Robust Logical Foundations for Mechanizing Post-Quantum Cryptography in Squirrel

April 16th, 2026 – v1

An extended abstract of this paper has been accepted at ACM CCS'26; this is the full version.

David Baelde
Univ Rennes, CNRS, IRISA
Rennes, France
david.baelde@irisa.fr

Antoine Dallon
AMIAD
Rennes, France
antoine.dallon@m4x.org

Stéphanie Delaune
Univ Rennes, CNRS, IRISA
Rennes, France
stephanie.delaune@irisa.fr

Charlie Jacomme
Université de Lorraine, CNRS, Inria, LORIA
Nancy, France
charlie.jacomme@inria.fr

Adrien Koutsos
Inria
Paris, France
adrien.koutsos@inria.fr

Abstract

The advent of quantum computers has initiated both research and standardization efforts toward the development of cryptographic primitives and communication protocols that remain secure against attackers equipped with quantum computers. Computer-aided verification has proven valuable in this context, as it helps identify flaws early and strengthens confidence in cryptographic systems. However, existing verification tools are typically designed for a specific attacker model (most often polynomial-time Turing machines in line with classical cryptographic assumptions) and therefore require adaptation to accurately capture the quantum setting.

In this work, we present a novel post-quantum extension of Squirrel, a proof assistant that provides computational guarantees for cryptographic primitives and protocols. Our extension is fully embedded in the higher-order logic underlying Squirrel, allowing logical terms to directly represent quantum values. This design choice makes the extension generic, preserves compatibility with the latest features of Squirrel, and supports its long-term integration into the tool. We implement our approach within Squirrel and validate it through several case studies. In particular, we obtain post-quantum security guarantees for multiple KEM combiners, as well as for two hybrid key-exchange protocols.

CCS Concepts

• **Security and privacy** → **Logic and verification**; **Cryptography**; • **Theory of computation** → *Cryptographic protocols*; *Denotational semantics*; **Quantum computation theory**.

Keywords

Computer-aided Cryptography, Security Protocols, Post-Quantum Cryptography, Logic, Verification

1 Introduction

Cryptographic protocols are fundamental to securing modern communication systems, making it imperative to rigorously validate their security. *Computer-aided cryptographic verification* addresses this need by providing formal, machine-checkable methods to establish strong security guarantees [13]. Over the past years, this has led to the development of several verification tools, including CryptoVerif [18], EasyCrypt [15], SSProve [1], and Squirrel [6]. Given a protocol and a security goal, such tools aim to prove that the success probability of any real-world adversary $\mathcal{A}$ is negligible, assuming the underlying primitives are themselves secure. In the classical setting, adversaries are modeled as probabilistic polynomial-time classical Turing machines (PPTM), a framework long regarded as a realistic representation of practical attacker capabilities. The advent of quantum computing, however, undermines this assumption, rendering the classical threat model inadequate for analyzing the security of cryptographic primitives and protocols.

Post-quantum cryptography (PQC) seeks to design cryptographic systems secure considering probabilistic polynomial-time *quantum* Turing machines (PQTM) to model the adversary. The field has progressed significantly as exemplified by the recent NIST standardization of schemes such as ML-KEM [31] and ML-DSA [30]. To ensure a secure transition and reduce the risks of adopting entirely new primitives, hybrid protocols and cryptographic combiners are increasingly employed. Notably, applications like Signal and Apple's iMessage have already deployed hybrid protocols combining classical and post-quantum cryptographic mechanisms: e.g. PQXDH [32], SPQR [22], and PQ3 [4, 29].

While a lot of progress has been made towards building and proving the security of post-quantum cryptographic primitives and protocols, computer-aided verification is still lagging behind. For example, a recent large-scale verification effort [2, 3] successfully established the IND-CCA security of ML-KEM in EasyCrypt, yet the proof remains limited to a classical adversary. This limitation arises because EasyPQC [14], the post-quantum extension of EasyCrypt, lacks support for key reasoning steps required to handle quantum attackers. Adapting an existing cryptographic verification framework to the post-quantum world usually poses two main challenges: *(i)* the framework must be adapted to model quantum computations; *(ii)* the way cryptographic arguments are captured by the framework must be adapted too. For instance, it is important that the simulator used to justify a cryptographic reduction

does not copy a quantum state, as this will violate the *no-cloning* theorem [26, 35]. PQC also requires to only consider security assumptions that do have post-quantum secure instantiations, and thus to forbid *e.g.* the Decisional Diffie-Hellman (DDH) assumption. While this is a major aspect of PQC, this issue is easy to solve, and has no bearing on the design of computer-aided PQC.

Squirrel [6] is a proof assistant dedicated to the verification of cryptographic protocols in the classical setting. In this framework, messages and adversarial computations are represented as terms (e.g. u), whose semantic interpretation, $[\![u]\!]$, corresponds to a distribution over bitstrings. The framework relies on a higher-order probabilistic logic [10], and manipulates two predicates:

- Computational indistinguishability, $u \sim v$, meaning that no PPTM can distinguish between $[\![u]\!]$ and $[\![v]\!]$ except with negligible probability;
- Overwhelming truth, $[\phi]$, asserting that ϕ holds with overwhelming probability.

Compared to game-based computational proofs, the distinguisher is split into two parts, with a final top-level distinguisher used in the interpretation of $\sim$, and an attacker interacting with the protocol abstracted as a function symbol att(), interpreted as a stateless PPTM. Both adversaries share access to an arbitrary long source of random coins and can recompute their shared state if needed.

Finally, cryptographic reasoning is encoded directly through inference rules. For instance, we have the following rules, presented here in simplified form.

$$\frac{[u = v] \qquad v \sim w}{u \sim w} \qquad\qquad \frac{[\mathsf{len}(m_0) = \mathsf{len}(m_1)]}{\mathsf{enc}(m_0, r_0, \mathsf{pk}(sk)) \sim \mathsf{enc}(m_1, r_1, \mathsf{pk}(sk))}$$

The first rule (Rewrite$_c$) allows rewriting u as v when they are overwhelmingly equal. The second (CPA$_c$) captures the IND-CPA assumption by asserting that encryptions of equal-length messages are indistinguishable under some assumptions (omitted here) on the use of the secret key sk. Thanks to its design, the logic is computationally sound and provides guarantees against PPTM attackers.

Addressing in Squirrel the two challenges of computer-aided PQC implies two core questions:

- With the arbitrary long source of randomness currently given to the attacker, can we model quantum computations and probabilistic measurements in a faithful way?
- How to reason over the quantum state of the attacker to ensure that its usage is correct?

Contributions. In this work, we address those challenges and present a rigorous extension of Squirrel to support post-quantum cryptography through the following contributions:

- We introduce a new execution model for Squirrel that explicitly represents adversarial state inside the terms and enables reasoning over it.
- We model quantum values and quantum computations inside Squirrel's logic, and demonstrate that this model is faithful up to a controlled and negligible error.
- We adapt Squirrel's reasoning rules and extend its recent features, including the **crypto** tactic [11] and the **smt** tactic [8], to our new semantics, thereby providing advanced reasoning capabilities and a flexible foundation for long-term PQC support.
- Finally, we validate our approach by analyzing four KEM combiners and two hybrid key exchange protocols based on KEMs [17, 21].

Related Works. Recently, several verification tools have introduced post-quantum extensions. As mentioned, EasyCrypt has been adapted to the post-quantum setting via EasyPQC [14]. Another extension of the logic behind EasyCrypt has been proposed and implemented in the `qrhl-tool`, based on Isabelle [33]. The latter approach consists of a full modeling of quantum computations and values inside a logic, which yields a complex but very expressive logic. It has been used for instance to prove the security of the Fujisaki-Okamoto transformation [34] in the QROM, and is capable of reasoning about quantum protocols.

More recently, CryptoVerif was extended to the post-quantum setting in [19]. Its semantics were adapted to ensure that the adversary is accessed only in a black-box fashion, which guarantees post-quantum soundness of reductions. However, the framework does not introduce quantum values, making it difficult to further extend CryptoVerif to settings like the QROM.

A first post-quantum version of Squirrel, named PQ-Squirrel, was previously proposed in [24]. While this approach enabled some early successes, it suffers from several significant drawbacks. In particular, it does not allow quantum values to be represented directly. As a result, the notion of a (quantum) state is kept implicit, following the approach used by Squirrel in the classical setting, where a given attacker state is recomputed when needed. However, in the quantum case, it is not always possible to do so, since duplicating, and thus recomputing, a quantum state is inherently impossible. To address this issue, the authors introduce syntactic conditions ensuring that terms are well-defined and accordingly restrict the application of all inference rules so that every intermediate term produced during a proof satisfies these conditions. In particular, even simple rules such as Rewrite$_c$ must be applied under additional constraints. This introduces practical difficulties, as such rules are often used implicitly in the implementation of Squirrel tactics. Since Squirrel does not construct explicit proof objects, it is very difficult to ensure that the syntactic conditions required by the theory are met. More generally, it is unclear how PQ-Squirrel can benefit from the latest improvements of Squirrel, such as SMT support [8] or automated generic cryptographic reductions [11]. Finally, the underlying semantics is not expressive enough to model the QROM, which ultimately limits the applicability of the approach. In conclusion, while [24] paved the way toward post-quantum reasoning in Squirrel, it falls short in terms of trust, maintainability, and expressiveness.

Outline. We introduce our execution model in Section 2. In Section 3, we recall the core logic of Squirrel and describe how it can directly model quantum values. Then, Section 4 presents how we faithfully model adversarial quantum computations within the logic. In Section 5, we describe our adaptation of Squirrel's proof system to this quantum setting. Finally, we discuss certain aspects of the implementation and illustrate our approach with several case studies in Section 6.

2 An Execution Model for PQC

In this section, we start with a high-level primer on quantum computing in Section 2.1, before introducing our execution model for post-quantum cryptography in Section 2.2. We illustrate our approach on a pair of protocols corresponding to the CCA experiment for a KEM combiner.

We first discuss two aspects of Squirrel's language which are particularly relevant to the design of our execution model — we postpone the formal presentation of the language to Section 3. First, Squirrel uses a *pure language* taking the form of a typed λ-calculus. While pure languages are well-suited for logical reasoning, cryptographic systems are usually modeled using an imperative style, which must thus go through an encoding. In particular, a call to a stateful adversary is usually modeled in Squirrel as a call to a pure function to which we give all past inputs of the attacker, so that it can internally recompute the current state. In this work, we will model this by a pure function that explicitly takes its state as input and returns the updated state. Roughly, for a stateful attacker $\mathcal{A}$ and a program p,

$$(\mathsf{out} \leftarrow \mathcal{A}(\mathsf{in}); p) \quad \text{becomes} \quad \textbf{let } (\mathsf{out}, \mathsf{st}') = \mathsf{att}(\mathsf{st}, \mathsf{in}) \textbf{ in } \tilde{p}$$

where $\tilde{p}$ is the functional encoding of p.

Second, probabilistic behaviors are captured using *eagerly sampled randomness*, which programs can access through arrays of pre-sampled random values called *names*. Concretely, a name $n : \tau_0 \to \tau_1$ is an array, indexed by elements of type τ_0, of independently and identically distributed (i.i.d.) values of type τ_1. Because Squirrel uses discrete probabilities, we require that τ_0 is finite and that we can sample from τ_1 using a finite number of bits.

