the other hand, disjunction does not commute with $[\cdot]$: $[\phi_0 \lor \phi_1]$ does not in general imply $[\phi_0] \tilde{\lor} [\phi_1]$.

This apparent contradiction comes from the fact that, contrary to intuition, global existential quantification is not the same as global disjunction. Indeed, global $\tilde{\exists}$ can take as a witness a random variable, while a disjunction forces to take a constant witness, and is thus a stronger condition. Basically, $[\phi_0] \tilde{\lor} [\phi_1]$ means that there exists a constant b, independent from η and ρ, such that $[\phi_b]$. On the other hand, $\tilde{\exists}b. [\phi_b]$ means that there exists a random variable b, that may depend on η and ρ, such that $[\phi_b]$, which is weaker.

There is then no contradiction: $[\phi_0 \lor \phi_1]$, which is equivalent to $[\exists b. \phi_b]$, implies the weaker $\tilde{\exists}b. [\phi_b]$ but not $[\phi_0] \tilde{\lor} [\phi_1]$.

We recall and prove Proposition 4.

Proposition 4. *Let $\mathcal{E}$ be a well-formed environment and ϕ a local formula such that $\mathcal{E} \vdash \forall(x : \tau).\phi : \mathsf{bool}$ and $\{\mathtt{fixed}, \mathtt{finite}\} \subseteq \mathsf{label}(\tau)$. For every model $\mathbb{M} : \mathcal{E}$:*

$$\mathbb{M} : \mathcal{E} \models [\forall(x : \tau).\phi] \quad \textit{iff} \quad \mathbb{M} : \mathcal{E} \models \tilde{\forall}(x : \tau).\,\mathsf{const}(x) \tilde{\Rightarrow} [\phi]$$

Proof. Using Proposition 2, we know that

$$\mathbb{M} : \mathcal{E} \models [\forall(x : \tau).\phi] \qquad \text{iff.} \qquad \mathbb{M} : \mathcal{E} \models \tilde{\forall}(x : \tau).\,[\phi]$$

To conclude, we are going to show that

$$\mathbb{M} : \mathcal{E} \models \tilde{\forall}(x : \tau).\,[\phi] \quad \text{iff.} \quad \mathbb{M} : \mathcal{E} \models \tilde{\forall}(x : \tau).\,\mathsf{const}(x) \tilde{\Rightarrow} [\phi]$$

The left-to-right direction is obvious. To prove the other direction, assume that

$$\mathbb{M} : \mathcal{E} \models \tilde{\forall}(x : \tau).\,\mathsf{const}(x) \tilde{\Rightarrow} [\phi] \tag{9}$$

holds, and let $A \in \mathbb{RV}_{\mathbb{M}}(\tau)$ be a η-indexed family of random variables. We must prove that

$$\Pr\left([\![\phi]\!]^{\eta,\rho}_{\mathbb{M}[x \mapsto A]:(\mathcal{E},x:\tau)}\right) \in \mathsf{ow}(\eta)$$

where the probability is over $\rho \in \mathbb{T}_{\mathbb{M},\eta}$.

Since $\mathtt{fixed} \in \mathsf{label}(\tau)$, we know that there exists $\mathbb{S}$ such that $[\![\tau]\!]^{\eta}_{\mathbb{M}} = \mathbb{S}$ for every $\eta \in \mathbb{N}$. For any $a \in \mathbb{S}$, we let B_a be the η-indexed sequence of *constant* random variables always taking value a. Then

$$\begin{aligned}
\Pr\left([\![\phi]\!]^{\eta,\rho}_{\mathbb{M}[x \mapsto A]:(\mathcal{E},x:\tau)}\right) \\
= \Pr\left([\![\phi]\!]^{\eta,\rho}_{\mathbb{M}[x \mapsto B_{A(\eta)(\rho)}]:(\mathcal{E},x:\tau)}\right) \\
\text{(By Proposition 9)} \\
\geq \Pr\left(\bigcap_{a \in [\![\tau]\!]^{\eta}_{\mathbb{M}}} [\![\phi]\!]^{\eta,\rho}_{\mathbb{M}[x \mapsto B_a]:(\mathcal{E},x:\tau)}\right)
\end{aligned}$$

For any $a \in [\![\tau]\!]^{\eta}_{\mathbb{M}}$, B_a is constant, hence by (9) we get

$$\Pr\left([\![\phi]\!]^{\eta,\rho}_{\mathbb{M}[x \mapsto B_a]:(\mathcal{E},x:\tau)}\right) \in \mathsf{ow}(\eta)$$

Since the conjunction of a collection of N (finite, independent from η) η-indexed families of boolean random variables $(X^1_\eta)_{\eta \in \mathbb{N}}, \ldots, (X^N_\eta)_{\eta \in \mathbb{N}}$ is overwhelmingly true whenever the random variables $(X^1_\eta)_{\eta \in \mathbb{N}}, \ldots, (X^N_\eta)_{\eta \in \mathbb{N}}$ are all overwhelmingly true, we know that

$$\Pr\left(\bigcap_{a \in [\![\tau]\!]^{\eta}_{\mathbb{M}}} [\![\phi]\!]^{\eta,\rho}_{\mathbb{M}[x \mapsto B_a]:(\mathcal{E},x:\tau)}\right) \in \mathsf{ow}(\eta)$$

which concludes this proof. ☐

Proposition 10. *The rules in Fig. 7 are sound.*

Proof (sketch). We details the proof of some of the rules below

- For the mixed-judgements rules.
  The L.BYGLOB rule is trivial (the premise and conclusion have the same semantics).
  The G.BYLOC rule is essentially a local weakening rule, forgetting all local hypotheses
  The L.REWRITE-EQUIV follows from the fact that $\mathbb{M} \models [\phi]$ iff $\mathbb{M} \models \phi \sim \mathsf{true}$, with a transitivity step through $(\Gamma_0 \Rightarrow \phi_0) \sim (\Gamma_1 \Rightarrow \phi_1)$.

The L.LOCALISE rule is admissible from L.CUT-LOC and L.BYGLOB:

$$\frac{\dfrac{\mathcal{E};\Theta,[\phi] \vdash [\phi] \;\; \text{L.AXIOM}}{\mathcal{E};\Theta,[\phi];\Gamma \vdash \phi} \text{L.BYGLOB} \qquad \dfrac{\mathcal{E};\Theta;\Gamma,\phi \vdash \psi}{\mathcal{E};\Theta,[\phi];\Gamma,\phi \vdash \psi} \text{L.WEAK}}{\mathcal{E};\Theta,[\phi];\Gamma \vdash \psi} \text{L.CUT-LOC}$$

- Local quantifiers rules soundness follows from Proposition 2 for L.R-$\forall$, and Proposition 4 for L.R-$\forall$-CONST. For example, for the L.R-$\forall$ rule, let $\mathbb{M}$ be a model of $\mathcal{E}$, then

$$\begin{aligned}
& [\![(\tilde{\wedge}\Theta) \tilde{\Rightarrow} [(\wedge\Gamma) \Rightarrow \forall(x : \tau).\phi]]\!]_{\mathbb{M}:\mathcal{E}} \\
\text{iff } & [\![(\tilde{\wedge}\Theta) \tilde{\Rightarrow} [\forall(x : \tau).(\wedge\Gamma) \Rightarrow \phi]]\!]_{\mathbb{M}:\mathcal{E}} \\
\text{iff } & [\![(\tilde{\wedge}\Theta) \tilde{\Rightarrow} \tilde{\forall}(x : \tau).[(\wedge\Gamma) \Rightarrow \phi]]\!]_{\mathbb{M}:\mathcal{E}} \\
& \qquad\qquad\qquad\qquad \text{(by Proposition 2)}
\end{aligned}$$

where we alpha-renamed x if necessary to avoid capture.
- For the L.L-$\exists$ rule

$$\frac{\mathcal{E}, x : \tau;\Theta;\Gamma,\phi \vdash \psi}{\mathcal{E};\Theta;\Gamma,\exists(x : \tau).\phi \vdash \psi}$$

Let $\mathbb{M} : \mathcal{E}$, then

$$\begin{aligned}
& \mathbb{M} \models (\tilde{\wedge}\Theta) \tilde{\Rightarrow} [(\wedge\Gamma) \Rightarrow (\exists(x : \tau).\phi) \Rightarrow \psi] \\
\text{iff } & \mathbb{M} \models (\tilde{\wedge}\Theta) \tilde{\Rightarrow} [\forall(x : \tau).(\wedge\Gamma) \Rightarrow \phi \Rightarrow \psi] \\
\text{iff } & \mathbb{M} \models (\tilde{\wedge}\Theta) \tilde{\Rightarrow} \tilde{\forall}(x : \tau).[(\wedge\Gamma) \Rightarrow \phi \Rightarrow \psi] \\
& \qquad\qquad\qquad\qquad \text{(By Proposition 2)} \\
\text{iff } & \mathbb{M} \models \tilde{\forall}(x : \tau).(\tilde{\wedge}\Theta) \tilde{\Rightarrow} [(\wedge\Gamma) \Rightarrow \phi \Rightarrow \psi]
\end{aligned}$$

Which concludes the proof of L.L-$\exists$.
- For the other local judgement right rules and local judgement left local rules, we exploits the particular semantics of terms interpreting boolean connectives and quantifiers as their usual first-order counter-parts. We omit the details.
- Our global logic is a first-order logic with higher-order terms with a semantics which only considers model of a particular form. Consequently, all usual first-order logic rules are sound for this logic. This covers the soundness of all the global judgements left and right rules (plus the G.AXIOM, G.ABSURD, G.CUT and G.WEAK rules), as well as the local judgements left global rules. ☐

D. Soundness of the Indistinguishability Rules

Proposition 11. *The rules in Fig. 8 are sound.*

Proof (sketch). The structural rules are quite straightforward (details of the proofs in the original CCSA logic can be found in [8]).

The reduction and induction rules are immediate, we omit the details.

We only detail one FA (Function Application rules), which deals with λ-abstraction and is related to the new higher-order aspect of our logic.

- G.EQUIV:FA$_\lambda$-APP

Assume that the following judgment is valid

$$\mathcal{E}; \Theta \vdash \begin{array}{l} \vec{u}_l, (\lambda(x:\tau).\, t_l), (\lambda(x:\tau).\, t_l') \\ \sim \vec{u}_r, (\lambda(x:\tau).\, t_r), (\lambda(x:\tau).\, t_r') \end{array} \quad (10)$$

We must show that

$$\mathcal{E}; \Theta \vdash \vec{u}_l, \lambda(x:\tau).\, (t_l \ t_l') \sim \vec{u}_r, \lambda(x:\tau).\, (t_r \ t_r') \quad (11)$$

is a valid judgement.

Let $\mathbb{M}$ be a model such that $\mathbb{M} \models \Theta$, and let $\mathcal{A} \in \mathsf{PPTMs}$ be an adversary against the conclusion of Eq. (11). We must show that

$$\mathsf{Adv}_{\mathbb{M}:\mathcal{E}}^{\eta}(\mathcal{A} : \vec{u}_l, \lambda(x:\tau).\, (t_l \ t_l') \sim \vec{u}_r, \lambda(x:\tau).\, (t_r \ t_r'))$$
$$\in \mathsf{negl}(\eta)$$

We prove this by reduction: we are going to build an adversary $\mathcal{B} \in \mathsf{PPTMs}$ against the conclusion of Eq. (10) such that $\mathcal{B}$'s advantage is exactly $\mathcal{A}$'s. Since, by hypothesis, Eq. (10) is a valid judgement, we know that $\mathcal{B}$'s advantage is negligible, which concludes this proof.

We now define $\mathcal{B}$. The machine $\mathcal{B}$ receives $\vec{u}_{\mathsf{LR}}$ (where LR is l on the left and r on the right), and can make queries to two higher-order terms (through the handlers it received):

$$\lambda(x:\tau).\, t_{\mathsf{LR}} \quad (12)$$
$$\lambda(x:\tau).\, t_{\mathsf{LR}}' \quad (13)$$

$\mathcal{B}$ is going to simulate the Turing machine $\mathcal{A}$ on input

$$\vec{u}_{\mathsf{LR}}, \lambda(x:\tau).\, t_{\mathsf{LR}} \ t_{\mathsf{LR}}'$$

To do this, $\mathcal{B}$ must be able to answer to $\mathcal{A}$'s queries to the function ($\mathcal{B}$ already know the rest of the inputs $\vec{u}_{\mathsf{LR}}$ which it directly gives to $\mathcal{A}$)

$$\lambda(x:\tau).\, t_{\mathsf{LR}} \ t_{\mathsf{LR}}'$$

Whenever $\mathcal{A}$ queries this function on an input x, $\mathcal{B}$ proceeds as follows:

– $\mathcal{B}$ queries the two functions Eq. (12) and Eq. (13) on input x, which gives back the values f and x';
– $\mathcal{B}$ returns to $\mathcal{B}$ the value $f \ x'$.