2.1 Quantum Adversary

We use terminating Turing machines to describe the computational capabilities of adversaries. The execution of a *deterministic* machine $\mathcal{M}$ on a bitstring input $u \in \{0,1\}^*$ yields a bitstring output $\mathcal{M}(u) \in \{0,1\}^*$. If the machine $\mathcal{M}$ is *probabilistic*, $\mathcal{M}(u)$ is a discrete bitstring distribution, which we represent as a formal sum:

$$\mathcal{M}(u) = \textstyle\sum_{v \in \{0,1\}^*} p_v \cdot v \quad \text{where} \quad \sum_v p_v = 1 \quad \text{and} \quad \forall v.\, p_v \in \mathbb{R}^+.$$

In Squirrel, probabilistic computations are modeled by letting $\mathcal{M}$ have access to an array of pre-sampled random bits. For example, the adversary against a cryptographic protocol is represented by a function symbol $\mathsf{att}(\cdot)$ which obtains randomness from an array of random bits called the adversarial tape ρ_{a}. This random tape is shared across all calls to the adversary, allowing them to re-use past randomness if needed. Because this tape is fixed, it is implicitly passed to $\mathsf{att}(\cdot)$ by the semantics of the language.

We now present at a high-level how quantum (Turing) machine operates, and refer the reader to Appendix C for details. In a quantum Turing machine [16, 25], the probability $p_v \in \mathbb{R}^+$ to return bitstring v is replaced by a *complex amplitude* $q_v \in \mathbb{C}$. Then, the execution of a *quantum machine* $\mathcal{M}$ yields a superposition of bitstrings, which can conveniently be represented by a normalized element of the Hilbert space $\mathcal{H}_{\{0,1\}^*}$:

$$\mathcal{M}(u) = \textstyle\sum_{v \in \{0,1\}^*} q_v \cdot |v\rangle \quad \text{where} \quad \sum_v |q_v|^2 = 1 \quad \text{and} \quad \forall v.\, q_v \in \mathbb{C}.$$

A classical bitstring outputs can be obtained from the quantum value $\mathcal{M}(u)$ by *measuring* it, which yields v with probability $|q_v|^2$.

Init()	$\mathsf{Encap}_{\mathsf{real}}()$	Pub()
$sk \xleftarrow{\$} \mathcal{SK}$	**return** (c, k)	**return** pk
$pk \leftarrow \mathsf{pub}(sk)$	$\mathsf{Encap}_{\mathsf{ideal}}()$	Decap(c_0)
$(k, c) \xleftarrow{\$} \mathsf{encap}(pk)$	**return** (c, k^*)	**if** $c_0 \neq c$ **then**
$k^* \xleftarrow{\$} \mathcal{K}$		**return** $\mathsf{decap}(c_0, sk)$

Figure 1: The IND-CCA cryptographic game for KEMs.

We can also perform a partial measurement. For instance, we can measure the first N bits of a quantum superposition $\mathcal{M}(u)$ of bitstrings of length at least N, which yields a classical output and a quantum value:

$$\mathcal{M}(u) \xrightarrow[\text{partial measure}]{} \mathsf{Distr}\big(\{0,1\}^N \times \mathcal{H}_{\{0,1\}^*}\big)$$

where $\mathsf{Distr}(\mathbb{S})$ denotes the set of discrete distributions over $\mathbb{S}$. This is how a *quantum* adversary interacts with a *classical* protocol. The quantum adversary is provided with its quantum state st and classical inputs in coming from the protocol execution. After evaluating $\mathcal{M}(\mathsf{st}, \mathsf{in})$, a partial measurement is done to extract a classical value out to be forwarded to the protocol and the next quantum state st'.

$$\mathcal{M}(\mathsf{st}, \mathsf{in}) \xrightarrow[\text{partial measure}]{} (\mathsf{out}, \mathsf{st}') \in \mathsf{Distr}\big(\{0,1\}^N \times \mathcal{H}_{\{0,1\}^*}\big)$$

To model this, we need to provide eagerly sampled randomness for the quantum measurement. Squirrel already used eagerly sampled randomness, such as the adversarial tape mentioned above, to model the mechanisms by which classical computations obtain random bits. Here, we use eagerly sampled randomness to model the *physical mechanism* by which a quantum state can (partially) collapse according to some probabilistic computational rule. Concretely, we consider a dedicated name $\mathsf{qrnd} : \tau_0 \to \mathsf{qrand}$, where values of type qrand serve as random seed for a quantum measurement, and τ_0 allows providing independent randomness to each successive calls to $\mathcal{M}$. For instance, with msg and $\mathsf{Hilb}_{\mathsf{msg}}$ the Squirrel types for, resp., bitstrings and quantum bitstrings, the t-th call to the quantum adversary will be represented by:

$$\mathsf{att}(\mathsf{qrnd}\ t, (t, \mathsf{st}, \mathsf{in})) : \mathsf{msg} \star \mathsf{Hilb}_{\mathsf{msg}}$$

2.2 Interaction with the Protocol

We now present our new execution model for post-quantum cryptography. To clarify the presentation, we instantiate it on an example: IND-CCA for KEMs. We recall in Figure 1 the standard IND-CCA game for KEMs [23]. It comprises an initialization function which samples the *secret key* sk, computes the matching *public-key* $pk \leftarrow \mathsf{pub}(sk)$, and runs the encapsulation function encap to obtain the public *encapsulation* c, and its *output key* k. Then, the game provides three oracles. The adversary can obtain the public key pk from the Pub() oracle. The real version of Encap() returns the encapsulation (c, k), while the idealized oracle returns (c, k^*) where k^* is sampled at random during initialization. Finally, the $\mathsf{Decap}(c_0)$ oracle decapsulates any public encapsulation c_0 that is not the target encapsulation c. The procedures pub and decap

```
name qrnd : timestamp → qrand.
let rec frame (t : timestamp) : (timestamp ★ Hilb_msg ★ msg) =
 match t with Init → (t, ϵ, witness)
            | _ → (t, state t, transcript t)
and transcript (t : timestamp) : msg =
 match t with Init → ϵ
            | _ → ⟨ transcript (pred t), input t, output t ⟩
and state (t : timestamp) : Hilb_msg =
 match t with Init → witness
            | _ → att(qrnd (pred t), frame (pred t))#2
and input (t : timestamp) : msg =
 match t with Init → ϵ
            | _ → att(qrnd (pred t), frame (pred t))#1
and output (t : timestamp) : msg = ... (* protocol specific, uses input t *)
```

$(\cdot\#i)$ denotes tuple projection, i.e. $(t\#i)$ is the i-th element of a tuple t. To initialize the functions at time Init, we use ϵ for the empty bitstring, and witness for an arbitrary value of type Hilb_{msg}.

Figure 2: Post-quantum execution model.

are deterministic, whereas encap is probabilistic, even though no randomness is explicitly provided as a parameter to this procedure.

We model interaction of the adversary with the oracles of Figure 1 along a fixed execution trace. Such a trace can be seen as an ordered sequence of timestamps, each representing a particular call to an oracle of the protocol. To study the IND-CCA security of a KEM, we would rely on the timestamps:

$$(t : \mathsf{timestamp}) ::= \mathsf{Init} \mid \mathsf{Pub} \mid \mathsf{Encap} \mid \mathsf{Decap}\ j$$

Init, Pub and Encap represent a call to the corresponding oracle, and the timestamp (Decap j) is indexed by a value j of type index to distinguish different calls to Decap. Further, we consider:

$$< \ : \ \mathsf{timestamp} \to \mathsf{timestamp} \to \mathsf{bool}$$
$$\mathsf{pred} \ : \ \mathsf{timestamp} \to \mathsf{timestamp}$$

where $<$ specifies in which order oracles are called, and $\mathsf{pred}(t)$ is the timestamp preceding t in the execution trace. We assume that Init is minimal for $<$, and arbitrarily let $\mathsf{pred}(\mathsf{Init}) = \mathsf{Init}$.

We present in Figure 2 how we model the interaction of a quantum adversary with a protocol along a fixed execution trace, using functions defined by (mutual) recursion over the execution trace, ordered by $<$. The function transcript t represents the list of all messages sent over the network up-to time t, and is obtained by extending the previous transcript with input t and output t, that respectively model the input and output at time t. The function state t is the (quantum) state of the adversary at time t, and the knowledge frame t of the adversary is composed from t, its state, and the transcript up-to t. As discussed in Section 2.1, the attacker computations at time t is represented by:

$$\mathsf{att}(\mathsf{qrnd}\ (\mathsf{pred}\ t), \mathsf{frame}\ (\mathsf{pred}\ t))$$

where frame (pred t) is the knowledge of the adversary *before* calling oracle t, and qrnd (pred t) is the randomness used to model the partial measurement performed at the end of the quantum adversary's computation. Note that we give back the full frame to the attacker: while redundant for a stateful adversary, this simplifies proofs and brings the model closer to that of the classical attacker in Squirrel. The input and state at time t are obtained from this value

```
and output (t : timestamp) = match t with
 | Init → ϵ
 | Pub → (pk_1, pk_2)
 | Encap → ((c_1, c_2), h(⟨c_1, c_2⟩, dprf(k_1, k_2))
 | Decap j → if (input t)≠(c_1, c_2) then h(⟨(input t)#1,(input t)#2⟩, dprf(d_1, d_2))
```

where $pk_i = (\mathsf{pub}_i\ sk_i)$, $c_i = (\mathsf{encap}_i\ r_i\ pk_i)\#1$, $k_i = (\mathsf{encap}_i\ r_i\ pk_i)\#2$, and $d_i = (\mathsf{decap}_i\ (\mathsf{input}\ \mathsf{t})\#i\ ski)$.

Figure 3: Dual PRF KEM combiner: output macro.

using, resp., the first and second projection (see Figure 2). Finally, output t defines the output at time t using input t (and possibly past inputs), and is the only function that is protocol specific.

Example 1. *We give an example of a definition of* output *on a specific protocol. We consider the dual PRF hybrid KEM combiner as described in [17]. Informally, a dual PRF* $\mathsf{dprf}(k, x)$ *is a PRF when either the key material k or the input x is random. For $i \in \{1, 2\}$, (sk_i, pk_i) denotes the key pair, and* encap_i *and* decap_i *represent, resp., the encapsulation and decapsulation algorithm of KEM_i. The dual PRF KEM combiner is described in Figure 3.*

To construct a hybrid KEM from a dual PRF dprf*, a naive approach consists in applying* dprf *on top of the two output keys k_1 and k_2, i.e. encapsulation would return* $((c_1, c_2), \mathsf{dprf}(k_1, k_2))$*. This is actually insufficient to ensure that the KEM combiner satisfies IND-CCA as soon as this property holds on KEM_1 or KEM_2. Thus, instead of using* $\mathsf{dprf}(k_1, k_2)$ *as a shared secret, we use it as a key for another PRF function* h *applied to $\langle c_1, c_2\rangle$. Note that the randomness of the encapsulation procedure is explicitly represented here. To model the idealized version of the IND-CCA game, we replace in Figure 3 the part of the output of* Encap *corresponding to the output key with a random value rand, i.e., we replace* $((c_1, c_2), \mathsf{h}(\langle c_1, c_2\rangle, \mathsf{dprf}(k_1, k_2))$ *by* $((c_1, c_2), rand)$

Discussion. Modeling an execution along a fixed interaction implies a non-adaptive adversary. It can be generalized by following the approach of [9], but a non-adaptive presentation is simpler and better supported by Squirrel. The execution model implemented in Squirrel differs from our presentation in two ways: it supports oracles with calling restrictions, i.e. oracle that can only be called under some conditions; and it does not require that all timestamps are scheduled through the addition of a special undef timestamp. These simplifications have no bearing on the rest of the paper.

3 Core Logical Setting

We recall the core logical setting of higher-order CCSA logic [10] used in Squirrel. We stress that our approach uses this core logic without modifications, only imposing a particular semantics on specific types and predicates. As a result, the core logical rules of [10], such as the rewriting rule (Rewrite$_c$) seen in introduction, obviously remain valid in our instance of the logic.

3.1 Syntax of Terms

The logic relies on simply typed lambda terms. In this work, we assume that base types are organized into classical base types $\mathbb{B}$, notably containing bool and msg, and quantum superposition base types Hilb_{τ_b} for each $\tau_b \in \mathbb{B}$. Then, types τ are generated from

these base types using arrow and tuple constructs:

$$\tau ::= \tau_b \mid \mathsf{Hilb}_{\tau_b} \mid \tau \star \ldots \star \tau \mid \tau \to \tau \qquad (\tau_b \in \mathbb{B})$$

Terms are typed lambda terms, over a set of base symbols $\mathcal{S}$ which contains a set of *variables* $\mathcal{X}$:

$$t ::= s \mid \lambda x.\, t \mid t\ t \qquad (s \in \mathcal{S}, x \in \mathcal{X})$$

As in [10], we use a distinguished subset of symbols $\mathcal{N}$ to represent *names*, i.e. honest random samplings. A term is understood relatively to an *environment* $\mathcal{E}$, which is a list containing *declarations* $(s : \tau)$, and *definitions* $(s : \tau = t)$.

$$\mathcal{E} ::= \epsilon \mid \mathcal{E}, s : \tau \mid \mathcal{E}, s : \tau = t$$

where ϵ is the empty environment. Importantly, definitions may be recursive, subject to well-foundedness conditions [10]. A standard type system [10] expresses when a term t has type τ in $\mathcal{E}$, noted $\mathcal{E} \vdash t : \tau$. From now on, we only consider well-typed terms.

Example 2. *We usually work with standard function symbols on booleans, allowing to write logical formulas as boolean terms. When doing so we are assuming that the environment contains, e.g.,*

$$\wedge : \mathsf{bool} \to \mathsf{bool} \to \mathsf{bool} \qquad \forall_\tau : (\tau \to \mathsf{bool}) \to \mathsf{bool}$$
$$=_\tau : \tau \to \tau \to \mathsf{bool}$$

for all types τ. These symbols are used with usual syntactic sugar: we write $\phi \wedge \psi$ for $((\wedge)\ \phi)\ \psi$; we write $t = t'$ for $((=_\tau)\ t)\ t'$ when t, t' are terms of type τ; and $(\forall x : \tau.\, \phi)$ for $\forall_\tau(\lambda(x : \tau).\, \phi)$.

Example 3. *Figure 3 shows terms in our logic, in* Squirrel*'s concrete syntax. The associated environment contains primitives, e.g.,*

$$\mathsf{h} : \mathsf{msg} \to \mathsf{hkey} \to \mathsf{hash}$$
$$\mathsf{encap}_i : \mathsf{rand} \to \mathsf{kem\_pubkey} \to (\mathsf{msg} \star \mathsf{output\_key})$$

but also names, e.g. $r_i : \mathsf{rand}$. Pattern matching over timestamps is slightly more involved, but is obtained through a higher-order constant whose semantics is axiomatized (see [12, §A.2] for details).

3.2 Semantics of Terms

In a model $\mathbb{M}$ of the logic, each type τ is interpreted as a family of sets $[\![\tau]\!]^\eta_{\mathbb{M}}$, one for each value of the security parameter η. The parametrization in η is required, e.g., to model key spaces that vary with the security parameter. The interpretation of base types is given by the model, and lifted to arbitrary types in a standard manner [10]. In this work, we restrict to models where classical base types are countable (hence bitstring encodable) and where elements of Hilb_{τ_b} are quantum superpositions of elements of τ_b:

$$[\![\mathsf{Hilb}_{\tau_b}]\!]^\eta_{\mathbb{M}} \stackrel{\text{def}}{=} \mathcal{H}_{[\![\tau_b]\!]^\eta_{\mathbb{M}}} \qquad \text{for any } \tau_b \in \mathbb{B}$$

where $\mathcal{H}_{\mathbb{S}}$ denotes the Hilbert space with computational basis $\mathbb{S}$, as introduced in Appendix B.3. We also keep the usual assumption that $[\![\mathsf{bool}]\!]^\eta_{\mathbb{M}} = \{0,1\}$ and $[\![\mathsf{msg}]\!]^\eta_{\mathbb{M}} = \{0,1\}^*$ for all η.

A model also defines, for each value η of the security parameter, a set of random tapes $\mathbb{T}_{\mathbb{M},\eta}$. These tapes serve as the source of randomness for our terms, which are seen as probabilistic values. More precisely, a term of type τ is interpreted as an η-indexed family of random variables, i.e. for each η a function $f_\eta : \mathbb{T}_{\mathbb{M},\eta} \to [\![\tau]\!]^\eta_{\mathbb{M}}$. We let $\mathbb{RV}_{\mathbb{M}}(\tau)$ be the set of such families of random variables. In our models, tapes consists of pairs $\rho = (\rho_a, \rho_h)$ of finite bitstrings: ρ_a is used for the adversary's randomness, while ρ_h is used for honest random values.

$\mathbb{M}$ is a model w.r.t. some environment $\mathcal{E}$ when it provides, for each symbol s declared or defined in $\mathcal{E}$ with type τ, an interpretation $\mathbb{M}(s) \in \mathbb{RV}_{\mathbb{M}}(\tau)$. Given these interpretations, the semantics of any term t of type τ is obtained following natural (recursive) equations (see Appendix B.1 and [10] for details). When $\mathbb{M}$ is a model w.r.t. $\mathcal{E}$, and $a \in \mathbb{RV}_{\mathbb{M}}(\tau)$, we define the model $\mathbb{M}[s/a]$ w.r.t. $(\mathcal{E}, s : \tau)$, which extends $\mathbb{M}$ by mapping s to a, i.e. $(\mathbb{M}[s/a])(s) = a$.

We work with the usual assumptions on names [10], assuming that they correspond to i.i.d. values. More precisely, each name draws its randomness from a dedicated portion of the honest tape and computes from it, in polynomial time, a value in the desired interpretation domain. Finally, when working with environments containing symbols representing logical connectives, we only consider models where these connectives have their standard interpretation. For instance, $[\![\phi \wedge \psi]\!]^{\eta,\rho}_{\mathbb{M}} = 1$ iff. $[\![\phi]\!]^{\eta,\rho}_{\mathbb{M}} = [\![\psi]\!]^{\eta,\rho}_{\mathbb{M}} = 1$.

3.3 Global Formulas

Higher-order CCSA [10] is actually a first-order logic over the higher-order terms defined above, with specific cryptographic predicates. The syntax is standard, except that we decorate logical connectives to avoid confusion with the ones used in Example 2:

$$F ::= F_1 \mathbin{\tilde{\wedge}} F_2 \mid F_1 \mathbin{\tilde{\vee}} F_2 \mid F_1 \mathbin{\tilde{\Rightarrow}} F_2 \mid \tilde{\forall}(x : \tau).\, F \mid \tilde{\exists}(x : \tau).\, F$$
$$\mid [\phi] \mid [\phi]_e \mid \mathsf{const}(t) \mid \vec{u} \sim_c \vec{v} \mid \mathsf{adv}(t) \mid \ldots$$

We call these formulas *global*, and call *local formulas* the terms of type bool used as formulas as in Example 2.

We listed above the standard CCSA predicates [10] for doing classical cryptography, except that $\sim_c$ is usually noted $\sim$. We recall their semantics, which is fixed in our class of models. We will extend our logic with quantum predicates in the next section. For ϕ a term of type bool, the predicate $[\phi]$ represents *overwhelming truth*:

$$\mathbb{M} \models [\phi] \text{ iff. } \eta \mapsto \Pr([\![\phi]\!]^{\eta,\rho}_{\mathbb{M}} = 1) \text{ is overwhelming,}$$

where ρ is sampled uniformly in $\mathbb{T}_{\mathbb{M},\eta}$. For instance, $[u = v]$ states that the random variables u and v coincide with overwhelming probability. Then, $[\phi]_e$ is the exact variant of this predicate, where the probability must not just be overwhelming but equal to one. The predicate $\mathsf{const}(t)$ is satisfied when $[\![t]\!]^{\eta,\rho}_{\mathbb{M}}$ depends neither on η nor on ρ. Finally, when u and v are terms of type msg, $u \sim_c v$ means that the random variables are computationally indistinguishable, i.e. the following advantage is negligible for any PPTM $\mathcal{M}$:

$$|\Pr(\mathcal{M}(\eta, [\![u]\!]^{\eta,\rho}_{\mathbb{M}}, \rho_a) = 1) - \Pr(\mathcal{M}(\eta, [\![v]\!]^{\eta,\rho}_{\mathbb{M}}, \rho_a) = 1)|$$

More generally, we write $u_1, \ldots, u_n \sim_c v_1, \ldots, v_n$ when, for each i, the terms u_i and v_i have the same type of order at most 1. Order-0 types are classical base types, or tuples thereof, while order-1 types are those that can be written $\tau_1 \to \ldots \to \tau_k \to \tau'$ where τ_j $(1 \le j \le k)$ and τ' are of order 0. In that case, order-0 terms are given to the distinguisher $\mathcal{M}$ as multiple inputs, while order-1 terms give rise to oracles which may be queried by $\mathcal{M}$ on any term that it might compute. Finally, when t is of order 0 or 1, $\mathsf{adv}(t)$ means that t can be computed by a PPTM, i.e. a classical adversary.

4 Quantum Predicates

We now introduce the quantum predicates of our logic, and their semantics. This is the key ingredient to enable reasoning about quantum adversaries interacting with cryptographic protocols.

We formally define in Section 4.1 the $\mathsf{qadv}(\cdot;\cdot)$ predicate to capture polynomial-time quantum computations. This quantum variant of $\mathsf{adv}(\cdot)$ is technically much more involved, as we need to carefully track how randomness is used to model quantum measurements. This, in turn, allows us to define quantum indistinguishability in Section 4.2. Those two previous steps are carried out with finite tapes, as required by higher-order CCSA, which yields an imperfect, approximate model of quantum measurements. We show in Section 4.3 that, despite these approximations, our logic is sound with respect to true quantum computations.

4.1 Quantum Computations

We introduce quantum Turing machines and build on them to define a general model of polynomial-time hybrid adversaries, involving arbitrary interleavings of quantum and classical computations.

4.1.1 Quantum Turing machines. We briefly introduce our formal model for Quantum Turing Machines (QTM). Detailed definitions are given in Appendix C.2.