$\mathcal{B}$ perfectly simulates $\mathcal{A}$'s execution, hence both machine have exactly the same advantage against their respective game. Finally, it is clear that $\mathcal{B}$'s running time is polynomial whenever $\mathcal{A}$'s running time is, which conclude this proof. □

E. Soundness of the Additional Rules for Mixed Sequents

We provide some additional mixed-judgements rules:

G.L-LOC:∀
$$\frac{\mathcal{E}; \Theta, [\phi\{x \mapsto t\}] \vdash F \qquad \mathcal{E} \vdash t : \tau}{\mathcal{E}; \Theta, [\forall(x:\tau).\phi] \vdash F}$$

G.L-LOC:∧
$$\frac{\mathcal{E}; \Theta, [\phi], [\psi] \vdash F}{\mathcal{E}; \Theta, [\phi \wedge \psi] \vdash F}$$

G.L-LOC:false
$$\frac{}{\mathcal{E}; \Theta, [\mathsf{false}] \vdash F}$$

We now prove that these two rules, and the rules in Fig. 3, are sound.

Proof.
• The G.L-LOC:∧ rule is admissible from G.CUT, G.BYLOC, L.LOCALISE, and its local logic counter-part:

$$\frac{\Pi_1 \quad \Pi_2 \quad \mathcal{E}; \Theta, [\phi], [\psi] \vdash F}{\mathcal{E}; \Theta, [\phi \wedge \psi] \vdash F} \ \text{G.CUT} + \text{G.WEAK}$$

$$\Pi_1 \overset{\mathsf{def}}{=} \frac{\cfrac{\cdots}{\cfrac{\mathcal{E}; \Theta; \phi \wedge \psi \vdash \phi}{\cfrac{\mathcal{E}; \Theta, [\phi \wedge \psi]; \vdash \phi}{\mathcal{E}; \Theta, [\phi \wedge \psi] \vdash [\phi]} \ \text{G.BYLOC}} \ \text{L.LOCALISE}}$$

$$\Pi_2 \overset{\mathsf{def}}{=} \frac{\cfrac{\cdots}{\cfrac{\mathcal{E}; \Theta; \phi \wedge \psi \vdash \phi}{\cfrac{\mathcal{E}; \Theta, [\phi \wedge \psi]; \vdash \psi}{\mathcal{E}; \Theta, [\phi \wedge \psi] \vdash [\psi]} \ \text{G.BYLOC}} \ \text{L.LOCALISE}}$$

• The G.L-LOC:∀ rule is admissible from G.CUT, G.BYLOC, L.LOCALISE, and its local logic counter-part:

$$\frac{\Pi_1 \quad \mathcal{E}; \Theta, [\phi\{x \mapsto t\}] \vdash F}{\mathcal{E}; \Theta, [\forall(x:\tau).\phi] \vdash F} \ \text{G.CUT} + \text{G.WEAK}$$

$$\Pi_1 \overset{\mathsf{def}}{=} \frac{\cfrac{\cdots}{\cfrac{\mathcal{E}; \Theta; \forall(x:\tau).\phi \vdash \phi\{x \mapsto t\}}{\cfrac{\mathcal{E}; \Theta, [\forall(x:\tau).\phi]; \vdash \phi\{x \mapsto t\}}{\mathcal{E}; \Theta, [\forall(x:\tau).\phi] \vdash [\phi\{x \mapsto t\}]} \ \text{G.BYLOC}} \ \text{L.LOCALISE}}$$

• Finally, we have the rules G.L-LOC:∨ and G.R-LOC:→ requiring an additional constant condition. We prove the latter one.
Consider a model $\mathbb{M}$ of Θ. By the validity of the second premise, we know that either (a) $\llbracket \phi \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = 0$ for all η, ρ or (b) $\llbracket \phi \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = 1$ for all η, ρ. In case (a) we have $\llbracket \phi \Rightarrow \psi \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = 1$ for all η, ρ, thus $\mathbb{M} \models [\phi \Rightarrow \psi]$. In case (b) we have $\mathbb{M} \models [\phi]$, thus $\mathbb{M} \models [\psi]$ by the first premise, hence $\mathbb{M} \models [\phi \Rightarrow \psi]$. □

Remark 6. *Note that our semantics for global formulas states that $\mathsf{const}(t)$ holds if there exists c such that $\llbracket t \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho}$ and c are always equal, and not just equal with overwhelming probability. At a first glance, the latter seems more in line with the rest of our semantics, e.g. it would allow to rewrite an global equality $[t_0 = t_1]$ in an atom $\mathsf{const}(C[t_0])$. Unfortunately, this more lenient semantics for $\mathsf{const}(\cdot)$ is not strong enough for our purpose, because the property of being overwhelmingly equal is not stable by arbitrary composition.*

F. Soundness of Rules for Constant

We now recall and prove Proposition 5.

Proposition 5. *The rules in Fig. 4 are sound.*

Proof. The soundness of the L.R-∀-CONST rule is straightforward using Proposition 4.

The G.CONST:APP rule is trivial, we omit its proof.

The only rule remaining is the quantifier rule G.CONST:QUANT. We only prove the $\mathcal{Q} = \lambda$ case (the other cases are identical).

Let $\mathbb{M}$ be a model of $\mathcal{E}$ such that $\mathbb{M} : \mathcal{E} \models \Theta$. By hypothesis

$$\mathbb{M} : \mathcal{E} \models \tilde{\forall}(x : \tau). \, \mathsf{const}(x) \tilde{\Rightarrow} \mathsf{const}(t) \tag{14}$$

Since $\texttt{fixed} \in \mathsf{label}(\tau)$, we know that there exists $\mathbb{S}$ such that $[\![\tau]\!]_{\mathbb{M}}^{\eta} = \mathbb{S}$ for every $\eta \in \mathbb{N}$. For any $a \in \mathbb{S}$, we let B_a be the η-indexed sequence of *constant* random variables always taking value a. It is immediate that $\mathsf{const}(x)$ is satisfied by any model interpreting x as B_a. Hence, from Eq. (14), we know that

$$\mathbb{M}[x \mapsto B_a], (\mathcal{E}, x : \tau) \models \mathsf{const}(t)$$

Consequently, let c_a be such that

$$[\![t]\!]_{\mathbb{M}[x \mapsto B_a]:(\mathcal{E}, x:\tau)}^{\eta, \rho} = c_a \qquad \text{(for any } \eta \in \mathbb{N}, \rho \in \mathbb{T}_{\mathbb{M}, \eta})$$

Then

$$
\begin{aligned}
[\![\lambda(x : \tau). \, t]\!]_{\mathbb{M}:\mathcal{E}}^{\eta, \rho} &= \left(a \in [\![\tau]\!]_{\mathbb{M}}^{\eta} \mapsto [\![t]\!]_{\mathbb{M}[x \mapsto B_a]:(\mathcal{E}, x:\tau)}^{\eta, \rho}\right) \\
&= \left(a \in [\![\tau]\!]_{\mathbb{M}}^{\eta} \mapsto c_a\right)
\end{aligned}
$$

Hence taking $c \overset{\text{def}}{=} \left(a \in [\![\tau]\!]_{\mathbb{M}}^{\eta} \mapsto c_a\right)$ proves that

$$\mathbb{M} : \mathcal{E} \models \mathsf{const}(\lambda(x : \tau). \, t) \qquad \square$$

G. Additional Admissible Rules

We present in Fig. 9 two rules which are straightforward generalizations, in our new formalism, of rules of [14]. We then show that these rather complex rules are admissible from our (simpler) rules.

The G.EQUIV:SPLIT-λ rule of Fig. 9 allows to split a function according to some arbitrary meta-formulas ϕ_l and ϕ_r, using the terms dflt_l and dflt_r in the else branches. This is a generalization of the SPLIT-SEQ rule of [14], which only allowed the rule to apply to functions over finite type, and did not allow the user to choose the else branch (which, moreover, had to be identical on both side of the equivalence). To allow arbitrary else branches, we must leak dflt_l and dflt_r to the adversary in the premise. Note that we do not need to provide the adversary with the branching condition ϕ_l and ϕ_r.

The G.EQUIV:CONST-MANY-λ rule of Fig. 8 allows to replace function terms $\lambda(x : \tau). \, t_l$ and $\lambda(x : \tau). \, t_r$ by a finite collection of terms $s_l^1, \dots, s_l^n$ and $s_r^1, \dots, s_r^n$ whenever we can show that t_l and t_r are always equal to one of the terms in this finite set of possibilities, according to some conditions $\phi^1, \dots, \phi^n$ computable by the adversary.

Similarly to the previous rule, this is a generalization of the CONST-SEQ rule of [14], which only applies to finite types τ, and if the the conditions $\phi^1, \dots, \phi^n$ are constant.

Proposition 12. *The rules in Fig. 9 are sound.*

Proof.

- G.EQUIV:SPLIT-λ First, observe that for every boolean ϕ and values x and dflt, we have the equality

$$\mathcal{E}; \Theta \vdash \tilde{\forall}(x : \tau). \, x = \big(\text{if } x_2 = \mathsf{dflt} \text{ then } x_1 \text{ else } x_2\big) \tag{15}$$

when $x_1 \overset{\text{def}}{=}$ if ϕ then x and $x_2 \overset{\text{def}}{=}$ if $\neg\phi$ then x.

Assume that the following judgment is valid

$$
\mathcal{E}; \Theta \vdash
\begin{aligned}
& \vec{u}_l, \mathsf{dflt}_l, \lambda(x : \tau). \, \text{if } \quad \phi_l \, x \text{ then } t_l \text{ else } \mathsf{dflt}_l, \\
& \qquad\qquad \lambda(x : \tau). \, \text{if } \neg\phi_l \, x \text{ then } t_l \text{ else } \mathsf{dflt}_l \\
& \sim \vec{u}_r, \mathsf{dflt}_r, \lambda(x : \tau). \, \text{if } \quad \phi_r \, x \text{ then } t_r \text{ else } \mathsf{dflt}_r, \\
& \qquad\qquad \lambda(x : \tau). \, \text{if } \neg\phi_r \, x \text{ then } t_r \text{ else } \mathsf{dflt}_r
\end{aligned}
$$

We must show that

$$\mathcal{E}; \Theta \vdash \vec{u}_l, \lambda(x : \tau). \, t_l \sim \vec{u}_r, \lambda(x : \tau). \, t_r$$

is a valid judgement.
First, we use G.REWRITE with Eq. (15) to get

$$
\mathcal{E}; \Theta \vdash
\begin{aligned}
& \vec{u}_l, \lambda(x : \tau). \, \text{if } x_2^l = \mathsf{dflt}_l \text{ then } x_1^l \text{ else } x_2^l \\
& \sim \vec{u}_r, \lambda(x : \tau). \, \text{if } x_2^r = \mathsf{dflt}_r \text{ then } x_1^r \text{ else } x_2^r
\end{aligned}
$$

where $x_1^l \overset{\text{def}}{=}$ if ϕ_l then t_l and $x_2^l \overset{\text{def}}{=}$ if $\neg\phi_l$ then t_l (and similarly on the right side).
Then, we apply G.EQUIV:FA$_\lambda$-APP several times to break-up the application under the λ; G.EQUIV:CONST-λ and G.EQUIV:FA-ADV to remove the if _ then _ else _ and _ = _ functions; G.EQUIV:DUP to remove the duplicated x_2; and G.EQUIV:CONST-λ to remove the λ in front of dflt. This yields

$$
\mathcal{E}; \Theta \vdash
\begin{aligned}
& \vec{u}_l, \mathsf{dflt}_l, \, \lambda(x : \tau). \, x_2^l, \lambda(x : \tau). \, x_1^l \\
& \sim \vec{u}_r, \mathsf{dflt}_r, \lambda(x : \tau). \, x_2^r, \lambda(x : \tau). \, x_1^r
\end{aligned}
$$

which is exactly our premisse (up-to-permutation).