We follow traditional definitions from the literature, where a configuration of a quantum machine $\mathcal{M}$ is a superposition of classical configurations — recall that a classical configuration consists of a tape, a head position and a control state. The execution of a QTM then amounts to applying quantum operations to this quantum configuration. For ease of integration with our logic, we consider that QTMs are typed using the types of the logic: their inputs and outputs are in the domains of the types, and translated to and from bitstrings when needed.

The final output of $\mathcal{M}$ is obtained by performing a partial measure of its final state, and then only keeping the content of the output tapes. This is defined through the $\mathsf{Measure}^{\mathcal{M}}_{\mathbb{M},\eta}$ function.

If $\mathcal{M}$ is a QTM with outputs of type $\tau_1 \star \cdots \star \tau_m$, and $\mathcal{S}_{\mathcal{M}}$ represents the space of its quantum configurations, then we have:

$$\mathsf{Measure}^{\mathcal{M}}_{\mathbb{M},\eta} : \mathcal{S}_{\mathcal{M}} \to \mathsf{Distr}(\llbracket\tau_1\rrbracket^{\eta}_{\mathbb{M}} \times \cdots \times \llbracket\tau_m\rrbracket^{\eta}_{\mathbb{M}}).$$

In words, it maps the final quantum configuration of $\mathcal{M}$ to a probability distribution over possible outputs. Here, the output types τ_i may be classical or quantum base types: classical outputs are those which are observed in the partial measure, while quantum outputs are part of the remaining quantum state, still in superposition.

As usual in cryptography, machines have access to oracles. In a nutshell, this means that our machines have a special tape for writing oracle inputs. At any point in a computation, the machine may query an oracle of its choice on that input, and obtain (in a single computation step) the oracle's output to another dedicated tape. Oracles of a QTM have quantum inputs and outputs, and we call them *quantum-access oracles*. They may for instance be used in the quantum random oracle model (QROM) [20] to provide a quantum computation with access to a hash function, which it can use as part of quantum computations, i.e. to directly compute superpositions of hash functions.

Example 4. *Looking forward, we will represent quantum computations as terms in our logic. If f : msg → msg is meant to represent a quantum computation from bitstrings to bitstrings, then it must use the (implicit) random tapes to model the final measurement step — we are representing the distribution resulting from* Measure *as a random variable. However, the quantum computability of $(f\ 0)$ and $(f\ 1)$ may not imply that of the pair $\langle f\ 0, f\ 1\rangle$, if both calls to f rely on the same source of randomness to model the measurement step.*

In order to clearly deal with such issues, we will represent randomness explicitly — as is actually common in the CCSA approach. In this example, we would have f : rand → msg → msg, and f itself would be deterministic, i.e. independent of the implicit tapes from the semantics. In this style, assuming that r_0 and r_1 are two distinct names (i.e. two independent sources of randomness) we can safely claim that $\langle f\ r_0\ 0, f\ r_1\ 1\rangle$ is still quantum computable. Note that $f\ r_0\ (f\ r_1\ 1)$ is also quantum computable under the same assumption — this is the kind of nesting that happens in our execution model, where f is replaced by att *and r_i by* qrnd t.

We note that this modeling raises an issue: because our random tapes are finite, they cannot be used to perfectly model the probabilities of quantum measurements. We postpone this issue to Section 4.3.

4.1.2 Hybrid classical-quantum adversaries. A single QTM execution is not enough to correctly model an adversarial computation, which may be an interleaving of classical and quantum Turing Machines. Our final model for a quantum adversary allows such interleavings. In addition to the previously described quantum-access oracle, it notably introduces *classical-access* oracles, which may be queried during the classical computation phases. These oracles can typically be used to model a protocol party with which the quantum adversary interacts over the classical network, or the ROM.

To formally define such interleavings, we use sequences of random values S, where each element in the sequence provides the randomness used to model one quantum measurement. In Appendix C.2, we formally define $\mathsf{fold}_{\tau}(\mathcal{M}_c, \mathcal{M}_q, \mathbf{a}, S, \eta, \rho_{\mathrm{a}})$ which, starting with the input $\mathbf{a}$, interleaves computations between a classical machine $\mathcal{M}_c$ and quantum machine $\mathcal{M}_q$ that both have access to the adversarial random tapes ρ_{a}, and performs the successive measurements using the randomness source in S. Here, S both indicates the number of iterations to be performed, and provides the measurements' randomness.

For $R \in \mathbb{N}[X]$, we say that $(\mathcal{M}_c, \mathcal{M}_q, R)$ forms a PTIME hybrid computation if there exists a polynomial P such that, for all S of length $R(|\mathbf{a}| + \eta)$, the computation $\mathsf{fold}_{\tau}(\mathcal{M}_c, \mathcal{M}_q, \mathbf{a}, S, \eta, \rho_{\mathrm{a}})$ takes time at-most $P(|\mathbf{a}| + \eta)$ for all $\mathbf{a}$, η and ρ_{a}.

4.1.3 Adversarial computability predicate. We can finally define a predicate expressing when a term (of order at most 1) is computable by an adversary, i.e. a polynomial-time hybrid computation. For convenience, we will index measurement randomness over timestamps. To this end, we introduce a base type timestamp set to represent finite sets of elements of timestamp.

When t is an order-0 term, and t_S is of type timestamp set, we can form the predicate $\mathsf{qadv}(t; t_S)$. It is satisfied in a model $\mathbb{M}$ when $\llbracket t\rrbracket^{\eta,\rho}_{\mathbb{M}} = \mathsf{fold}_{\tau}(\mathcal{M}_c, \mathcal{M}_q, \epsilon, S, \eta, \rho_{\mathrm{a}})$ for some PTIME hybrid computation $(\mathcal{M}_c, \mathcal{M}_q, R)$, where S is the sequence of $\llbracket\mathsf{qrnd}\ \tau\rrbracket^{\eta,\rho}_{\mathbb{M}}$ for the successive values of $\tau \in \llbracket t_S\rrbracket^{\eta,\rho}_{\mathbb{M}}$. When t is an order-1

term, we naturally generalize the definition, asking that for any input $[\![x]\!]^{\eta,\rho}_{\mathbb{M}}$, $[\![t\ x]\!]^{\eta,\rho}_{\mathbb{M}} = \mathsf{fold}_\tau(\mathcal{M}_c, \mathcal{M}_q, [\![x]\!]^{\eta,\rho}_{\mathbb{M}}, S, \eta, \rho_{\mathsf{a}})$ (details in Appendix C.4).

4.1.4 Restricting the adversary. Equipped with our notion of hybrid classical-quantum adversarial computation, we can restrict our attention to models in which the adversarial function symbol att is a PQTM. Concretely, we only consider models $\mathbb{M}$ such that:

$$\mathbb{M} \models \tilde{\forall} t \in \mathsf{timestamp}.\ \mathsf{qadv}(\lambda x.\mathsf{att}(\mathsf{qrnd}\ t, x); t)$$

4.2 Quantum Indistinguishability

Let $\vec{u}, \vec{v}$ be two sequences of terms of types $\tau_1, \ldots, \tau_n$ of order at most one, and where every order-1 term is either of quantum type to quantum type, or of classical type to classical type, respectively modeling a quantum-access or a classical-access oracle. For any model $\mathbb{M}$, we let $\mathbb{M} \models \vec{u} \sim \vec{v}$ iff. for any hybrid classical-quantum computation $(\mathcal{M}_c, \mathcal{M}_q, R)$:

$$\eta \mapsto \left| \begin{array}{l} \Pr(\mathsf{fold}_\tau(\mathcal{M}_c, \mathcal{M}_q, [\![\vec{u}]\!]^{\eta,\rho}_{\mathbb{M}}, S, \eta, \rho_{\mathsf{a}}) = 1) \\ - \Pr(\mathsf{fold}_\tau(\mathcal{M}_c, \mathcal{M}_q, [\![\vec{v}]\!]^{\eta,\rho}_{\mathbb{M}}, S, \eta, \rho_{\mathsf{a}}) = 1) \end{array} \right| \text{ is negligible,}$$

where ρ and S are sampled independently at random in, resp., $\mathbb{T}_{\mathbb{M},\eta}$ and $([\![\mathsf{qrand}]\!]^{\eta}_{\mathbb{M}})^{R(\eta)}$.

Example 5. *Going back to Example 1, we denote by* $\mathsf{frame}^{\mathsf{Real}}$ *the recursive frame function obtained by considering Figures 2 and 3, and* $\mathsf{frame}^{\mathsf{Ideal}}$ *its idealized version described in Example 1, both modeling either the real or the ideal side of the IND-CCA KEM experiment. Then, the semantics of the formula* $\mathsf{frame}^{\mathsf{Real}} \sim \mathsf{frame}^{\mathsf{Ideal}}$ *matches the expectation: no adversary that has access to both quantum and classical computers can distinguish whether it interacts with the real or ideal part of the IND-CCA experiment.*

4.3 Adequacy of our Approximate Semantics

We now explain how we model quantum measurements using discrete random variables, and prove that our modeling is faithful.

4.3.1 Early-sampling discrete semantics. Recall that Squirrel uses an early-sampling semantics [10] where all randomness is sampled in advance and stored in a random tape ρ. In order for *global formulas* to have a well-defined semantics, the semantics $[\![t]\!]^{\eta}_{\mathbb{M}}$ of a *term* must be a *random variable* for any value η of the security parameter. To see why this is useful, consider t of type bool, and recall that:

$$\mathbb{M} \models [t] \ \text{ iff. } \ \eta \mapsto \Pr_{\rho \in \mathbb{T}_{\mathbb{M},\eta}}([\![t]\!]^{\eta,\rho}_{\mathbb{M}} = 1) \text{ is overwhelming}$$

The fact that $[\![t]\!]^{\eta}_{\mathbb{M}}$ is a random variable ensures that the probability is well-defined. This requirement is not limited to boolean terms and to, e.g., be able to define the semantics of a computational indistinguishability atom $u_0 \sim u_1$, we (roughly) need that for any adversary $\mathcal{A}$:

$$\Pr_{\rho \in \mathbb{T}_{\mathbb{M},\eta}}(\mathcal{A}([\![u_i]\!]^{\eta,\rho}_{\mathbb{M}}) = 1)$$

is well-defined, for any order-0 or order-1 terms u_0, u_1 (of type e.g. msg). To ensure that a term's semantics is always a random variable, [10] requires that the set of tapes $\mathbb{T}_{\mathbb{M},\eta}$ is always finite. Thus, Squirrel semantics is based on *discrete* random variables. This finite tape restriction solves the issue shown above, but poses difficulty to faithfully model *quantum measurements*.

4.3.2 Modeling quantum measurements. Recall that a partial quantum measurement maps a quantum value to a *distribution* of a classical bitstring output and a quantum output corresponding to the remaining unmeasured quantum state. As explained in Section 2, the randomness used to model a quantum measurement is retrieved from ρ using the dedicated name qrnd : timestamp set → qrand, where qrand is the type used for quantum measurement randomness. This creates two difficulties:

- Because of the finite tape restriction, qrand can only provide a finite number of random bits.
- Furthermore, while this number of bits can be arbitrarily large (as long as it is finite), it must be independent of the term being interpreted, as the type interpretation $[\![\mathsf{rand}]\!]^{\eta}_{\mathbb{M}}$ is the same for all terms. Thus, it is not possible to provide different amount of randomness to different terms.