- G.EQUIV:CONST-MANY-λ

We must prove that

$$\mathcal{E}; \Theta \vdash \vec{u}_l, \lambda(x : \tau). \, t_l \sim \vec{u}_r, \lambda(x : \tau). \, t_r$$

is a valid judgement. Since

$$\mathcal{E}; \Theta \vdash [\forall(x : \tau). \, \bigvee_{1 \le i \le n} \phi^i]$$

and

$$\mathcal{E}; \Theta \vdash [\bigwedge_{1 \le i \le n} \forall(x : \tau). \, \phi^i \Rightarrow (t_l = s_l^i \wedge t_r = s_r^i)]$$

are valid, we get that

$$
\mathcal{E}; \Theta \vdash
\left[
\forall(x : \tau). \, t_l =
\begin{array}{l}
\text{if } \phi_1 \text{ then } s_l^1 \text{ else} \\
\dots \\
\text{if } \phi_{n-1} \text{ then } s_l^{n-1} \text{ else } s_l^n
\end{array}
\right]
$$

is valid (we have a similar result for t_r).
We use G.REWRITE to rewrite this equality under the λ on both side. Then, as in the soundness proof of G.EQUIV:SPLIT-λ, we apply G.EQUIV:FA$_\lambda$-APP several times to break-up the applications under the λ; G.EQUIV:CONST-λ and G.EQUIV:FA-ADV to remove the if _ then _ else _ functions; and G.EQUIV:FA-ADV to remove the terms $(\lambda(x : \tau). \, \phi_i)_{1 \le i \le n}$, which yields

$$\mathcal{E}; \Theta \vdash \vec{u}_l, s_l^1, \dots, s_l^n \sim \vec{u}_r, s_r^1, \dots, s_r^n \qquad \square$$

G.EQUIV:SPLIT-λ

$$\mathcal{E};\Theta \vdash \begin{array}{l} \vec{u}_l, \mathsf{dflt}_l, \lambda(x:\tau).\,\mathsf{if} \quad \phi_l\ x\ \mathsf{then}\ t_l\ \mathsf{else}\ \mathsf{dflt}_l, \\ \lambda(x:\tau).\,\mathsf{if}\ \neg\phi_l\ x\ \mathsf{then}\ t_l\ \mathsf{else}\ \mathsf{dflt}_l \end{array} \sim \begin{array}{l} \vec{u}_r, \mathsf{dflt}_r, \lambda(x:\tau).\,\mathsf{if} \quad \phi_r\ x\ \mathsf{then}\ t_r\ \mathsf{else}\ \mathsf{dflt}_r, \\ \lambda(x:\tau).\,\mathsf{if}\ \neg\phi_r\ x\ \mathsf{then}\ t_r\ \mathsf{else}\ \mathsf{dflt}_r \end{array}$$
$$\overline{\mathcal{E};\Theta \vdash \vec{u}_l, \lambda(x:\tau).\,t_l \sim \vec{u}_r, \lambda(x:\tau).\,t_r}$$

G.EQUIV:CONST-MANY-λ

$$\frac{\mathcal{E};\Theta \vdash \tilde{\bigwedge}_{1\le i\le n}\mathsf{adv}(\lambda(x:\tau).\,\phi^i) \qquad \mathcal{E};\Theta \vdash [\forall(x:\tau).\,\bigvee_{1\le i\le n}\phi^i]}{\mathcal{E};\Theta \vdash [\bigwedge_{1\le i\le n}\forall(x:\tau).\,\phi^i \Rightarrow (t_l = s_l^i \wedge t_r = s_r^i)]}$$
$$\frac{\mathcal{E};\Theta \vdash \vec{u}_l, s_l^1,\ldots,s_l^n \sim \vec{u}_r, s_r^1,\ldots,s_r^n}{\mathcal{E};\Theta \vdash \vec{u}_l, \lambda(x:\tau).\,t_l \sim \vec{u}_r, \lambda(x:\tau).\,t_r}$$

Figure 9. Admissible additional indistinguishability rules.

APPENDIX F
FRESH RULES

This section is organised as follows: first, we prove the soundness of the G.EQUIV.FRESH rule in Appendix F-A; then, we give additional details on how we make the G.EQUIV.FRESH rule effective in Appendix F-B; we describe another equivalence freshness rule in Appendix F-C, which — together with the G.EQUIV.FRESH rule — allows to get rid of fresh names in an equivalence; and finally, we present the reachability freshness rule in Appendix F-D.

A. Soundness of The Fresh Equivalence Rule

We recall and prove Proposition 7.

Proposition 7. *The rule G.EQUIV.FRESH is sound.*

Proof. Consider an instance of the G.EQUIV.FRESH rule, and let $\mathbb{M}$ be a model of $\mathcal{E}$. We must show that $\mathbb{M}$ satisfies the judgement

$$\mathcal{E};\Theta \vdash \vec{u},\mathsf{n}\ t \sim \vec{u},\mathsf{n}_{\mathsf{fresh}}\ ()$$

For every $\eta \in \mathbb{N}$, let $\beta_{\mathbb{M}}^\eta : \mathbb{T}_{\mathbb{M},\eta} \to \mathbb{T}_{\mathbb{M},\eta}$ be the bijection swapping the random bits used by

$$\llbracket \mathsf{n} \rrbracket_{\mathbb{M}}(\eta, \llbracket t \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho})(\cdot) \text{ and } \llbracket \mathsf{n}_{\mathsf{fresh}}\ () \rrbracket_{\mathbb{M}}(\eta, \llbracket () \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho})(\cdot)$$

Said otherwise, for every tape $\rho = (\rho_a, \rho_h)$, if we let ρ_h' be such that $\beta_{\mathbb{M}}^\eta(\rho) = (\rho_a, \rho_h')$ then, for every name symbol n_0 of type $\tau_0 \to \tau$ and $a \in \llbracket \tau_0 \rrbracket_{\mathbb{M}}^\eta$:

$$\llbracket \mathsf{n}_0 \rrbracket_{\mathbb{M}}(\eta, a)(\rho_h') = $$
$$\begin{cases} \llbracket \mathsf{n}_{\mathsf{fresh}} \rrbracket_{\mathbb{M}}(\eta, \llbracket () \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho})(\rho_h) & \text{if } \mathsf{n}_0 = \mathsf{n} \text{ and } a = \llbracket t \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} \\ \llbracket \mathsf{n} \rrbracket_{\mathbb{M}}(\eta, \llbracket t \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho})(\rho_h) & \text{if } \mathsf{n}_0 = \mathsf{n}_{\mathsf{fresh}} \text{ and } a = \llbracket () \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} \\ \llbracket \mathsf{n}_0 \rrbracket_{\mathbb{M}}(\eta, a)(\rho_h) & \text{otherwise} \end{cases}$$

Let ρ be such that

$$\llbracket \phi_{\mathsf{fresh}}^{\mathsf{n},t}(\vec{u}, t) \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = 1 \tag{16}$$

Let us show, using Proposition 6, that

$$\llbracket \vec{u} \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = \llbracket \vec{u} \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\beta_{\mathbb{M}}^\eta(\rho)} \quad \text{and} \quad \llbracket t \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = \llbracket t \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\beta_{\mathbb{M}}^\eta(\rho)} \tag{17}$$

Let $(\vec{\alpha}, \psi, (x\ \vec{v})) \in \mathcal{ST}_{\mathcal{E}}(\vec{u}, t)$ and $\mathbb{M}'$ such that
- x is a variable declaration bound in $\mathcal{E}$;
- $\mathbb{M}'$ extends $\mathbb{M}$ into a model of $(\mathcal{E}, \vec{\alpha})$;
- $\llbracket \psi \rrbracket_{\mathbb{M}':\mathcal{E},\vec{\alpha}}^{\eta,\rho} = 1$.

To apply Proposition 6, we must prove that

$$\mathbb{M}(x)(\eta)(\rho)(a) = \mathbb{M}(x)(\eta)(\beta_{\mathbb{M}}^\eta(\rho))(a) \tag{18}$$

where $a \stackrel{\mathsf{def}}{=} \llbracket \vec{v} \rrbracket_{\mathbb{M}':\mathcal{E},\vec{\alpha}}^{\eta,\rho}$. There are 3 possible cases:
- If x is an adversarial function declaration, then since $\mathbb{M}(x)(\eta)$ can only use the component ρ_a of ρ, and since ρ and $\beta_{\mathbb{M}}^\eta(\rho)$ coincide on ρ_a, Eq. (18) trivially holds.
- If x is a name symbol in $\mathcal{N}$, we have three cases:
  - If $x \ne \mathsf{n}$ and $x \ne \mathsf{n}_{\mathsf{fresh}}$ then (18) holds by construction of $\beta_{\mathbb{M}}^\eta(\rho)$.
  - If $x = \mathsf{n}$, then we know that $\vec{v}$ starts by a term v and that $(\vec{\alpha}, \psi, \mathsf{n}\ v) \in \mathcal{ST}_{\mathcal{E}}(\vec{u}, t)$.
    Moreover, thanks to Eq. (16) and Eq. (2), we have
    $$\llbracket \forall \vec{\alpha}.\,\psi \Rightarrow t \ne v \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = 1$$
    Since $\mathbb{M}'$ extends $\mathbb{M}$ into a model of $(\mathcal{E}, \vec{\alpha})$, and since $\llbracket \psi \rrbracket_{\mathbb{M}':\mathcal{E},\vec{\alpha}}^{\eta,\rho} = 1$, we deduce that
    $$\llbracket t \ne v \rrbracket_{\mathbb{M}':\mathcal{E},\vec{\alpha}}^{\eta,\rho} = 1$$
    or equivalently $\llbracket t \rrbracket_{\mathbb{M}':\mathcal{E},\vec{\alpha}}^{\eta,\rho} \ne a$ (as $a \stackrel{\mathsf{def}}{=} \llbracket v \rrbracket_{\mathbb{M}':\mathcal{E},\vec{\alpha}}^{\eta,\rho}$). Hence (18) holds by construction of $\beta_{\mathbb{M}}^\eta(\rho)$.
- Finally, the case $x = \mathsf{n}_{\mathsf{fresh}}$ cannot happen, since we require that $\mathsf{n}_{\mathsf{fresh}}$ does not appear in $\vec{u}$ and t.

Also, by construction of the bijection $\beta_{\mathbb{M}}^\eta$, we have

$$\llbracket \mathsf{n}\ t \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\beta_{\mathbb{M}}^\eta(\rho)} = \llbracket \mathsf{n}_{\mathsf{fresh}}\ () \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} \tag{19}$$

Putting everything together, we get that:

$$\begin{aligned} \llbracket \vec{u},\mathsf{n}\ t \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\beta_{\mathbb{M}}^\eta(\rho)} &= \llbracket \vec{u} \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\beta_{\mathbb{M}}^\eta(\rho)}, \llbracket \mathsf{n}\ t \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\beta_{\mathbb{M}}^\eta(\rho)} \\ &= \llbracket \vec{u} \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\beta_{\mathbb{M}}^\eta(\rho)}, \llbracket \mathsf{n}_{\mathsf{fresh}}\ () \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} \quad \text{(by Eq. (19))} \\ &= \llbracket \vec{u} \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho}, \llbracket \mathsf{n}_{\mathsf{fresh}}\ () \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} \quad \text{(by Eq. (17))} \\ &= \llbracket \vec{u},\mathsf{n}_{\mathsf{fresh}}\ () \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} \end{aligned}$$

Since this equality holds for every ρ such that $\llbracket \phi_{\mathsf{fresh}}^{\mathsf{n},t}(\vec{u}, t) \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho}$, and since, by hypothesis, $\mathbb{M}$ satisfies $\mathcal{E};\Theta \vdash [\phi_{\mathsf{fresh}}^{\mathsf{n},t}(\vec{u}, t)]$, we know that $\Pr_\rho(\llbracket \phi_{\mathsf{fresh}}^{\mathsf{n},t}(\vec{u}, t) \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho}) \in \mathsf{ow}(\eta)$, which implies that

$$\left(\rho \mapsto \llbracket \vec{u},\mathsf{n}\ t \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\beta_{\mathbb{M}}^\eta(\rho)}\right) =_{\mathsf{ow}} \left(\rho \mapsto \llbracket \vec{u},\mathsf{n}_{\mathsf{fresh}}\ () \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho}\right) \tag{20}$$

23

Since $\beta_{\mathsf{M}}^{\eta}$ is a bijection and tapes are sampled uniformly at random, we get that for every $\mathcal{A} \in \mathsf{PPTMs}$:

$$\Pr_{\rho}\left(\mathcal{A}(1^{\eta}, [\![\vec{u}, \mathsf{n}\ t]\!]_{\mathsf{M}:\mathcal{E}}^{\eta,\rho}, \rho_{\mathsf{a}})\right) = \Pr_{\rho}\left(\mathcal{A}(1^{\eta}, [\![\vec{u}, \mathsf{n}\ t]\!]_{\mathsf{M}:\mathcal{E}}^{\eta,\beta_{\mathsf{M}}^{\eta}(\rho)}, \rho_{\mathsf{a}})\right)$$

Moreover, by Eq. (20) we have

$$\Pr_{\rho}\left(\mathcal{A}(1^{\eta}, [\![\vec{u}, \mathsf{n}\ t]\!]_{\mathsf{M}:\mathcal{E}}^{\eta,\beta_{\mathsf{M}}^{\eta}(\rho)}, \rho_{\mathsf{a}})\right)$$
$$=_{\mathsf{ow}} \Pr_{\rho}\left(\mathcal{A}(1^{\eta}, [\![\vec{u}, \mathsf{n}_{\mathsf{fresh}}\ ()]\!]_{\mathsf{M}:\mathcal{E}}^{\eta,\rho}, \rho_{\mathsf{a}})\right)$$

Hence $\mathsf{Adv}_{\mathsf{M}:\mathcal{E}}^{\eta}(\mathcal{A} : \vec{u}, \mathsf{n}\ t \sim \vec{u}, \mathsf{n}_{\mathsf{fresh}}\ ()) \in \mathsf{negl}(\eta)$. $\qquad\square$

B. Making the Fresh Equivalence Rule Effective

We first define a subsumption relation between occurrences. Intuitively, o_0 is subsumed by o_1, written $\mathsf{o}_0 \sqsubseteq \mathsf{o}_1$, if the set of terms represented by o_0 is smaller (w.r.t. set inclusion) than the set of terms represented by o_1.

Definition 2. *Let $\mathcal{E}$ be a well-formed environment and Θ a set of global formulas. We say that an occurrence $(\vec{\alpha}_0, \phi_0, t_0)$ is subsumed by another occurrence $(\vec{\alpha}_1, \phi_1, t_1)$ w.r.t. $\mathcal{E}$ and Θ, which we write $(\vec{\alpha}_0, \phi_0, t_0) \sqsubseteq_{\mathcal{E},\Theta} (\vec{\alpha}_1, \phi_1, t_1)$, if*

$$[\![\forall \vec{\alpha}_0.\ \phi_0 \Rightarrow (\exists \vec{\alpha}_1.\ \phi_1 \wedge t_0 = t_1)]\!]_{\mathsf{M}:\mathcal{E}}^{\eta,\rho} = 1$$

for every $\mathsf{M} : \mathcal{E}$ of Θ, and for every $\eta \in \mathbb{N}$ and $\rho \in \mathbb{T}_{\mathsf{M},\eta}$.

We lift this subsumption relation to (possibly infinite) sets of occurrences in the natural way: $\mathbb{S}_0 \sqsubseteq \mathbb{S}_1$ if each occurrence in $\mathbb{S}_0$ is subsumed by some occurrence in $\mathbb{S}_1$.

We now show that when an occurrence o_1 subsumes another occurrence o_0, then the freshness formula associated to o_0 is entailed by the freshness formula associated to o_1. Also, this entailment is naturally lifted to sets of occurrences.

Proposition 13. *Let $\mathcal{E}$ be a well-formed environment, Θ a set of global formulas, n a name symbol and t a term. For any set of occurrences $\mathbb{S}_0$ and $\mathbb{S}_1$ if $\mathbb{S}_0 \sqsubseteq_{\mathcal{E},\Theta} \mathbb{S}_1$ then for every model $\mathsf{M} : \mathcal{E}$ of Θ, $\eta \in \mathbb{N}$ and $\rho \in \mathbb{T}_{\mathsf{M},\eta}$:*

$$[\![\phi]\!]_{\mathsf{M}:\mathcal{E}}^{\eta,\rho} = 1 \text{ for every } \phi \in \mathcal{S}_{\mathsf{fresh}}^{\mathsf{n},t}(\mathbb{S}_1) \text{ implies}$$
$$[\![\phi]\!]_{\mathsf{M}:\mathcal{E}}^{\eta,\rho} = 1 \text{ for every } \phi \in \mathcal{S}_{\mathsf{fresh}}^{\mathsf{n},t}(\mathbb{S}_0)$$

The proof is given in Section F-B1.

Hence, to make the fresh rule effective, it is sufficient to find a *finite* set of occurrences $\mathbb{S}$ such that $\mathcal{ST}_{\mathcal{E}}(\vec{u}, t) \sqsubseteq_{\mathcal{E},\Theta} \mathbb{S}$, and to let $\phi_{\mathsf{fresh}}^{\mathsf{n},t}(\vec{u}, t)$ be the conjunction of all formulas in $\mathcal{S}_{\mathsf{fresh}}^{\mathsf{n},t}(\mathbb{S})$. Essentially, $\mathbb{S}$ is a sound *over-approximation* of $\mathcal{ST}_{\mathcal{E}}(\vec{u}, t)$: it "covers" all the occurrences in $\mathcal{ST}_{\mathcal{E}}(\vec{u}, t)$, hence in particular all occurrences of the name n in $\vec{u}$ and t.

There are many ways to build such a set $\mathbb{S}$. Of course, different sets $\mathbb{S}$ yield different formulas, with different degrees of precision. To make things more concrete, we present one effective way of building such a set $\mathbb{S}$ — this set will not be the most precise w.r.t. occurrences subsumption (such a most precise set may even not exist). Given a vector of terms $\vec{u}$ in some environment $\mathcal{E}$, we must build a finite set of occurrences $\mathcal{ST}_{\mathcal{E}}^{\#}(\vec{u})$ s.t. $\mathcal{ST}_{\mathcal{E}}(\vec{u}) \sqsubseteq_{\mathcal{E},\Theta} \mathcal{ST}_{\mathcal{E}}^{\#}(\vec{u})$.

To make this formal, we first introduce a more standard (and effective) notion of sub-terms, which correspond to the idea of *direct occurrence*. We let $\mathcal{ST}_{\mathcal{E}}^{0}(t)$ be the set of generalised subterms of t *without recursion*. The set $\mathcal{ST}_{\mathcal{E}}^{0}(t)$ is defined using the same equations than $\mathcal{ST}_{\mathcal{E}}(t)$ of Fig. 1, except for the case $t\ t'$ when t is a variable x defined in $\mathcal{E}$, which is replaced by $\mathcal{ST}_{\mathcal{E}}^{0}(t')$.

Coming back to the intuition presented in Example 10, we observe that any occurrence in $\mathcal{ST}_{\mathcal{E}}(\vec{u})$ is either:

i) A *direct* occurrence in , appearing in $\vec{u}$ without unrolling any definition. The set of such direct occurrences is computed by $\mathcal{ST}_{\mathcal{E}}^{0}(\vec{u})$.

ii) An *indirect* occurrence , appearing directly in the body of a definition $x_0 : \tau = \lambda y.\ t_0$ (*i.e.* appearing in $\mathcal{ST}_{\mathcal{E}}^{0}(t_0)$). This occurrence may appear at an arbitrary depth in $\mathcal{ST}_{\mathcal{E}}(\vec{u})$, but must have started from a defined variable x_1 applied to some argument s such that $x_1\ s$ is directly reachable in $\vec{u}$ (*i.e.* in $x_1\ s \in \mathcal{ST}_{\mathcal{E}}^{0}(\vec{u})$). Moreover, since recursive definitions are well-founded, we know that $y \leq_{x_0,x_1} s$ (where y is the argument of x_0, and $\leq_{x_0,x_1}$ is the formula representing the well-founded order of the recursive definitions in $\mathcal{E}$, see Appendix B).
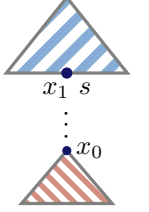
Formally, for any environment $\mathcal{E}$, we let $\mathcal{ST}_{\mathcal{E}}^{\#}(\vec{u})$ be set of occurrences

$$\mathcal{ST}_{\mathcal{E}}^{0}(\vec{u}) \cup \bigcup_{\substack{(x_0:\tau_0 \to \tau_1 = \lambda y.\ t_0) \in \mathcal{E} \\ (x_1 : \_ = \_) \in \mathcal{E} \\ (x_1\ s) \in \mathcal{ST}_{\mathcal{E}}^{0}(\vec{u})}} (\mathsf{fv}(s)).(y : \tau_0).[y \leq_{x_0,x_1} s]\mathcal{ST}_{\mathcal{E}}^{0}(t_0)$$

Proposition 14. *For any environment $\mathcal{E}$, $\mathcal{ST}_{\mathcal{E}}^{\#}(\vec{u})$ is a* finite *and* effectively computable *set of occurrences such that* $\mathcal{ST}_{\mathcal{E}}(\vec{u}) \sqsubseteq_{\mathcal{E},\Theta} \mathcal{ST}_{\mathcal{E}}^{\#}(\vec{u})$.

Proof (sketch). We observe that any occurrence $(\vec{\alpha}, \phi, t)$ in $\mathcal{ST}_{\mathcal{E}}(\vec{u})$ appears either directly in $\vec{u}$, or indirectly in the body of some inductively defined variable $x_0 : \tau_0 \to \tau_1 = \lambda y.\ t_0$. In the latter case, we know that ϕ entails $y \leq_{x_0,x_1} u$ (by well-foundedness of the model). Hence

$$(\vec{\alpha}, \phi, t) \sqsubseteq_{\mathcal{E},\Theta} (\vec{\alpha}, (y \leq_{x_0,x_1} u) \wedge \phi, t)$$

Then, if we let ψ be the condition such that $((\vec{\alpha}_0, y), \psi, t) \in \mathcal{ST}_{\mathcal{E}}^{0}(t_0)$ (where $(\vec{\alpha}_0, y) \subseteq \vec{\alpha}$), we have that $\phi \Rightarrow \psi$ holds for all tapes (indeed, it is clear that $\phi = \phi_0 \wedge \psi$ for some formula ϕ_0). Using the fact that weakening the condition of an occurrence yields a more general occurrence (w.r.t. $\sqsubseteq_{\mathcal{E},\Theta}$), we get $(\vec{\alpha}, (y \leq_{x_0,x_1} u) \wedge \phi, t) \sqsubseteq_{\mathcal{E},\Theta} (\vec{\alpha}, (y \leq_{x_0,x_1} u) \wedge \psi, t)$.

Hence, by transitivity of $\sqsubseteq_{\mathcal{E},\Theta}$, we have $(\vec{\alpha}, \phi, t) \sqsubseteq_{\mathcal{E},\Theta} (\vec{\alpha}, (y \leq_{x_0,x_1} u) \wedge \psi, t)$. We conclude by removing from $\vec{\alpha}$ all variables that are not bound in $(y \leq_{x_0,x_1} u) \wedge \psi$ and t, which gives an occurrence in

$$(\mathsf{fv}(u)).(y : \tau_0).[y \leq_{x_0,x_1} u]\mathcal{ST}_{\mathcal{E}}^{0}(t_0) \qquad\square$$

1) Proof of Proposition 13: We prove the following proposition, which is a generalised version of Proposition 13.

Proposition 15. *Let $\mathcal{E}$ be a well-formed environment, Θ a set of global formulas, n a name symbol and t a term.*

- *for any occurrences $(\vec{\alpha}_0, \phi_0, t_0)$ and $(\vec{\alpha}_1, \phi_1, t_1)$, if*

$$(\vec{\alpha}_0, \phi_0, t_0) \sqsubseteq_{\mathcal{E},\Theta} (\vec{\alpha}_1, \phi_1, t_1)$$

then for every model $\mathbb{M} : \mathcal{E}$ of Θ, $\eta \in \mathbb{N}$ and $\rho \in \mathbb{T}_{\mathbb{M},\eta}$:

$$\llbracket \forall \vec{\alpha}_1.\, \phi_1 \Rightarrow t \neq t_1 \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = 1 \text{ implies}$$
$$\llbracket \forall \vec{\alpha}_0.\, \phi_0 \Rightarrow t \neq t_0 \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = 1 \quad (21)$$

- *for any set of occurrences $\mathbb{S}_0$ and $\mathbb{S}_1$ if*

$$\mathbb{S}_0 \sqsubseteq_{\mathcal{E},\Theta} \mathbb{S}_1$$

then for every model $\mathbb{M} : \mathcal{E}$ of Θ, $\eta \in \mathbb{N}$ and $\rho \in \mathbb{T}_{\mathbb{M},\eta}$:

$$\llbracket \phi \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = 1 \text{ for every } \phi \in \mathcal{S}_{fresh}^{\mathsf{n},t}(\mathbb{S}_1) \text{ implies}$$
$$\llbracket \phi \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = 1 \text{ for every } \phi \in \mathcal{S}_{fresh}^{\mathsf{n},t}(\mathbb{S}_0) \quad (22)$$

Proof. The proof of the second point trivially follows from the first. We detail the first point, which is quite simple. Assume that

$$(\vec{\alpha}_0, \phi_0, t_0) \sqsubseteq_{\mathcal{E},\Theta} (\vec{\alpha}_1, \phi_1, t_1) \quad (23)$$

Take a model $\mathbb{M} : \mathcal{E}$ of Θ, and let $\eta \in \mathbb{N}$ and $\rho \in \mathbb{T}_{\mathbb{M},\eta}$. Assume that:

$$\llbracket \forall \vec{\alpha}_1.\, \phi_1 \Rightarrow t \neq t_1 \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = 1 \quad (24)$$

By definition of $\sqsubseteq_{\mathcal{E},\Theta}$, we know from Eq. (23) that

$$\llbracket \forall \vec{\alpha}_0.\, \phi_0 \Rightarrow (\exists \vec{\alpha}_1.\, \phi_1 \wedge t_0 = t_1) \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = 1 \quad (25)$$

We must show that

$$\llbracket \forall \vec{\alpha}_0.\, \phi_0 \Rightarrow t \neq t_0 \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = 1$$