This raises the following question: *how can we faithfully model quantum measurements in an early-samplings discrete semantics?*

4.3.3 Approximation and exact models. We now explain how we solve this modeling issue. Our approach relies on two kinds of *models* for our logic, which gives rise to two different *term* semantics:

- The *approximation models*, which are the ones defined in Section 3, provide a finite number of bits for quantum measurements through qrand. More precisely, we let $\mathsf{r}^{\eta}_{\mathbb{M}} \in \mathbb{N}$ be such that $[\![\mathsf{qrand}]\!]^{\eta}_{\mathbb{M}} = \{0,1\}^{2 \cdot \mathsf{r}^{\eta}_{\mathbb{M}}}$. This yields an *approximated term semantics* in which quantum measurements are approximated using $2 \cdot \mathsf{r}^{\eta}_{\mathbb{M}}$ unbiased random bits, following a procedure described below in Section 4.3.4.
- *Exact models* evaluate quantum measurements without errors using an infinite (but countable) number of random bits. More precisely, we let $\mathsf{r}^{\eta}_{\mathbb{M}} = \omega$ where ω denotes the first infinite ordinal, and:

$$[\![\mathsf{qrand}]\!]^{\eta}_{\mathbb{M}} = \{0,1\}^{\mathsf{r}^{\eta}_{\mathbb{M}}} = \{0,1\}^{\mathbb{N}} \qquad (\text{if } \mathsf{r}^{\eta}_{\mathbb{M}} = \omega)$$

  We equip this set with the Lebesgue measure, by identifying $\{0,1\}^{\mathbb{N}}$ with the real interval $[0,1]$. This yields an *exact term semantics* which faithfully models quantum measurements.

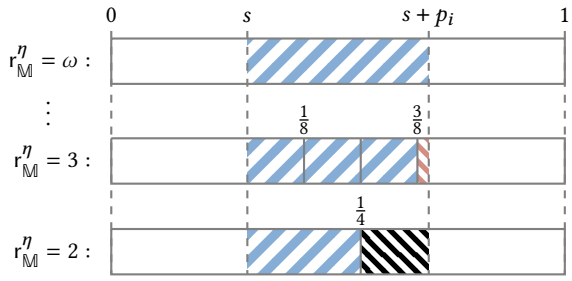
Approximation models are written $\mathbb{M}$, while exact model are written $\mathbb{M}_{\mathsf{e}}$. A model is an approximation model unless specified otherwise.

The semantics of a global formula is always well-defined in an approximation model, but may not be well-defined in an exact model (as explained in Section 4.3.1). Thus, the proof system we will present in Section 5, and the post-quantum version of Squirrel we implemented based on this proof system, rely on approximation models. More precisely, a global formula is *valid* if it is satisfied in all approximation models, and a rule

$$\frac{\Phi_1 \qquad \ldots \qquad \Phi_n}{\Phi}$$

is *sound* if Φ is valid whenever the premises $\Phi_1, \ldots, \Phi_n$ are valid.

This creates a modeling gap, as we show validity w.r.t. approximation models but expect it w.r.t. exact models. To close this gap, we will give some sufficient condition guaranteeing that a global formula has a well-defined exact semantics, and we will show that the global formulas expressing the usual security properties satisfy



We consider $e_i \neq e_{\text{default}}$, and pose $s = \sum_{j<i} p_j$. The area in hashed blue, indicates the probability $\tilde{p}_i$ that $\text{approx}_{\mathsf{r}}(\Delta)$ samples e_i. For $\mathsf{r} = \omega$, this is exactly p_i. For $\mathsf{r} < +\infty$, the quantity $\tilde{p}_i$ is always bound by p_i, and the approximation error is represented in hashed red. In this example, $\tilde{p}_i = \frac{1}{4}$ and the error is $p_i - \frac{1}{4}$ for $\mathsf{r} = 2$, and $\tilde{p}_i = \frac{3}{8}$ and the error is $p_i - \frac{3}{8}$ for $\mathsf{r} = 3$. The approximation error for all elements is redirected to e_{default}. If r is large enough, a small non-zero probability that can be sampled using r bits is added to every element (not depicted here), to ensure that the blue area $\tilde{p}_i$ is always strictly positive.

Figure 4: Graphical representation of an example of the approximation $\text{approx}_{\mathsf{r}}(\Delta)$ of a discrete distribution $\Delta = \sum_i p_i\, e_i$.

this criterion. Furthermore, we will prove that, under these conditions, the statistical distance between the approximated and exact semantics is negligible for models in which $\mathsf{r}^\eta_{\mathbb{M}}$ is large enough.

4.3.4 Approximated quantum measurements. As announced in Section 4.1.2, in a hybrid classical-quantum computation

$$\text{fold}_\tau\,(\mathcal{M}_c, \mathcal{M}_q, \mathbf{a}, S, \eta, \rho_{\mathsf{a}}),$$

we approximate each measurement $\text{Measure}^{\mathcal{M}_q}_{\mathbb{M},\eta}$ following an execution of the quantum machine $\mathcal{M}_q$ using a *discrete distribution approximation operator* $\text{approx}_{\mathsf{r}^\eta_{\mathbb{M}}}(\cdot)$. If $\Delta = \sum_i p_i\, e_i$ is a discrete distribution associating to each element e_i in its support a probability weight $p_i > 0$, then its approximation $\text{approx}_{\mathsf{r}}(\Delta) = \sum_i \tilde{p}_i\, e_i$ is a distribution that can be computed using $2 \cdot \mathsf{r}$ unbiased coin toss. Roughly, $\tilde{p}_i$ is an approximation to the r most-significant bits of p_i, which is then modified to ensure that $\tilde{p}_i$ is always strictly positive using the remaining r bits when r is large enough (if r is too small, it may not be possible to ensure that $\tilde{p}_i > 0$ for all i). We redirect the approximation error $p_i - \tilde{p}_i$ of all elements to an arbitrary point e_{default} in Δ's support. We provide a graphical example in Figure 4, and postpone details to Appendix B.4.

The exact model $\mathbb{M}_{\mathsf{e}}$ associated to an approximation model $\mathbb{M}$ is obtained from $\mathbb{M}$ by setting $\mathsf{r}^\eta_{\mathbb{M}}$ to ω, and by modifying the approximation of the measurements in the interpretation of att accordingly.

Remark 1. *Let $supp(X) = \{a \mid \Pr(X = a) > 0\}$ denote the support of X. Let t be a term, and consider its semantics $[\![t]\!]^\eta_{\mathbb{M}}$. When switching from an exact model $\mathbb{M}_{\mathsf{e}}$ to an associated approximation model $\mathbb{M}$, we are only changing the probabilities assigned to elements during a quantum measurement without introducing new elements, thus:*

$$supp([\![t]\!]^\eta_{\mathbb{M}}) \subseteq supp([\![t]\!]^\eta_{\mathbb{M}_{\mathsf{e}}}).$$

Further, we have equality of the supports if $[\![t]\!]^\eta_{\mathbb{M}}$ can only take a finite number of values and $\mathsf{r}^\eta_{\mathbb{M}}$ is large enough (see Appendix B.4.2).

4.3.5 Adequacy theorem. We identify a fragment of our logics for which adequacy holds. This fragment restricts the terms $u, v, \ldots$ that may occur in the predicates of a global formula that rely on the *probabilistic* semantics of the terms, i.e. the overwhelming truth $[\cdot]$ and indistinguishability $\sim$ predicates. We restrict these predicates to *well-formed terms*, which are essentially order-1 terms that contain no measurements, or order-0 terms that can be simulated by a PQTM (details are in Definition 4 in Appendix D.3). Some other predicates rely on the *functional* interpretation of the terms in a way that does not depend on how the measurement are performed. For instance, this is the case for all predicates that only care about the set of values that a term t may take, and not the actual probabilities assigned to each value. This notably includes $\text{const}(\cdot)$ when applied to a term of finite type, or the exact truth $[\cdot]_{\mathsf{e}}$, for which adequacy holds trivially by Remark 1: as soon as $\mathsf{r}^\eta_{\mathbb{M}}$ is large enough, we have equality of the supports of the predicates in both models and thus equality of their truth value. Finally, global quantification must be restricted to be over values whose semantics do not depend on the amount of quantum randomness available. Concretely, we limit quantification over $\text{const}(\cdot)$ variable — although alternative restrictions could be used, we use constancy, as it both simple and sufficient for our case-studies.

Essentially, a global formula is well-formed if all the terms appearing in the overwhelming truth $[\cdot]$ and equivalence predicates $\sim$ are well-formed, if quantifications are restricted to be over $\text{const}(\cdot)$ values, and if the terms appearing in the $[\cdot]_{\mathsf{e}}$ and $\text{const}(\cdot)$ atoms are of finite type. We are now ready to state our adequacy results (missing details can be found in Appendix D.4).

Theorem 1. *Let $\mathbb{M}$ be a model, $\mathbb{M}_{\mathsf{e}}$ the corresponding exact model, and F_{e} a well-formed global formula. If $\mathsf{r}^\eta_{\mathbb{M}}$ is large enough then:*

$$\mathbb{M} \models F_{\mathsf{e}} \quad \textit{iff.} \quad \mathbb{M}_{\mathsf{e}} \models F_{\mathsf{e}}.$$

Corollary 1. *Let F_{e} be a well-formed global formula. If:*

$$\models F_{\mathsf{e}} \quad \textit{then} \quad \mathbb{M}_{\mathsf{e}} \models F_{\mathsf{e}} \quad \textit{for any exact model } \mathbb{M}_{\mathsf{e}}.$$

5 Adapting SQUIRREL's Proof System

SQUIRREL is an interactive theorem prover reasoning over judgments of the form $\mathcal{E}; \Theta \vdash F$, where such a judgment is valid iff. the formula $\tilde{\wedge}\Theta \tilde{\Rightarrow} F$ is satisfied by all (approximation) models of $\mathcal{E}$. SQUIRREL relies on a proof system operating over such judgments, whose inference rules can be broadly split into five categories:

- Core logical rules, which do not rely on reductionist arguments and are purely logical. This actually includes the advanced **smt** tactic recently integrated within SQUIRREL [8].
- Basic reductionist rules capturing properties of $\sim$ like reflexivity, transitivity, or the fact that one can push some attacker computations from the terms into the top-level distinguisher.
- Rules allowing to automatically remove parts of an equivalence that are redundant and can be computed from its other elements (relying on the so-called bi-deduction [7]).
- Rules for the automated reduction of an equivalence to a user-given cryptographic game, using a generic simulator synthesis technique implemented in the **crypto** tactic [11, 12].
- Rules dedicated to specific cryptographic axioms. SQUIRREL provides such rules for a particular set of cryptographic assumptions, e.g. including PRF and IND-CCA.

We present, for each category mentioned above, how we adapt the corresponding rules to the quantum setting.