Take $\vec{a}_0 \in \llbracket \vec{\tau}_0 \rrbracket_{\mathbb{M}}^{\eta}$ (where $\vec{\tau}_0$ are the types of $\vec{\alpha}_0$), and assumes that $\llbracket \phi_0 \rrbracket_{\mathbb{M}[\vec{\alpha}_0 \mapsto \vec{a}_0]:(\mathcal{E},\vec{\alpha}_0)}^{\eta,\rho} = 1$. It is sufficient to show that

$$\llbracket t \neq t_0 \rrbracket_{\mathbb{M}[\vec{\alpha}_0 \mapsto \vec{a}_0]:(\mathcal{E},\vec{\alpha}_0)}^{\eta,\rho} = 1 \quad (26)$$

Since $\llbracket \phi_0 \rrbracket_{\mathbb{M}[\vec{\alpha}_0 \mapsto \vec{a}_0]:(\mathcal{E},\vec{\alpha}_0)}^{\eta,\rho} = 1$, we know using Eq. (25) that there exists $\vec{a}_1 \in \llbracket \vec{\tau}_1 \rrbracket_{\mathbb{M}}^{\eta}$ (where $\vec{\tau}_1$ are the types of $\vec{\alpha}_1$) such that

$$\llbracket \phi_1 \wedge t_0 = t_1 \rrbracket_{\mathbb{M}[\vec{\alpha}_0 \mapsto \vec{a}_0, \vec{\alpha}_1 \mapsto \vec{a}_1]:(\mathcal{E},\vec{\alpha}_0,\vec{\alpha}_1)}^{\eta,\rho} = 1$$

By Eq. (24), we know that

$$\llbracket \phi_1 \Rightarrow t \neq t_1 \rrbracket_{\mathbb{M}[\vec{\alpha}_1 \mapsto \vec{a}_1]:(\mathcal{E},\vec{\alpha}_1)}^{\eta,\rho} = 1 \quad (27)$$

Since $\vec{\alpha}_0 \cap \vec{\alpha}_1 = \emptyset$, this holds even if we extend the model with $\vec{\alpha}_0$, hence

$$\llbracket \phi_1 \Rightarrow t \neq t_1 \rrbracket_{\mathbb{M}[\vec{\alpha}_0 \mapsto \vec{a}_0, \vec{\alpha}_1 \mapsto \vec{a}_1]:(\mathcal{E},\vec{\alpha}_0,\vec{\alpha}_1)}^{\eta,\rho} = 1 \quad (28)$$

Eq. (27) and Eq. (28) together gives us Eq. (26). □

C. Removing Fresh Names

Let n_{fresh} be a name symbol of type $\mathsf{unit} \to \tau$. Assume that n_{fresh} is fresh, *i.e.* that it does not appear in $\mathcal{E}, \vec{u}_1, \vec{u}_2$, except in its declaration. Then we have the rule

G.EQUIV.SIMPL-NAMES
$$\frac{\mathcal{E};\Theta \vdash \vec{u}_1 \sim \vec{u}_2}{\mathcal{E};\Theta \vdash \vec{u}_1, \mathsf{n}_{fresh}\ () \sim \vec{u}_2, \mathsf{n}_{fresh}\ ()}$$

The soundness of this rule is easily proved, by showing that the name n_{fresh} can be directly sampled by the adversary using its own random bits (which changes nothing since n_{fresh} does not appear in the rest of the computations).

Using this rule in combination with two applications of the G.EQUIV.FRESH rule allows to get rid of names which can be proved fresh, through the following (admissible) rule

$$\frac{\mathcal{E};\Theta \vdash [\phi_{fresh}^{\mathsf{n}_1,t_1}(\vec{u}_1, t_1)] \tilde{\wedge} [\phi_{fresh}^{\mathsf{n}_2,t_2}(\vec{u}_2, t_2)]}{\mathcal{E};\Theta \vdash \vec{u}_1, \mathsf{n}_1\ t_1 \sim \vec{u}_2, \mathsf{n}_2\ t_2}$$

where $[\phi_{fresh}^{\mathsf{n}_1,t_1}(\vec{u}_1, t_1)]$ and $[\phi_{fresh}^{\mathsf{n}_2,t_2}(\vec{u}_2, t_2)]$ captures, resp., the freshness of $\mathsf{n}_1\ t_1$ and $\mathsf{n}_2\ t_2$. This rule is essentially the rule implemented in SQUIRREL.

D. Freshness in Local Formulas

Let n be a name symbol with type $\tau_0 \to \tau$, and t_0, t be terms of type, resp., τ_0 and τ, in some well-typed environment $\mathcal{E}$. Assume that τ is a large type, *i.e.* that $\mathsf{large} \in \mathsf{label}(\tau)$. Let Θ be a set of global hypotheses. We ask that all declared variables x of $\mathcal{E}$ occur only in eta-long form, and are either names or only depend on the adversarial tape (*i.e.* $\mathsf{adv}(x) \in \Theta$).

The local freshness rule states that if an equality $\llbracket \mathsf{n}\ t_0 = t \rrbracket_{\mathbb{M}:\mathcal{E}}$ holds overwhelmingly often, then we know that $\mathsf{n}\ t_0$ must appear somewhere in t's computation. Indeed, if $\mathsf{n}\ t_0$ does not appear in t's computation then $\llbracket t \rrbracket_{\mathbb{M}:\mathcal{E}}$ and $\llbracket \mathsf{n}\ t_0 \rrbracket_{\mathbb{M}:\mathcal{E}}$ are independent random variables. Since $\llbracket \mathsf{n}\ t_0 \rrbracket_{\mathbb{M}:\mathcal{E}}$ takes values in $\llbracket \tau \rrbracket_{\mathbb{M}}$, which is a large type, it follows that they have a negligible probability of collision.

This idea is captured by the following rule

L.FRESH
$$\frac{}{\mathcal{E};\Theta;\Gamma \vdash \mathsf{n}\ t_0 = t \Rightarrow \neg\phi_{fresh}^{\mathsf{n},t_0}(t, t_0)}$$

where $\phi_{fresh}^{\mathsf{n},t_0}(t, t_0)$ is a formula expressing the freshness of $\mathsf{n}\ t_0$ in t, t_0, defined as in Section IV-A.

Proposition 16. *The rule L.FRESH is sound.*

Proof. Let $\mathbb{M}$ be a model of $\mathcal{E}$, let us prove that

$$\mathcal{E};\Theta;\Gamma \vdash \left(\mathsf{n}\ t_0 = t \wedge \phi_{fresh}^{\mathsf{n},t_0}(t, t_0) \right) \Rightarrow \mathsf{false}$$

Assume that $\mathbb{M} \models \Theta$, we must prove that

$$\Pr_{\rho} \left(\llbracket (\wedge\Gamma) \wedge \left(\mathsf{n}\ t_0 = t \wedge \phi_{fresh}^{\mathsf{n},t_0}(t, t_0) \right) \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} \right) \in \mathsf{negl}(\eta)$$

Then

$$\Pr_{\rho} \left(\llbracket (\wedge\Gamma) \wedge \left(\mathsf{n}\ t_0 = t \wedge \phi_{fresh}^{\mathsf{n},t_0}(t, t_0) \right) \rrbracket_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} \right)$$

$$\leq \Pr_{\rho}\left([\![\mathsf{n}\ t_0 = t \wedge \phi_{\mathsf{fresh}}^{\mathsf{n},t_0}(t,t_0)]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho}\right)$$

$$= \Pr_{\rho}\left([\![\mathsf{n}\ t_0 = t]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} \cap [\![\phi_{\mathsf{fresh}}^{\mathsf{n},t_0}(t,t_0)]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho}\right)$$

$$= \sum_{\substack{a\in[\![\tau]\!]_{\mathbb{M}}^{\eta} \\ a'\in[\![\tau_0]\!]_{\mathbb{M}}^{\eta}}} \Pr_{\rho}\left([\![t]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = a \cap [\![\mathsf{n}]\!]_{\mathbb{M}}(\eta,a')(\rho_{\mathsf{h}}) = a \cap \right. \qquad (29)$$
$$\left. [\![t_0]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = a' \cap [\![\phi_{\mathsf{fresh}}^{\mathsf{n},t_0}(t,t_0)]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho}\right)$$

where ρ_{h} is the honest tape in ρ.

Intermediary result: We start by assuming the following results (we show it at the end of this proof).

For any $a \in [\![\tau]\!]_{\mathbb{M}}^{\eta}$, $a' \in [\![\tau_0]\!]_{\mathbb{M}}^{\eta}$ and $\rho = (\rho_{\mathsf{a}}, \rho_{\mathsf{h}}) \in \mathbb{T}_{\mathbb{M},\eta}$ such that $[\![\phi_{\mathsf{fresh}}^{\mathsf{n},t_0}(t,t_0)]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = 1$, we have

$$[\![t]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = [\![t]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho'} \qquad [\![t_0]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = [\![t_0]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho'}$$
$$[\![\mathsf{n}]\!]_{\mathbb{M}}(\eta,a')(\rho_{\mathsf{h}}) = [\![\mathsf{n}]\!]_{\mathbb{M}}(\eta,a')(\rho_{\mathsf{h}}'') \qquad (30)$$

where, morally, we split the randomness ρ into disjoint part ρ' and ρ'', where the latter contain only the random bits used by n a'. More precisely:

- *ρ' is the (pair of) tapes in $\mathbb{T}_{\mathbb{M},\eta}$ obtained by setting to 0 in ρ the random bits read by n a',*
- *ρ'' is the (pair of) tapes in $\mathbb{T}_{\mathbb{M},\eta}$ obtained by setting to 0 in ρ all random bits but the bits read by n a' (and where ρ_{h}'' is the honest component of ρ'').*

Notice that sampling ρ is equivalent from sampling independently ρ' and ρ'', according to the distributions described above (as all bits in ρ are sampled uniformly and independently at random).

Upper-bounding Eq. (29): Continuing from Eq. (29):

$$\Pr_{\rho}\left([\![(\wedge\Gamma) \wedge \left(\mathsf{n}\ t_0 = t \wedge \phi_{\mathsf{fresh}}^{\mathsf{n},t_0}(t,t_0)\right)]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho}\right)$$

$$\leq \sum_{\substack{a\in[\![\tau]\!]_{\mathbb{M}}^{\eta} \\ a'\in[\![\tau_0]\!]_{\mathbb{M}}^{\eta}}} \Pr_{\rho',\rho''}\left([\![t]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho'} = a \cap [\![\mathsf{n}]\!]_{\mathbb{M}}(\eta,a')(\rho_{\mathsf{h}}'') = a \cap \right.$$
$$\left. [\![t_0]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho'} = a' \cap [\![\phi_{\mathsf{fresh}}^{\mathsf{n},t_0}(t,t_0)]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho}\right)$$
$$\text{(Using Eq. (30))}$$

$$\leq \sum_{\substack{a\in[\![\tau]\!]_{\mathbb{M}}^{\eta} \\ a'\in[\![\tau_0]\!]_{\mathbb{M}}^{\eta}}} \Pr_{\rho',\rho''}\left([\![t]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho'} = a \cap [\![\mathsf{n}]\!]_{\mathbb{M}}(\eta,a')(\rho_{\mathsf{h}}'') = a \cap \right.$$
$$\left. [\![t_0]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho'} = a'\right)$$

$$\leq \sum_{\substack{a\in[\![\tau]\!]_{\mathbb{M}}^{\eta} \\ a'\in[\![\tau_0]\!]_{\mathbb{M}}^{\eta}}} \Pr_{\rho'}\left([\![t]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho'} = a \cap [\![t_0]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho'} = a'\right)$$
$$\cdot \Pr_{\rho''}\left([\![\mathsf{n}]\!]_{\mathbb{M}}(\eta,a')(\rho_{\mathsf{h}}'') = a\right)$$

By independence: indeed, since ρ' and ρ'' are independent, $[\![t]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho'}$ and $[\![t_0]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho'}$ are independent from $[\![\mathsf{n}]\!]_{\mathbb{M}}(\eta,a')(\rho_{\mathsf{h}}'')$. Continuing:

$$\leq \sum_{\substack{a\in[\![\tau]\!]_{\mathbb{M}}^{\eta} \\ a'\in[\![\tau_0]\!]_{\mathbb{M}}^{\eta}}} \Pr_{\rho'}\left([\![t]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho'} = a \cap [\![t_0]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho'} = a'\right) \cdot \frac{1}{2^{c_\tau \cdot \eta}}$$

$$\text{(where } c_\tau > 0, \text{ since } \tau \text{ is a large type)}$$

$$= \frac{1}{2^{c_\tau \cdot \eta}} \cdot \sum_{\substack{a\in[\![\tau]\!]_{\mathbb{M}}^{\eta} \\ a'\in[\![\tau_0]\!]_{\mathbb{M}}^{\eta}}} \Pr_{\rho'}\left([\![t]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho'} = a \cap [\![t_0]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho'} = a'\right)$$

$$= \frac{1}{2^{c_\tau \cdot \eta}}. \qquad \text{(as the sum above evaluates to 1)}$$

As this quantity is negligible, this concludes the proof.

Intermediary result (proof): Let us show the result admitted above. From the three equalities in Eq. (30), the third one is straightforward:

$$[\![\mathsf{n}]\!]_{\mathbb{M}}(\eta,a')(\rho_{\mathsf{h}}) = [\![\mathsf{n}]\!]_{\mathbb{M}}(\eta,a')(\rho_{\mathsf{h}}'')$$

We focus on the first two equalities. As their proofs are identical, we only show that

$$[\![t]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = [\![t]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho'}$$

To do this, we apply Proposition 6. Let $(\vec{\alpha}, \psi, (x\ \vec{v})) \in \mathcal{ST}_{\mathcal{E}}(t,t_0)$ and $\mathbb{M}'$ such that

- x is a variable declaration bound in $\mathcal{E}$;
- $\mathbb{M}'$ extends $\mathbb{M}$ into a model of $(\mathcal{E}, \vec{\alpha})$;
- $[\![\psi]\!]_{\mathbb{M}':\mathcal{E},\vec{\alpha}}^{\eta,\rho} = 1$.

To apply Proposition 6, we must prove that

$$\mathbb{M}(x)(\eta)(\rho)(b) = \mathbb{M}(x)(\eta)(\rho')(b) \qquad (31)$$

where $b \overset{\mathsf{def}}{=} [\![\vec{v}]\!]_{\mathbb{M}':\mathcal{E},\vec{\alpha}}^{\eta,\rho}$. The proof goes as in the proof of Proposition 7. There are 2 possible cases:

- If x is an adversarial function declaration, then since $\mathbb{M}(x)(\eta)$ can only use the component ρ_{a} of ρ, and since ρ and ρ' coincide on ρ_{a}, Eq. (31) trivially holds.
- If x is a name symbol in $\mathcal{N}$, we have three cases:
  - If $x \neq \mathsf{n}$ then (31) holds by construction of ρ'.
  - If $x = \mathsf{n}$, then we know that $\vec{v}$ starts by a term v and that $(\vec{\alpha}, \psi, \mathsf{n}\ v) \in \mathcal{ST}_{\mathcal{E}}(t,t_0)$.
    Since $[\![\phi_{\mathsf{fresh}}^{\mathsf{n},t_0}(t,t_0)]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = 1$, we know that

    $$[\![\phi]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = 1 \text{ for every } \phi \in \mathcal{S}_{\mathsf{fresh}}^{\mathsf{n},t_0}(\mathcal{ST}_{\mathcal{E}}(t,t_0))$$

    (this is the condition that $\phi_{\mathsf{fresh}}^{\mathsf{n},t_0}(t,t_0)$ must satisfy to have a valid instance of the L.FRESH rule).
    Hence, by definition of $\mathcal{S}_{\mathsf{fresh}}^{\mathsf{n},t_0}(\cdot)$ and since $(\vec{\alpha}, \psi, \mathsf{n}\ v) \in \mathcal{ST}_{\mathcal{E}}(t,t_0)$:

    $$[\![\forall\vec{\alpha}.\ \psi \Rightarrow t_0 \neq v]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = 1$$

    Since $\mathbb{M}'$ extends $\mathbb{M}$ into a model of $(\mathcal{E}, \vec{\alpha})$, and since $[\![\psi]\!]_{\mathbb{M}':\mathcal{E},\vec{\alpha}}^{\eta,\rho} = 1$, we deduce that

    $$[\![t_0 \neq v]\!]_{\mathbb{M}':\mathcal{E},\vec{\alpha}}^{\eta,\rho} = 1$$

    or equivalently $[\![t_0]\!]_{\mathbb{M}':\mathcal{E},\vec{\alpha}}^{\eta,\rho} \neq b$ (as $b \overset{\mathsf{def}}{=} [\![v]\!]_{\mathbb{M}':\mathcal{E},\vec{\alpha}}^{\eta,\rho}$).
    Hence (31) holds by construction of ρ'. $\qquad\square$

APPENDIX G
CRYPTOGRAPHIC RULES

A. Checking the PPTM Side-Condition

We describe how we establish that a term t can be computed in polynomial-time.

Let $\mathcal{E}$ be a well-typed environment and Θ a set of global hypotheses well-typed in $\mathcal{E}$. Let $\vec{\alpha}$ be a set of (typed) variables, and t a term well-typed in $\mathcal{E}, \vec{\alpha}$. The judgement

$$\mathcal{E}; \Theta; \vec{\alpha} \vdash_{\mathsf{pptm}} t \quad \text{where } \vec{\alpha} = \alpha_1 : \tau_1, \ldots, \alpha_n : \tau_n$$

states that there exists a PPTM $\mathcal{A}$ such that $\mathcal{A}$ computes $[\![t]\!]$ by following the *same computation steps* as the recursive

$$\frac{\text{PT.INPUT}}{\mathcal{E};\Theta;\vec{\alpha},x \vdash_{\mathsf{pptm}} x}$$

$$\frac{\text{PT.APP} \qquad \mathcal{E};\Theta;\vec{\alpha} \vdash_{\mathsf{pptm}} t \qquad \mathcal{E};\Theta;\vec{\alpha} \vdash_{\mathsf{pptm}} t'}{\mathcal{E};\Theta;\vec{\alpha} \vdash_{\mathsf{pptm}} t\, t'}$$

$$\frac{\text{PT.NAME} \qquad \mathsf{n} \in \mathcal{N} \qquad \mathcal{E};\Theta;\vec{\alpha} \vdash_{\mathsf{pptm}} t}{\mathcal{E};\Theta;\vec{\alpha} \vdash_{\mathsf{pptm}} \mathsf{n}\, t}$$

$$\frac{\text{PT.ADV} \qquad \mathcal{E};\Theta \vdash \mathsf{adv}(x) \qquad (x:\tau) \in \mathcal{E}}{\mathcal{E};\Theta;\vec{\alpha} \vdash_{\mathsf{pptm}} x}$$

$$\frac{\text{PT.QUANT} \qquad \mathtt{polynomial} \in \mathsf{label}(\tau) \\ \mathcal{Q} \in \{\exists;\forall\} \qquad \mathcal{E};\Theta;\vec{\alpha},x \vdash_{\mathsf{pptm}} t}{\mathcal{E};\Theta;\vec{\alpha} \vdash_{\mathsf{pptm}} \mathcal{Q}(x:\tau).\,t}$$

$$\frac{\text{PT.LAMBDA} \qquad \mathcal{E};\Theta;\vec{\alpha},x \vdash_{\mathsf{pptm}} t}{\mathcal{E};\Theta;\vec{\alpha} \vdash_{\mathsf{pptm}} \lambda(x:\tau).\,t}$$

$$\frac{\text{PT.DEF:DELTA} \qquad (x:\tau_0 \to \tau = t) \in \mathcal{E} \\ \mathcal{E};\Theta;\vec{\alpha} \vdash_{\mathsf{pptm}} t}{\mathcal{E};\Theta;\vec{\alpha} \vdash_{\mathsf{pptm}} x}$$

$$\frac{\text{PT.WEAK} \qquad \mathcal{E}_0 \text{ well-typed} \qquad \mathcal{E}_0 \vdash \Theta_0 \\ \mathcal{E}_0 \vdash t:\tau \qquad \mathcal{E}_0;\Theta_0;\vec{\alpha} \vdash_{\mathsf{pptm}} t}{\mathcal{E}_0,\mathcal{E}_1;\Theta_0,\Theta_1;\vec{\alpha} \vdash_{\mathsf{pptm}} t}$$

Figure 10. Rules for the PPTM side-condition.

definition of the semantics $\llbracket \cdot \rrbracket$. Formally, this judgement is valid iff for any model $\mathbb{M}:\mathcal{E}$ such that $\mathbb{M} \models \Theta$, there must exists a PPTM $\mathcal{A}$ such that for any $\eta \in \mathbb{N}$, $\rho \in \mathbb{T}_{\mathbb{M},\eta}$, for any $(e_i)_{i\in[1;n]} \in (\llbracket \tau_i \rrbracket^\eta_{\mathbb{M}})_{i\in[1;n]}$

$$\mathcal{A}(1^\eta, (e_i)_{i\in[1;n]}, \rho) = \llbracket t \rrbracket^{\eta,\rho}_{\mathbb{M}[(\alpha_i \mapsto e_i)_{i\in[1;n]}]:(\mathcal{E},\vec{\alpha})} \qquad (32)$$

where $\mathcal{A}$ running-time is polynomial w.r.t. 1^η. Moreover, the above must hold for any subterm in $\mathcal{ST}_{\mathcal{E}}(t)$, extending the environment if necessary to deal with bound variables (this essentially forces $\mathcal{A}$ to follows the same steps as $\llbracket \cdot \rrbracket$).

We now explain how Eq. (32) must be understood when t has a higher-order type. Assume w.l.o.g. that t has type $\tau_0^t \to \ldots \to \tau_l^t \to \tau$ where τ is a base-type. Then $\mathcal{A}$ must be compute in polynomial-time a closure $\mathcal{A}(1^\eta, \vec{e}, \rho)$ such that, on any input $(a_i)_{i\in[1;l]} \in (\llbracket \tau_i^t \rrbracket^\eta_{\mathbb{M}})_{i\in[1;l]}$

$$\mathcal{A}(1^\eta, (e_i)_{i\in[1;n]}, \rho)(a_1, \ldots, a_l) = \\ \llbracket t \rrbracket^{\eta,\rho}_{\mathbb{M}[(\alpha_i \mapsto e_i)_{i\in[1;n]}]:(\mathcal{E},\vec{\alpha})}(a_1, \ldots, a_l)$$

in polynomial-time in 1^η.

Rules: We present in Fig. 10 a set of rules which allows to establish that a term can be computed in polynomial-time. The rule PT.INPUT is trivial, as x is an input of the machine. PT.APP simply composes the two PPTMs given by the premises. PT.NAME exploits that fact that we required that names are interpreted as polynomial-time computable random samplings. PT.ADV is immediate, since $\mathsf{adv}(x)$ requires that x is computable by a PPTM which only accesses the adversarial component of the random tape (note that the PPTIME judgement has no restrictions over the random tapes). Note that we cannot generalise this rule to an arbitrary term t, as we have no guarantee that the machine provided by $\mathsf{adv}(t)$ follows the semantics $\llbracket \cdot \rrbracket$. The PT.QUANT relies on the fact that if there exists a PPTM $\mathcal{B}$ computing t for any input $x:\tau$, and if τ is polynomial, then we can compute $\forall(x:\tau).\,t$ by evaluating $\mathcal{B}$ over all values of $\llbracket \tau \rrbracket$ (*e.g.* with a for loop), which is possible since we required that $\mathtt{polynomial}$ base-types

are enumerable in polynomial-time. The PT.LAMBDA simply builds a closure to the machine given by the premise. The rule PT.DEF:DELTA is immediate: the machine for the premise already computes t. Finally, PT.WEAK is a simple weakening rule allowing to remove unused definitions.

Proposition 17. *The rules in Fig. 10 are sound.*

We already quickly justified the soundness of each rules. We omit the details.

Handling recursive definitions: For now, we did not present any rule allowing to deal with recursive definitions (PT.DEF:DELTA does not allow to deal with recursive calls, considering our set of rules). Designing general-purpose criteria to establish that recursive procedures are polynomial-time goes beyond the scope of this work.

Still, we choose to present a basic criterion which covers the simple case where we have (mutually) recursive functions over *finite and fixed* types. This result can handle the recursive definition already in SQUIRREL (*e.g.* see Example 6), which is all we need for now.

Proposition 18. *Let $\mathcal{E}$ be an environment such that*

$$\mathcal{E} = \mathcal{E}_0, (x_i : \tau_i \to \tau_i' = \lambda(y_i : \tau_i).\,t_i)_{i\in[1;n]}$$

where $\mathcal{E}_0$ contains only variables declarations. Assume that for every $i \in [1;n]$, the type τ_i of the (possibly recursive) defined variable x_i is finite and fixed, i.e. $\{\mathtt{finite}, \mathtt{fixed}\} \subseteq \mathsf{label}(\tau_i)$.