5.1 Core Logical Rules

Thanks to our general approach which fully embeds the quantum values inside the logic, many rules are directly inherited from [10]. For instance, consider the following rewriting rule:

$$\textsc{Rewrite}_{\mathsf{c}}\ \frac{\mathcal{E};\Theta \vdash \vec{u}, t \sim_{\mathsf{c}} \vec{v} \qquad \mathcal{E};\Theta \vdash [s = t]}{\mathcal{E};\Theta \vdash \vec{u}, s \sim_{\mathsf{c}} \vec{v}}$$

This rule relies on the fact that, since indistinguishability is up to a negligible probability of failure, it can absorb the difference between two overwhelmingly equal terms s and t. Its quantum counterpart Rewrite, obtained by replacing $\sim_{\mathsf{c}}$ by its quantum version $\sim$, is also sound, since $\sim$ is defined up-to a negligible probability of failure as well. Proving this rule's soundness does not involve reductionist or computational arguments. This allows us to consider, in intermediate proof steps, terms that cannot be produced by QTMs (e.g. because they duplicate a quantum state), but have nonetheless a well-defined semantics in our logic. For instance, the following rewriting is valid (though likely of little use):

$$\mathsf{att}(\mathsf{qrnd}\ t, 0) \quad \text{into} \quad \big(\mathsf{att}(\mathsf{qrnd}\ t, 0), \mathsf{att}(\mathsf{qrnd}\ t, 0)\big)\#1.$$

This constitutes one of the main advantages of our approach compared to [24]. Indeed, as we do not modify the semantics of the underlying higher-order logic, in contrast to [24], we directly inherit from [10] all rules that are not justified via reductionist arguments. This includes all standard logical rules such as quantifier introduction and case analysis, as well as more advanced rules like rewriting. Furthermore, we also obtain for free the **smt** tactic of Squirrel [8], which can automatically discharge certain proof obligations using purely logical, non-reductionist reasoning by delegating them to an SMT solver.

5.2 Basic Reductionist Rules

One of the most commonly used rules in Squirrel is the function application rule, which in a simplified form can be expressed as:

$$\mathrm{FA}_{\mathsf{c}}\ \frac{\mathcal{E};\Theta \vdash \mathsf{adv}(f) \qquad \mathcal{E};\Theta \vdash u \sim_{\mathsf{c}} v}{\mathcal{E};\Theta \vdash (f\ u) \sim_{\mathsf{c}} (f\ v)}$$

This rule states that classical indistinguishability is preserved by classical adversarial computation. The soundness of this rule follows from a straightforward reduction: any attacker distinguishing the conclusion can be transformed into an attacker distinguishing the premise by composing it with the attacker that computes f on its inputs. To lift this rule to the quantum setting, a natural idea would be to replace the classical predicate $\mathsf{adv}(\cdot)$ with its quantum version $\mathsf{qadv}(\cdot;\cdot)$. For example, this could allow us to exploit the fact that $\mathsf{qadv}(\lambda x.\mathsf{att}(\mathsf{qrnd}\ t, x); t)$ holds. However, this rule would be unsound, as illustrated below.

Example 6. *Given a distinguisher $\mathcal{A}$ for*

$$\mathsf{att}\big(\mathsf{qrnd}\ t, \mathsf{att}(\mathsf{qrnd}\ t, u)\big) \sim \mathsf{att}(\mathsf{qrnd}\ t, v) \tag{1}$$

it is not possible to construct a distinguisher $\mathcal{B}$ for $\mathsf{att}(\mathsf{qrnd}\ t, u) \sim v$ *by moving the top-level computation* $\mathsf{att}(\mathsf{qrnd}\ t, \cdot)$ *into $\mathcal{B}$. Indeed, when we move this computation inside $\mathcal{B}$, we also move the quantum measurement it includes. This creates an issue, as the quantum measurements in the top-level distinguisher of an indistinguishability $\sim$ — and thus in $\mathcal{B}$ — can only use fresh randomness (c.f. Section 4.2) while the randomness* $\mathsf{qrnd}\ t$ *is not fresh in Eq. (1).*

To detect quantum measurement randomness reuse like in Example 6, we need to precisely track which measurement randomness is being accessed in a term. To this end, we introduce a dedicated predicate, $\mathsf{qrand}(u; t_S)$, where u is an arbitrary term and t_S is of type timestamp set. This predicate states that u uses the name qrnd only with indices from t_S. For instance, if u is a term without binders and conditional, $\mathsf{qrand}(u; t_S)$ holds if $t \in t_S$ for any sub-term $(\mathsf{qrnd}\ t)$ of u. We leave the full definition to Appendix E.2.

Building on this predicate, we can define a sound function application rule for quantum indistinguishability $\sim$. This rule applies when f is an order-1 function, returning either a quantum or a classical value, and is stated as follows:

$$\textsc{FA:Single}\ \frac{\begin{array}{c}\mathcal{E};\Theta \vdash [t_S \cap t'_S = \emptyset]_{\mathsf{e}} \qquad \mathcal{E};\Theta \vdash \mathsf{qadv}(f; t_S) \\ \mathcal{E};\Theta \vdash \vec{u}, w_0 \sim \vec{v}, w_1 \\ \mathcal{E};\Theta \vdash \mathsf{qrand}(\vec{u}, w_0; t'_S)\ \tilde{\wedge}\ \mathsf{qrand}(\vec{v}, w_1; t'_S)\end{array}}{\mathcal{E};\Theta \vdash \vec{u}, f\ w_0 \sim \vec{v}, f\ w_1}$$

It formally states that if f can be simulated by a quantum machine that uses only t_S as source of measurement randomness, and if t_S is disjoint from the measurement randomness used by $\vec{u}, \vec{v}, w_0$ and w_1, then the computations performed by f can be pushed inside the top-level distinguisher. We provide the full set of such rules, together with their soundness proofs, in Appendix E.2.

5.3 Bi-Deduction

Squirrel relies on a *bi-deduction* engine [7] to automate many reasoning steps on the indistinguishability predicate. At its core, bi-deduction captures reasoning steps that can be justified by basic cryptographic reductions, that is, reductions which do not rely on cryptographic hardness assumptions.

Example 7. *Consider the following reasoning step:*

$$\frac{u_0, v_0, \lambda(x : \mathsf{msg}).\, H_0(x) \ \sim_{\mathsf{c}}\ u_1, v_1, \lambda(x : \mathsf{msg}).\, H_1(x)}{u_0, H_0(u_0) \ \sim_{\mathsf{c}}\ u_1, H_1(u_1)}$$

This result follows directly from a simple reduction argument. Consider an adversary $\mathcal{A}$ that distinguishes the lower indistinguishability statement. We construct an adversary $\mathcal{B}$ against the upper indistinguishability by defining $\mathcal{B}(u, v, f) \stackrel{\text{def}}{=} \mathcal{A}(u, f\ u)$. We have:

$$\mathcal{B}(u_i, v_i, \lambda(x : \mathsf{msg}).\, H_i(x)) = \mathcal{A}(u_i, H_i(u_i)),$$

and thus, the advantage of $\mathcal{B}$ against the premise is exactly the advantage of $\mathcal{A}$ against the conclusion. By assumption, $\mathcal{B}$ has negligible advantage, which immediately implies that $\mathcal{A}$ does as well.

Classical bi-deduction. We recall the classical bi-deduction framework, already used in Squirrel to automate basic reductionist arguments. In a classical bi-deduction judgment:

$$\#(\vec{u}_0; \vec{u}_1) \rhd_{\mathsf{c}} \#(\vec{v}_0; \vec{v}_1)$$

the sequences of terms $\vec{u}_0$, $\vec{u}_1$ (resp. $\vec{v}_0, \vec{v}_1$) have the same length, and are sometimes denoted in bold as $\vec{\boldsymbol{u}}$ (resp. $\vec{\boldsymbol{v}}$) to emphasize that they form a pair. Moreover, all terms in $\vec{u}_0, \vec{u}_1, \vec{v}_0, \vec{v}_1$ are classical

and of order at most 1. The judgment $\#(\vec{u}_0;\vec{u}_1) \rhd_c \#(\vec{v}_0;\vec{v}_1)$ holds iff. there exists a polynomial-time function f, i.e. a function satisfying $\mathsf{adv}(f)$, such that for both $i \in \{0,1\}$, the outputs $\vec{v}_i$ can be computed from the inputs $\vec{u}_i$ using f, i.e. $f(\vec{u}_i) = \vec{v}_i$.

The following rule establishes the connection between classical bi-deduction and classical indistinguishability:

$$\textsc{Bideduce-}\rhd_c \quad \frac{\vec{u}_0 \sim_c \vec{u}_1 \qquad \#(\vec{u}_0;\vec{u}_1) \rhd_c \#(\vec{v}_0;\vec{v}_1)}{\vec{v}_0 \sim_c \vec{v}_1}$$

The soundness of this rule relies on reduction similar to the one used in Example 7. For an overview of the bi-deduction proof system employed by Squirrel, we refer the reader to [5, 7].

Quantum bi-deduction. We want our quantum extension of Squirrel to support bi-deduction. To achieve this, we need to generalize the set of functions f that can witness a bi-deduction, in order to include functions with quantum inputs and outputs. For simplicity, we restrict ourselves to error-free quantum functions [16]. An error-free QTM $\mathcal{M}$ is a machine such that, for all input w, its output distribution $\mathsf{Measure}(\mathcal{M}(w))$ is a Dirac $\mathbb{1}_{w'}$. In such a case, we can omit the measurement $\mathsf{Measure}(\cdot)$, and directly write $\mathcal{M}(w) = w'$. We introduce a new predicate $\mathsf{qadv}_{\mathsf{E}}(f)$ stating that f is an error-free polynomial-time function (details are in Appendix C.4.3). This predicate is simpler than the more general $\mathsf{qadv}(\cdot;\cdot)$ predicate, as it does not need to track the randomness used in measurements. In turn, this allows us to avoid tracking randomness usage in the quantum bi-deduction predicate $\rhd$, which yields a simpler proof-system that is amenable to automation.

In a quantum bi-deduction $\#(\vec{u}_0;\vec{u}_1) \rhd \#(\vec{v}_0;\vec{v}_1)$, we allow quantum inputs and outputs of order at-most zero. Our execution model for our quantum machines (c.f. Appendix C.2) supports machines with a single quantum output. Therefore, we require that each $\vec{v}_i$ contains at most one term of quantum type. The semantics of $\rhd$ is then a straightforward generalization of that of $\rhd_c$ to quantum error-free computations f:

$$\mathbb{M} \models \#(\vec{u}_0;\vec{u}_1) \rhd \#(\vec{v}_0;\vec{v}_1) \qquad \text{if and only if,}$$
$$\mathbb{M} \models \tilde{\exists} f.\ \mathsf{qadv}_{\mathsf{E}}(f) \mathbin{\tilde{\wedge}} [f(\vec{u}_0) = \vec{v}_0]_{\mathsf{e}} \mathbin{\tilde{\wedge}} [f(\vec{u}_1) = \vec{v}_1]_{\mathsf{e}}$$

It is related to equivalence with respect to a quantum adversary via the rule Bideduce-$\rhd$, which is the counterpart of the classical Bideduce-$\rhd_c$ obtained by replacing $\sim_c$ and $\rhd_c$ by $\sim$ and $\rhd$.