Let $(x_i')_{i\in[1;n]}$ be fresh variable names. If, given input y_i, the bodies of the defined variable $(x_i)_{i\in[1;n]}$ are all computable by a PPTM, assuming (by recurrence) that other defined variables are computable, i.e.

$$\mathcal{E};\Theta;\vec{\alpha}, (x_i')_{i\in[1;n]}, y_i \vdash_{\mathsf{pptm}} (t_i\{x_i \mapsto x_i' \mid i \in [1;n]\})$$

then the following rule is a sound rule

$$\frac{\text{PT.DEF:REC-FINITE} \qquad \mathcal{E};\Theta;\vec{\alpha}, (x_i)_{i\in[1;n]} \vdash_{\mathsf{pptm}} s}{\mathcal{E};\Theta;\vec{\alpha} \vdash_{\mathsf{pptm}} s}$$

Note that we cannot easily weaken this result to only require that recursion is over values of a type τ for τ polynomial. Indeed, while this would bound the depth of the call-tree by a polynomial, this is insufficient. Take:

$$f = \lambda x.\mathsf{if}\ x = 0\ \mathsf{then}\ \mathsf{one}\ \mathsf{else}\ (f(x-1)||(x-1))$$

where $||$ denotes bit-string concatenation and one is the bit-string of length 1 with a single bit set to 1. The function f doubles in size at each recursive calls, and we clearly have $|f(\eta)| \approx 2^\eta$, which is exponential. Bounding the depth of recursive calls is insufficient, it is necessary to also bound the size of the output of each call.

Proof of Proposition 18 (sketch). Take a model $\mathbb{M}$, then recursion depth is finite and independent of η. Since all the bodies are computable in polynomial-time assuming that recursive calls have been computed, we compose a finite number of PPTM, which tields a PPTM. $\qquad\square$

27

There exists a rich literature on techniques to establish that recursive procedures are polynomial-time (for example using the syntactic restriction of [38], Hoare logics for costs [39], or implicit computational complexity [40]), from which we could profit to handle more complex procedures. As such extensions would be completely independent from the rest of this paper, we leave them as future work.

B. EUF Rule

We prove Proposition 8 in this section. For this purpose, we fix an instance of the rule, and a model $\mathbb{M} : \mathcal{E}$ of the premises.

We construct a machine $\mathcal{M}$ which, given η, ρ as input, computes $[\![s]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}}$ and $[\![m]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}}$ using two oracles:

- The first oracle takes no input and returns $[\![\mathsf{pk}\ (\mathsf{k}\ t)]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}}$.
- The second oracle takes one input m and returns $[\![\mathsf{sign}\ x\ (\mathsf{k}\ t)]\!]^{\eta,\rho}_{\mathbb{M}[x \mapsto m]:(\mathcal{E}, x:\mathsf{message})}$.

Our machine relies on a recursive procedure $I_\sigma(u)$ which, given a term u with free variables in $\mathcal{E} \cup \vec{\alpha}$ such that $\mathcal{E}; \Theta; \vec{\alpha} \vdash_{\mathsf{pptm}} u$, given a semantic assignment σ of domain $\vec{\alpha}$, computes $[\![u]\!]^{\eta,\rho}_{\mathbb{M}[\vec{\alpha} \mapsto \sigma(\vec{\alpha})]:(\mathcal{E}, \vec{\alpha})}$ or fails, as follows:

- If $u \in \vec{\alpha}$, return $\sigma(u)$.
- If u is a declared variable of $\mathcal{E}$, run the machine provided by $\mathsf{adv}(u)$.
- There is no case for defined variables, because terms are in eta-long form. If $u \equiv x\ u'$ for a defined variable with $(x : \tau = \lambda y.v) \in \mathcal{E}$, return $I_\sigma(v\{y \mapsto u'\})$.
- If u is a conditional (if ϕ then t_1 else t_2), compute $I_\sigma(\phi)$. If it is 1, return $I_\sigma(t_1)$, otherwise return $I_\sigma(t_2)$.
- **Key case**, when u is of the form $\mathsf{k}\ t'$. Compute $I_\sigma(t')$ and $I(t)$. If they are equal, fail. Otherwise, fall back to the next case, *i.e.* access the honest tape bits corresponding to name $\mathsf{k}\ t'$.
- If u is a name $\mathsf{n}\ u'$, compute u' and use the polynomial time computation associated to n to extract the relevant bits from the honest tape.
- **Oracle case 1**, when u is of the form $(\mathsf{pk}\ (\mathsf{k}\ t'))$. Compute $I_\sigma(t')$ and $I(t)$. If they are equal, call the public key oracle. Otherwise fall back to the application case below, *i.e.* use the machines associated to pk and k to explicitly compute the desired interpretation from $I_\sigma(t')$.
- **Oracle case 2**, when u is of the form $(\mathsf{sign}\ m\ (\mathsf{k}\ t'))$. Compute $I_\sigma(t')$ and $I(t)$. If they are equal, call the signature oracle on $I_\sigma(m)$. Otherwise fall back to the application case below, *i.e.* use the machines associated to sign and k to explicitly compute the desired interpretation from $I_\sigma(t')$.
- If u is a quantifier $\mathcal{Q}x.\phi$, it must be on a polynomial type by the pptm assumption. Enumerate the elements of $[\![\tau]\!]^{\eta}_{\mathbb{M}}$, compute $I_{\sigma[x \mapsto e]}(\phi)$ for each element e, and aggregate these results as needed to compute the semantics of the quantifier $\mathcal{Q}$.
- If u is an application, compute the two subterms and apply one to the other.
- If u is an abstraction $\lambda x.u'$, return the closure expecting one extra input e and returning $I_{\sigma[x \mapsto e]}(u')$.

This recursive computation strictly follows the inference rules of the pptm side condition; it is clear that recursive calls are performed on subterms that also satisfy this side condition, and that our procedure does compute the required interpretations. In fact, without the oracle and key cases, the procedure would exactly be the one witnessing the validity of the pptm assumption on t. In particular, it runs in polynomial time.

Because t is constant, the term $(\mathsf{k}\ t)$ can be viewed as a random key generation process that is independent of all other computations. Further, our machine $\mathcal{M}$ never accesses the bits of the honest random tape corresponding to $[\![\mathsf{k}\ t]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}}$. Indeed, it only runs adversarial computations and, for the tape accesses that it performs directly (that is, the case of names) the special key case avoids $[\![\mathsf{k}\ t]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}}$. The machine $\mathcal{M}$ can thus be seen as an adversary against the EUF-CMA game where the randomly chosen key is represented by $[\![\mathsf{k}\ t]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}}$.

We further observe that, for the values of η, ρ such that $[\![\phi^{\mathsf{k}}_{\mathsf{key}}(s,m,t)]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}} = [\![\phi^{\mathsf{k}}_{\mathsf{sign}}(s,m,t)]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}} = 1$, the following conditions hold for all the terms u such that $I_\sigma(u)$ is called in $\mathcal{M}$:

1) $[\![\forall \vec{\alpha}.\ \psi \Rightarrow t \neq t_0]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}} = 1$ for every $(\vec{\alpha}, \psi, \mathsf{k}\ t_0) \in \mathcal{ST}^{\mathsf{euf}}_{\mathcal{E},\mathsf{k},t}(u)$.
2) $[\![\forall \vec{\alpha}.\ \psi \Rightarrow (t = t_0) \Rightarrow (m \neq m_0)]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}} = 1$ for every $(\vec{\alpha}, \psi, \mathsf{sign}\ m_0\ (\mathsf{k}\ t_0)) \in \mathcal{ST}^{\mathsf{euf}}_{\mathcal{E},\mathsf{k},t}(u)$.

To see that this is true, first note that the conditions hold by assumption on s and m, on which the initial calls to I are performed. Then, assuming that the conditions hold for some call to $I_\sigma(u)$, we show that it holds for recursive calls:

- In the key and oracle cases, a recursive call is performed on t, but we have our two conditions by assumption on t just like s and m.
- In the case of a defined variable, we perform a recursive call on the expansion of the definition, but this follows the definition of our generalised subterms.
- In the conditional case, we call $I(t_i)$ only on the relevant subterm, which is crucial for the conditions to carry over to it.
- In the oracle cases, we call $I(\mathsf{k}\ t')$ only when we $I_\sigma(t') \neq I(t)$, which is in line with the special treatment of this subterm in $\mathcal{ST}^{\mathsf{euf}}_{\mathcal{E},\mathsf{k},t}(u)$.
- In all other cases, the recursive call is on a direct subterm for which the condition easily carries over.

The previous observation implies that, in the key case for the computation of I, the failure case is impossible when our conditions hold, and that the signature oracle is never called on $[\![m]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}}$. If, moreover, we had $[\![\mathsf{verify}\ s\ m\ (\mathsf{pk}\ (\mathsf{k}\ t))]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}} = 1$, our machine would win the EUF-CMA game. Since this can only happen with negligible probability, we can conclude:

$$\mathbb{M} \models [\neg(\phi^{\mathsf{k}}_{\mathsf{key}}(s,m,t) \wedge \phi^{\mathsf{k}}_{\mathsf{sign}}(s,m,t) \wedge \mathsf{verify}\ s\ m\ (\mathsf{pk}\ (\mathsf{k}\ t)))]$$

C. PRF Rule

We now briefly describe how we derive a rule for the PRF cryptographic assumption [41] which supports corruptions. We use a variant of the PRF assumption which states that a PPTM attacker, given access to a hashing oracle, has a negligible

$$\frac{\mathcal{E}; \Theta; \emptyset \vdash_{\mathsf{pptm}} \vec{u}, m, t \qquad \mathcal{E}; \Theta \vdash \mathsf{const}(t) \qquad \mathcal{E}; \Theta; \emptyset \vdash \psi_{\mathsf{key}}^{\mathsf{k}}(\vec{u}, m, t)}{\mathcal{E}; \Theta \vdash \vec{u}, (\mathsf{hash}\ m\ (\mathsf{k}\ t)) \sim \vec{u}, \mathsf{if}\ \psi_{\mathsf{msg}}^{\mathsf{k}}(\vec{u}, m, t)\ \mathsf{then}\ \mathsf{n}_{\mathsf{fresh}}\ ()\ \mathsf{else}\ \mathsf{hash}\ m\ (\mathsf{k}\ t)}$$

Figure 11. Variant PRF rule.

chance of distinguishing a randomly sampled value (in the appropriate space) from the hash of a message that has not been used in a query to the hashing oracle.

To model this assumption, we assume the following declaration, for some types key and hash:

$$\mathsf{hash} : \mathsf{message} \rightarrow \mathsf{key} \rightarrow \mathsf{hash}$$

We further assume a name $\mathsf{k} : \tau \rightarrow \mathsf{key}$ that will be used to model the key generation mechanism of our hash function.

In order to reflect the computations with a hash oracle, we adapt the generalised subterms, similarly to what we did for EUF-CMA, this time with the following key defining equation:

$$\mathcal{ST}_{\mathcal{E}, \mathsf{k}, t}^{\mathsf{prf}}(\mathsf{hash}\ m\ (\mathsf{k}\ t')) \overset{\mathsf{def}}{=}$$
$$\{(\epsilon, \mathsf{true}, m), (\epsilon, \mathsf{true}, t')\} \cup [t \neq t'] \mathcal{ST}_{\mathcal{E}, \mathsf{k}, t}^{\mathsf{prf}}(\mathsf{k}\ t')$$

The following rule captures the PRF assumption, *i.e.* it is sound w.r.t. the class of models where hash is interpreted as a hash function that satisfies the PRF assumption:

G.PRF

$$\frac{\mathcal{E}; \Theta; \emptyset \vdash_{\mathsf{pptm}} \vec{u}, m, t}{\mathcal{E}; \Theta \vdash \mathsf{const}(t) \qquad \mathcal{E}; \Theta; \emptyset \vdash \psi_{\mathsf{key}}^{\mathsf{k}}(\vec{u}, m, t) \wedge \psi_{\mathsf{msg}}^{\mathsf{k}}(\vec{u}, m, t)}{\mathcal{E}; \Theta \vdash \vec{u}, (\mathsf{hash}\ m\ (\mathsf{k}\ t)) \sim \vec{u}, (\mathsf{n}_{\mathsf{fresh}}\ ())}$$

where $\mathsf{n}_{\mathsf{fresh}} : \mathsf{unit} \rightarrow \mathsf{hash}$ is a name that does not occur anywhere else in the rule and the local formulas $\psi_{\mathsf{key}}^{\mathsf{k}}(\vec{u}, m, t)$ and $\psi_{\mathsf{msg}}^{\mathsf{k}}(\vec{u}, m, t)$ satisfy the following conditions for any $\mathbb{M} \models \Theta$:

- If $[\![\psi_{\mathsf{key}}^{\mathsf{k}}(\vec{u}, m, t)]\!]_{\mathbb{M}:\mathcal{E}}^{\eta, \rho} = 1$ then $[\![\forall \vec{\alpha}.\ \psi \Rightarrow t \neq t_0]\!]_{\mathbb{M}:\mathcal{E}}^{\eta, \rho} = 1$ for every $(\vec{\alpha}, \psi, \mathsf{k}\ t_0) \in \mathcal{ST}_{\mathcal{E}, \mathsf{k}, t}^{\mathsf{prf}}(\vec{u}, m, t)$.
- If $[\![\psi_{\mathsf{msg}}^{\mathsf{k}}(\vec{u}, m, t)]\!]_{\mathbb{M}:\mathcal{E}}^{\eta, \rho} = 1$ then

$$[\![\forall \vec{\alpha}.\ \psi \Rightarrow (t = t_0) \Rightarrow (m \neq m_0)]\!]_{\mathbb{M}:\mathcal{E}}^{\eta, \rho} = 1$$

  for every $(\vec{\alpha}, \psi, \mathsf{hash}\ m_0\ (\mathsf{k}\ t_0)) \in \mathcal{ST}_{\mathcal{E}, \mathsf{k}, t}^{\mathsf{prf}}(\vec{u}, m, t)$.

We sketch the soundness argument of our rule. Assume, by contradiction, that the rule does not hold, *i.e.* consider a model $\mathbb{M}$ for which the premises are satisfied, such that $\mathbb{M} \models \Theta$ but $\mathbb{M}$ does not satisfy the equivalence in conclusion. In other words, there exists a PPTM $\mathcal{D}$ that can distinguish $[\![\vec{u}, \mathsf{hash}\ m\ (\mathsf{k}\ t)]\!]_{\mathbb{M}:\mathcal{E}}^{\eta, \rho}$ from $[\![\vec{u}, \mathsf{n}_{\mathsf{fresh}}\ ()]\!]_{\mathbb{M}:\mathcal{E}}^{\eta, \rho}$ with non-negligible advantage.

We construct a PRF adversary as follows:

1) First compute $[\![\vec{u}]\!]_{\mathbb{M}:\mathcal{E}}^{\eta, \rho}$ and $[\![m]\!]_{\mathbb{M}:\mathcal{E}}^{\eta, \rho}$, similarly to how it is done in the previous section for EUF-CMA. In particular, never access k at index t, and use the hashing oracle to compute the interpretation of subterms $(\mathsf{hash}\ m_0\ (\mathsf{k}\ t_0))$ when t_0 interprets into $[\![t]\!]_{\mathbb{M}:\mathcal{E}}^{\eta, \rho}$.
2) Query the PRF challenge oracle with $[\![m]\!]_{\mathbb{M}:\mathcal{E}}^{\eta, \rho}$. Call c the resulting value: in the left game, c is $[\![\mathsf{hash}\ m\ (\mathsf{k}\ t)]\!]_{\mathbb{M}:\mathcal{E}}^{\eta, \rho}$; in the right game, it is $\mathsf{n}_{\mathsf{fresh}}\ ()$.

3) Call $\mathcal{D}$ with $([\![\vec{u}]\!]_{\mathbb{M}:\mathcal{E}}^{\eta, \rho}, c)$.

As before, our game is an equivalent reformulation of the cryptographers' game in the CCSA framework: we use k at a specific index to model the key sampled by the game (and make sure that this key is not accessed by the adversary); we use $\mathsf{n}_{\mathsf{fresh}}$ to model the value sampled in the challenge (and guarantee, again, that this source of randomness is not used elsewhere).

In general, the first step of our PRF adversary may fail, in case the interpretation of the terms forces an access to $[\![\mathsf{k}\ t]\!]_{\mathbb{M}:\mathcal{E}}^{\eta, \rho}$ outside of oracle computations. However, this only happens with negligible probability because $\psi_{\mathsf{key}}^{\mathsf{k}}(\vec{u}, m, t)$ is overwhelming.

Our PRF adversary may query the PRF challenge with an invalid message m, if it has been previously used in a hash oracle query. Again, this can only happen with negligible probability, because $\psi_{\mathsf{msg}}^{\mathsf{k}}(\vec{u}, m, t)$ is overwhelming,

Hence the advantage of our PRF adversary is negligibly close to the advantage of $\mathcal{D}$, which is non-negligible: we have contradicted the PRF cryptographic assumption on the interpretation of hash.

D. CCA Rule

In this section, we present the proof rule encoding the IND-CCA-1 assumption, used to reason about the secrecy of encrypted messages.

We present here the rule for the case of (randomised) asymmetric encryption, *i.e.* encryption schemes using a public encryption key, and a different, private decryption key. The case of symmetric encryption, where the encryption and decryption key are the same, is similar, as discussed at the end of the section.

Assume $\mathcal{E}$ contains symbols pk, aenc, adec, modelling resp. the public key associated to a private key, the encryption of a message with a random value and a public key, and the decryption of a ciphertext with a private key, with types

$$\mathsf{pk} : \mathsf{key} \rightarrow \mathsf{pubkey}$$
$$\mathsf{aenc} : \mathsf{message} \rightarrow \mathsf{rand} \rightarrow \mathsf{pubkey} \rightarrow \mathsf{ciphertext}$$
$$\mathsf{adec} : \mathsf{ciphertext} \rightarrow \mathsf{key} \rightarrow \mathsf{message}$$

Their expected behaviour when correctly used is described by the axiom:

$$[\forall m : \mathsf{message}, r : \mathsf{rand}, k : \mathsf{key}.$$
$$\mathsf{adec}\ (\mathsf{aenc}\ m\ r\ (\mathsf{pk}\ k))\ k = m]$$

We will also assume functions $\mathsf{len} : \mathsf{message} \rightarrow \mathsf{int}$ and $\mathsf{zero} : \mathsf{int} \rightarrow \mathsf{message}$. We consider models where $\mathsf{len}\ m$ returns the length of the bit-string interpreting m, and $\mathsf{zero}\ n$ returns a bit-string consisting of n zeros.

The (non-adaptive) *Indistinguishability under Chosen Ciphertext Attacks* (IND-CCA-1) assumption [28] is a standard cryptographic hypothesis, stating that an adversary cannot distinguish the encryption of a message m from the encryption of a string of len m zeros, *i.e.* zero (len m); even when given access to the public encryption key, and restricted access to a decryption oracle. More precisely, for randomly sampled k and r, an adversary given access to pk k and to a decryption oracle returning the decryption of any ciphertext with k is asked to produce a message m of his choice. He is then provided by a so-called "left-or-right" encryption oracle (which may only be called once) with either aenc m r (pk k) or aenc (zero (len m)) r (pk k). He can afterwards no longer call the decryption oracle, and is asked to guess which ciphertext has been returned. The assumption is that such an attacker cannot do so with a non-negligible advantage.

We capture this assumption, for two distinct indexed names r and k, by the rule

G.CCA1

$$\frac{\begin{array}{c} \mathcal{E}; \Theta; \emptyset \vdash_{\mathsf{pptm}} \vec{u}, m, t_r, t_k \\ \mathcal{E}; \Theta \vdash \mathsf{const}(t_r) \qquad \mathcal{E}; \Theta \vdash \mathsf{const}(t_k) \\ \mathcal{E}; \Theta \vdash [\phi_{\mathsf{key}}^{\mathsf{k}}(\vec{u}, m, t_r, t_k) \wedge \phi_{\mathsf{rand}}^{\mathsf{r}}(\vec{u}, m, t_r, t_k)] \end{array}}{\begin{array}{c} \mathcal{E}; \Theta \vdash \quad \vec{u}, \ \mathsf{aenc} \ m \ (\mathsf{r} \ t_r) \ (\mathsf{pk} \ (\mathsf{k} \ t_k)) \\ \sim \vec{u}, \ \mathsf{aenc} \ (\mathsf{zero} \ (\mathsf{len} \ m)) \ (\mathsf{r} \ t_r) \ (\mathsf{pk} \ (\mathsf{k} \ t_k)) \end{array}}$$

where $\phi_{\mathsf{key}}^{\mathsf{k}}(\vec{u}, m, t_r, t_k)$ and $\phi_{\mathsf{rand}}^{\mathsf{r}}(\vec{u}, m, t_r, t_k)$ are, similarly to the case of L.EUF, any formulas that imply, respectively

- the correct use of the key k t_k: the adversary only has access to the public key and the decryption oracle, meaning that k t_k may only appear in pk (k t_k) or adec _ (k t_k);
- the correct use of the randomness r t_r: this random value is sampled by the left-or-right encryption oracle, and thus may not appear anywhere else.

Basically, under these conditions, the challenge ciphertexts encrypting m or zeros cannot be distinguished (even put together with any vector of terms $\vec{u}$ that also satisfy the conditions).

As in the case of rule L.EUF, to formalise these conditions, we need to adapt the notion of generalised subterms to account for the oracles. We define $\mathcal{ST}_{\mathcal{E},\mathsf{k},t_k}^{\mathsf{cca}}(v)$ recursively just like $\mathcal{ST}_{\mathcal{E}}(v)$ (Fig. 1), with exceptions when v is a function application:

$$\mathcal{ST}_{\mathcal{E},\mathsf{k},t_k}^{\mathsf{cca}}(\mathsf{pk} \ (\mathsf{k} \ t_0)) \stackrel{\mathsf{def}}{=}$$
$$\{(\epsilon, \mathsf{true}, \mathsf{pk} \ (\mathsf{k} \ t_0)), (\epsilon, \mathsf{true}, \mathsf{pk})\}$$
$$\cup \ \mathcal{ST}_{\mathcal{E},\mathsf{k},t_k}^{\mathsf{cca}}(t_0) \cup [t_0 \neq t_k] \mathcal{ST}_{\mathcal{E},\mathsf{k},t_k}^{\mathsf{cca}}(\mathsf{k} \ t_0)$$
$$\mathcal{ST}_{\mathcal{E},\mathsf{k},t_k}^{\mathsf{cca}}(\mathsf{adec} \ t_1 \ (\mathsf{k} \ t_0)) \stackrel{\mathsf{def}}{=}$$
$$\{(\epsilon, \mathsf{true}, \mathsf{adec} \ t_1 \ (\mathsf{k} \ t_0)), (\epsilon, \mathsf{true}, \mathsf{adec})\}$$
$$\cup \ \mathcal{ST}_{\mathcal{E},\mathsf{k},t_k}^{\mathsf{cca}}(t_0) \cup \mathcal{ST}_{\mathcal{E},\mathsf{k},t_k}^{\mathsf{cca}}(t_1) \cup [t_0 \neq t_k] \mathcal{ST}_{\mathcal{E},\mathsf{k},t_k}^{\mathsf{cca}}(\mathsf{k} \ t_0)$$

We can then formally express the conditions on $\phi_{\mathsf{key}}^{\mathsf{k}}(\vec{u}, m, t_r, t_k)$ and $\phi_{\mathsf{rand}}^{\mathsf{r}}(\vec{u}, m, t_r, t_k)$. For every model $\mathbb{M} : \mathcal{E}$ of Θ, every $\eta \in \mathbb{N}$ and $\rho \in \mathbb{T}_{\mathbb{M},\eta}$, we require that:

1) if $[\![\phi_{\mathsf{key}}^{\mathsf{k}}(\vec{u}, m, t_r, t_k)]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = 1$ then
$$[\![\forall \vec{\alpha}. \ \psi \Rightarrow t_k \neq t_0]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = 1$$
for every $(\vec{\alpha}, \psi, \mathsf{k} \ t_0) \in \mathcal{ST}_{\mathcal{E},\mathsf{k},t_k}^{\mathsf{euf}}(\vec{u}, m, t_r, t_k)$;
2) if $[\![\phi_{\mathsf{rand}}^{\mathsf{k}}(\vec{u}, m, t_r, t_k)]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = 1$ then
$$[\![\forall \vec{\alpha}. \ \psi \Rightarrow t_r \neq t_0]\!]_{\mathbb{M}:\mathcal{E}}^{\eta,\rho} = 1$$
for every $(\vec{\alpha}, \psi, \mathsf{r} \ t_0) \in \mathcal{ST}_{\mathcal{E},\mathsf{k},t_k}^{\mathsf{euf}}(\vec{u}, m, t_r, t_k)$.

In practice, we effectively compute these two formulas in a way similar to what is explained for the G.EQUIV.FRESH rule in Appendix F-B.

Proposition 19. *The rule* G.CCA1 *is sound.*

That proposition is proved similarly to the soundness proofs for the previous two cryptographic rules.

Symmetric case: The rule above applies in the case of asymmetric encryption. A corresponding rule, though slightly more complex, can be written in the case of symmetric encryption.

In that case, the attacker does not get access to the encryption key, since it is the same as the decryption key. Instead, he has access to an additional oracle, that encrypts any message, and can be called any number of times, at any point. This slightly changes the definition of $\mathcal{ST}_{\mathcal{E},\mathsf{k},t_k}^{\mathsf{cca}}(\cdot)$, to account for this additional oracle, in the expected way: occurrences of the key as third argument of the encryption function are not recorded. We also must add another condition, similar to $\phi_{\mathsf{rand}}^{\mathsf{r}}(\cdot)$, to ensure that the same random is never used to encrypt two different plaintexts – indeed, encryptions can only be computed by the oracle, who samples a fresh random value each time.