Furthermore, we can adapt the proof rules for the classical bi-deduction predicate to their quantum counterparts, ensuring that quantum inputs are used linearly, i.e. at most once. A typical example of this is the transitivity rules for classical (left) and quantum (right) bi-deduction:

$$\textsc{BD.Trans-}\rhd_c \quad \frac{\vec{u} \rhd_c \vec{w} \qquad \vec{u},\vec{w} \rhd_c \vec{v}}{\vec{u} \rhd_c \vec{v}} \qquad\qquad \textsc{BD.Trans-}\rhd \quad \frac{\vec{u}_c,\vec{u}_q^1 \rhd \vec{w} \qquad \vec{u}_c,\vec{u}_q^2,\vec{w} \rhd \vec{v}}{\vec{u}_c,\vec{u}_q^1,\vec{u}_q^2 \rhd \vec{v}}$$

In the classical rule, the original input $\vec{u}$ can be fully reused in the second part of the computation: $\vec{u},\vec{w} \rhd \vec{v}$. In the quantum rule, only the classical components $\vec{u}_c$ of the inputs can be duplicated, while the quantum components $\vec{u}_q^1, \vec{u}_q^2$ must be split between the first and second premises.

The full proof-system for quantum bi-deduction is presented in Appendix E.3. It functions essentially like the classical bi-deduction proof system [5, 7], except that *quantum inputs* must be used *linearly*. There are no restrictions on classical inputs.

5.4 Cryptographic Bi-Deduction

We now turn to adapting Squirrel's tactics for capturing cryptographic reductions. The most flexible such tactic in Squirrel is the **crypto** tactic [11]. This tactic relies on the notion of *cryptographic bi-deduction*, which extends the bi-deduction judgment presented in Section 5.3 with additional annotations that track various properties of the simulator being constructed. We provide a high-level account of how this tactic can be used in our post-quantum setting, and defer the full formalization to Appendix E.4.

Classical cryptographic bi-deduction. Consider a secure cryptographic game $\mathcal{G}$, defined by a set of oracles with which an attacker may interact. Informally, a cryptographic bi-deduction judgment

$$\boldsymbol{C};(\boldsymbol{\phi},\boldsymbol{\psi}) \vdash \vec{\boldsymbol{u}} \rhd_{\mathcal{G}} \vec{\boldsymbol{v}}$$

states that for any model $\mathbb{M}$, there exists a simulator $\mathcal{S}$ with access to the oracles of $\mathcal{G}$ such that:

- $\mathcal{S}$ is a polynomial-time program;
- for each $i \in \{0,1\}$, the execution of $\mathcal{S}$, starting from a memory satisfying ϕ_i and given the inputs $[\![\vec{u}_i]\!]_{\mathbb{M}}$, produces the outputs $[\![\vec{v}_i]\!]_{\mathbb{M}}$ and results in a memory satisfying ψ_i;
- the *randomness footprint* of $\mathcal{S}$ is characterized by $\boldsymbol{C}$ (intuitively, any random sampling performed by $\mathcal{S}$ is recorded in $\boldsymbol{C}$, which is crucial, in particular, to ensure that $\mathcal{S}$ does not access the game's randomness).

The precise syntax and semantics of cryptographic bi-deduction are quite technical, we refer the interested reader to [11] for details.

Cryptographic bi-deduction enables the synthesis of cryptographic simulators for a given game $\mathcal{G}$, and thus supports cryptographic reductions. This connection is formalized by the following rule from [11], which links cryptographic bi-deduction to computational indistinguishability. For clarity, we present a simplified version of the rule, omitting the constraints $\boldsymbol{C}$, the pre- and post-conditions $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$, as well as all associated side conditions:

$$\textsc{Reduction}_c \quad \frac{\emptyset \rhd_{\mathcal{G}} \#(\vec{u}_0;\vec{u}_1)}{\vec{u}_0 \sim_c \vec{u}_1}$$

We assume here that $\mathcal{G}$ is secure against any polynomial-time classical adversary.

Quantum cryptographic bi-deduction. Our goal is to extend the **crypto** tactic of [11] to enable the synthesis of simulators for post-quantum cryptography. As for standard bi-deduction in Section 5.3, one possible approach would be to design a quantum variant of cryptographic bi-deduction, capable of synthesizing *quantum* simulators rather than *classical* ones. However, this would be a challenging undertaking. Cryptographic bi-deduction is already significantly more complex than standard bi-deduction, and adapting it to a quantum setting would further increase the complexity of the judgment and require revisiting the entire proof system of [11]. Instead, we adopt a simpler approach in which we synthesize a *classical* simulator $\mathcal{S}$ against a game $\mathcal{G}$ that is *extended* with a quantum API $\mathcal{Q}$. This API allows $\mathcal{S}$ to perform quantum computations in a manner that is safe by construction. Concretely, $\mathcal{Q}$ encapsulates a quantum state that

```
game Q = {
 (* quantum state *)
 var state : H_message = witness;
 (* classical state, storing the latest protocol input computed by the attacker *)
 var input : message = empty;
 (* update state using att *)
 oracle step (t, tr) = {
  r <-$ qrand;
  (input,state) = att(r, (t,state,tr)); }
 (* retrieve the last input given by the attacker *)
 oracle get_input () = { return input; }
}
```

Figure 5: A safe quantum API.

can only be manipulated through a restricted interface, ensuring that quantum operations remain well-formed. In particular, the API provides no mechanism for cloning the quantum state.

We present such a safe quantum API in Figure 5. Its internal state consists of a single quantum variable state and a classical variable input. The simulator $\mathcal{S}$ can invoke the oracle step to update the value of state. This oracle samples fresh quantum randomness r and updates state using $\mathsf{att}(\cdot)$ together with additional classical inputs provided by the simulator (t denotes a timestamp and tr the transcript). Moreover, step stores the next input produced by $\mathsf{att}(\cdot)$ in the variable input. At any time, the simulator may retrieve the value of the classical variable input by calling the get_input oracle, since classical values can be safely extracted from the API.

The quantum API Q ensures that the quantum state is handled safely, *i.e.* according to a feasible quantum computation. In particular, the variable state cannot be cloned and can only be updated through the quantum polynomial-time procedure $\mathsf{att}(\cdot)$ using fresh quantum randomness. As a result, for any *classical* polynomial-time program $\mathcal{S}$, the interaction $\mathcal{S}^Q$ between $\mathcal{S}$ and Q constitutes a *quantum* polynomial-time program. Consequently, Q can be used to construct quantum simulators against the game $\mathcal{G}$, thereby establishing quantum indistinguishability via quantum reductions to $\mathcal{G}$. Informally, we have a rule of the form:

$$\frac{\emptyset \rhd_{\mathcal{G}\cdot Q} \#(\vec{u}_0; \vec{u}_1)}{\vec{u}_0 \sim \vec{u}_1}$$

To justify the soundness of this rule, we observe that the cryptographic bi-deduction premise guarantees the existence of a simulator $\mathcal{S}$ such that $\mathcal{S}$ is a classical polynomial-time program and:

$$\forall i \in \{0,1\}, \mathcal{S}^{\mathcal{G}_i \cdot Q}() = [\![\vec{u}_i]\!].$$

Since Q is a safe quantum API and $\mathcal{S}$ is a *classical* polynomial-time program, it follows that the composed program $\mathcal{S}_0^{\mathcal{G}_i} \stackrel{\text{def}}{=} \mathcal{S}^{\mathcal{G}_i \cdot Q}$ is a *quantum* polynomial-time program. The equation above can therefore be rewritten as:

$$\forall i \in \{0,1\}, \mathcal{S}_0^{\mathcal{G}_i}() = [\![\vec{u}_i]\!]$$

Hence, $\mathcal{S}_0$ constitutes a valid quantum simulator with respect to the game $\mathcal{G}$, and the indistinguishability $\vec{u}_0 \sim \vec{u}_1$ holds provided that $\mathcal{G}$ is secure against quantum polynomial-time adversaries.

In Appendix E.4, we exploit the safe quantum API of Figure 5 to design an induction rule specialized for our quantum execution model. This induction rule handles the quantum state in the background, and only asks the user to prove that the output of the protocol can be simulated when they are provided with the past inputs, which can be done in practice using a *classical* simulator. Using this induction rule and the existing machinery for classical cryptographic bi-deduction, we obtain a sound automated procedure for quantum simulator synthesis.

5.5 Dedicated Cryptographic Tactics

Squirrel provides builtin tactics for a small set of common cryptographic assumptions. While the bi-deduction-based **crypto** tactic is quite powerful, builtin tactics are tailor-made for typical use cases and can thus be easier to use. Each builtin cryptographic tactic checks a specific syntactic side condition, in order to ensure that the terms under consideration can be simulated using the relevant cryptographic game. For instance, for IND-CPA, we must notably verify that the encryption key only occurs in key position in the terms. Further, these tactics verify that terms can be simulated in polynomial-time: this is checked through the pptm judgment from [10], which is verified in the tool using a simple, syntax-directed automatic procedure.

To obtain quantum variant of these tactics, we must modify the syntactic side condition to allow for quantum simulators that manipulate the quantum state in a valid way. Naturally, we must also ensure that these quantum simulators run in polynomial-time.

These modifications are in fact similar to what we did for **crypto**: on the theoretical side, we adapt the pptm judgment into a new pqtm judgment, by adapting its inductive rule using a similar approach based on the quantum-safe API from Figure 5; on the practical side, this gives rise to the same syntactic side conditions ensuring that the quantum state manipulations required for simulating the terms fall into our safe fragment.

6 Implementation and Case Studies

In this section, we provide details about the implementation and the case studies carried out to validate our new approach. The case studies fall into two categories. First, we conducted proofs for several KEM combiners to establish their robustness under standard assumptions such as CPA and CCA. Second, we proved strong secrecy for two key-exchange protocols from the literature. In particular, we reproved the BCGNP protocol [21] which had already been analyzed in PQ-Squirrel [24], as an argument showing that our approach does not lose in expressivity w.r.t. that prior one. The results obtained are summarized in Table 1.

The proofs presented in this section were initially developed in the classical setting and subsequently adapted to the post-quantum setting. While writing the initial set of proofs required a substantial effort (approximately one to two person-months), their adaptation to PQC proved comparatively straightforward and mainly involved minor adjustments aimed at limiting explicit manipulation of quantum states. Overall, this adaptation was completed in less than two days. Note that none of these case studies can be carried out using PQ-Squirrel as they rely on, e.g., the **crypto** or **smt** tactics, which are not available in that framework. Note that [24] had to go

around the lack of support for generic cryptographic games (which is provided by the **crypto** tactic) by modeling the KEM used in BCGNP as an asymmetric encryption. In our proof for this protocol, we directly rely on the IND-CCA game for KEMs.

6.1 Implementation

The new execution model presented in Section 2, together with the associated proof system described in Section 5, have been integrated into Squirrel. The additional checks required by the tactic **fa**, which is based on the function application rule of Section 5.2, are automatically done by the tool. Concretely, **fa** checks the freshness of the randomness of a quantum measurement using Squirrel existing machinery used by the information-theoretic **fresh** tactic. We adapted the **deduce** tactic, which exploits the bi-deduction rules of Section 5.3, by implementing a linearity check for quantum inputs. For **crypto**, we implemented our new specialized induction rule and reused the existing cryptographic bi-deduction machinery. Finally, no adaptation was necessary for **smt**, since, as mentioned earlier, the quantum version of this tactic is obtained "for free".

6.2 KEM Combiners

We analyze several KEM combiners from the literature to establish their CPA or CCA robustness. For instance, for the Dual-PRF KEM combiner introduced in Section 2, we show that it satisfies CCA security assuming a dual PRF and that KEM1 is CCA. The development is roughly 740 LoC long, with 170 LoC dedicated to the specification; the remaining lines are proof-scripts. Squirrel verifies this proof in approximately 10 seconds on a standard laptop. For the Dual-PRF combiner, we prove that no quantum adversary can distinguish between the real and ideal games of Example 1 using a sequence of explicit game-hops written in the file.

Among the combiners we studied, most are symmetric constructions, and therefore the proofs are essentially identical whether the security assumption is placed on KEM1 or KEM2. These symmetric constructions include XOR, XOR-then-MAC, and Dual-PRF. For the Nested Dual-PRF combiner, which is not symmetric, we developed two distinct but similar proofs. For all these combiners, we investigated CPA and CCA security under the appropriate assumption on the underlying KEM. The CPA proofs are not subsumed by the CCA ones as the former ones are established only using CPA as an assumption. Further, the proofs regarding CCA security are more involved. Indeed, designing a hybrid CPA from two CPA is relatively easy, whereas building a hybrid CCA from two CCA implies some ciphertext integrity property, which is not trivial to get when one of the two KEM may be insecure. Note that we did not consider the CCA analysis of the XOR construction, since it is well known that the XOR combiner does not achieve CCA security.

6.3 Key Exchange Protocols

We carried out the analysis of the BCGNP key exchange protocol [21], a two-message protocol that establishes a session key of the form $\mathrm{expd}(\mathit{transcript}, k_I) \oplus \mathrm{expd}(\mathit{transcript}, k_R)$, computed from the output keys k_I and k_R of the exchanged encapsulations. Strong secrecy of this session key, from both the initiator's and the responder's perspective, is established in two separate files. The corresponding entry in Table 1 reports the sum (in terms of LoC and execution time) of these two files. Each proof relies on an intermediate protocol representing the idealized variant in which the key k_I (resp. k_R) is replaced with fresh randomness.

Finally, we analyzed a more complex key-exchange protocol taken from [17], whose specification is given in Figure 6 of the appendix. The protocol, denoted $C_{\mathtt{SigMA}}$, combines a KEM with Sigma style authentication [28], and additionally relies on signatures, a MAC primitive, and a KDF to derive the session key. Our focus is on establishing strong secrecy of the session key K from the responder's point of view. As before, the proof proceeds via an idealization step. We first carry out the analysis in a setting with only honest agents and a single session, and then extend the result to the more complex case of an arbitrary number of sessions. These proofs rely on several intermediate authentication lemmas ensuring that whenever an agent terminates, the protocol has proceeded as expected and both agents agree on the entire exchanged transcript.

All the proofs we conducted were obtained first without using the **smt** tactic. As a result, our developments do not depend on the installation of any external solver. Nevertheless, the **smt** tactic can be used to carry out proofs in the presence of a quantum attacker. To illustrate this point, we performed an additional proof on the $C_{\mathtt{SigMA}}$ protocol in the simpler scenario, this time relying on **smt**. As shown in Table 1, using **smt** reduces the length of the proof scripts by 25%, although it doubles its execution time.

7 Conclusion

We designed and implemented a novel version of Squirrel that supports the mechanization of post-quantum cryptographic proofs. From a theoretical point-of-view, this was made possible by embedding quantum values inside Squirrel's logic, and by distancing the semantics of the logic from quantum computability requirements. This novel approach is more maintainable and expressive than the earlier PQ-Squirrel. In particular, we have been able to adapt to our setting all recent improvements to Squirrel, including its **crypto** tactic. We are confident that our framework will enable the integration of future developments with post-quantum reasoning. Our post-quantum version of Squirrel enabled us to carry-out novel case studies, taking advantage of Squirrel's latest additions.

Our approach paves the way for new directions of research in the CCSA line of work. Because our logic explicitly represents quantum values, we can notably look toward supporting the QROM, or even extend Squirrel to prove the security of quantum protocols, *i.e.* protocols exploiting quantum mechanics for security purposes.

Acknowledgments

This work received funding from the France 2030 program managed by the French National Research Agency under the RESQUE project, as well as by grant agreements ANR-22-PECY-0006 and ANR-22-PETQ-0008. This work was also supported by the French National Agency for Research as part of the HOPR project ANR-24-CE48-5521-01.

References

[1] Carmine Abate, Philipp G. Haselwarter, Exequiel Rivas, Antoine Van Muylder, Théo Winterhalter, Catalin Hritcu, Kenji Maillard, and Bas Spitters. 2021. SSProve: A Foundational Framework for Modular Cryptographic Proofs in Coq. In *34th*

Protocols	Assumptions	Properties	LoC	Time
XOR Combiner [27]	CPA KEM1	CPA	140/80	< 1s
XOR-then-MAC Combiner [17]	CPA KEM1, EUF mac	CPA	170/80	1s
	CCA KEM1, EUF mac	CCA	170/1000	15s
Dual-PRF Combiner [17]	CPA KEM1, dual PRF	CPA	170/90	1.5s
	CCA KEM1, dual PRF	CCA	170/570	10s
Nested Dual-PRF Combiner [17]	CPA KEM1, dual PRF, PRF	CPA	180/130	2s
	CCA KEM1, dual PRF, PRF	CCA	180/700	14s
	CPA KEM2, dual PRF	CPA	180/100	1.5s
	CCA KEM2, dual PRF	CCA	180/600	11s
$C_{\texttt{SigMA}}$ [17] - one session	CPA KEM, PRF kdf, EUF sign	Secrecy & Auth.	250/1250	26s
- one session using **smt**	CPA KEM, PRF kdf, EUF sign	Secrecy & Auth.	250/900	51s
- multiple sessions	CPA KEM, PRF kdf, EUF sign	Secrecy & Auth.	250/1750	110s
BCGNP [21]	CPA KEM with public-key reuse, PRF	Secrecy	260/920	16s

Table 1: Summary of our case studies in Squirrel. LoC x/y indicates the number of LoC for the specification/proof.

IEEE Computer Security Foundations Symposium, CSF 2021, Dubrovnik, Croatia, June 21-25, 2021. IEEE, 1–15. doi:10.1109/CSF51468.2021.00048

[2] José Bacelar Almeida, Manuel Barbosa, Gilles Barthe, Benjamin Grégoire, Vincent Laporte, Jean-Christophe Léchenet, Tiago Oliveira, Hugo Pacheco, Miguel Quaresma, Peter Schwabe, Antoine Séré, and Pierre-Yves Strub. 2023. Formally verifying Kyber Episode IV: Implementation correctness. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2023, 3 (2023), 164–193.

[3] José Bacelar Almeida, Santiago Arranz Olmos, Manuel Barbosa, Gilles Barthe, François Dupressoir, Benjamin Grégoire, Vincent Laporte, Jean-Christophe Léchenet, Cameron Low, Tiago Oliveira, Hugo Pacheco, Miguel Quaresma, Peter Schwabe, and Pierre-Yves Strub. 2024. Formally Verifying Kyber - Episode V: Machine-Checked IND-CCA Security and Correctness of ML-KEM in EasyCrypt. In *CRYPTO (2) (Lecture Notes in Computer Science, Vol. 14921)*. Springer, 384–421.

[4] Apple. [n. d.]. iMessage with PQ3: The new state of the art in quantum-secure messaging at scale. https://security.apple.com/blog/imessage-pq3/.

[5] David Baelde, Stéphanie Delaune, Charlie Jacomme, Adrien Koutsos, and Joseph Lallemand. 2024. The Squirrel Prover and its Logic. *ACM SIGLOG News* 11, 2 (2024), 62–83.

[6] David Baelde, Stéphanie Delaune, Charlie Jacomme, Adrien Koutsos, and Solène Moreau. 2021. An Interactive Prover for Protocol Verification in the Computational Model. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*. IEEE, 537–554.

[7] David Baelde, Stéphanie Delaune, Adrien Koutsos, and Solène Moreau. 2022. Cracking the Stateful Nut: Computational Proofs of Stateful Security Protocols using the Squirrel Proof Assistant. In *CSF*. IEEE, 289–304.

[8] David Baelde, Stéphanie Delaune, and Stanislas Riou. 2025. SMT-Based Automation for Overwhelming Truth. In *CSF*. IEEE, 505–520.

[9] D. Baelde, C. Fontaine, A. Koutsos, G. Scerri, and T. Vignon. 2024. A Probabilistic Logic for Concrete Security. In *2024 IEEE 37th Computer Security Foundations Symposium (CSF)*. IEEE Computer Society, Los Alamitos, CA, USA, 484–499. doi:10.1109/CSF61375.2024.00032

[10] David Baelde, Adrien Koutsos, and Joseph Lallemand. 2023. A Higher-Order Indistinguishability Logic for Cryptographic Reasoning. In *LICS*. IEEE, 1–13.

[11] David Baelde, Adrien Koutsos, and Justine Sauvage. 2024. Foundations for Cryptographic Reductions in CCSA Logics. In *CCS*. ACM, 2814–2828. https://hal.science/hal-04511718v3

[12] David Baelde, Adrien Koutsos, and Justine Sauvage. 2026. Leveraging Cryptographic Simulator Synthesis for Formally Verifying the FOO E-Voting Protocol. In *Usenix 2026 - 35th Usenix security symposium*. Baltimore, United States. https://inria.hal.science/hal-05453231

[13] Manuel Barbosa, Gilles Barthe, Karthik Bhargavan, Bruno Blanchet, Cas Cremers, Kevin Liao, and Bryan Parno. 2021. SoK: Computer-Aided Cryptography. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*. IEEE, 777–795. doi:10.1109/SP40001.2021.00008

[14] Manuel Barbosa, Gilles Barthe, Xiong Fan, Benjamin Grégoire, Shih-Han Hung, Jonathan Katz, Pierre-Yves Strub, Xiaodi Wu, and Li Zhou. 2021. EasyPQC: Verifying post-quantum cryptography. In *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security*. 2564–2586.

[15] Gilles Barthe, Benjamin Grégoire, Sylvain Heraud, and Santiago Zanella-Béguelin. 2011. Computer-Aided Security Proofs for the Working Cryptographer. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings (Lecture Notes in Computer Science, Vol. 6841)*, Phillip Rogaway (Ed.). Springer, 71–90. doi:10.1007/978-3-642-22792-9_5

[16] Ethan Bernstein and Umesh V. Vazirani. 1997. Quantum Complexity Theory. *SIAM J. Comput.* 26, 5 (1997), 1411–1473.

[17] Nina Bindel, Jacqueline Brendel, Marc Fischlin, Brian Goncalves, and Douglas Stebila. 2019. Hybrid Key Encapsulation Mechanisms and Authenticated Key Exchange. In *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019, Chongqing, China, May 8-10, 2019 Revised Selected Papers (Lecture Notes in Computer Science, Vol. 11505)*, Jintai Ding and Rainer Steinwandt (Eds.). Springer, 206–226. doi:10.1007/978-3-030-25510-7_12

[18] Bruno Blanchet. 2006. A Computationally Sound Mechanized Prover for Security Protocols. In *IEEE Symposium on Security and Privacy*. Oakland, California, 140–154.

[19] Bruno Blanchet and Charlie Jacomme. 2024. Post-Quantum Sound CryptoVerif and Verification of Hybrid TLS and SSH Key-Exchanges. In *37th IEEE Computer Security Foundations Symposium, CSF 2024, Enschede, Netherlands, July 8-12, 2024*. IEEE, 543–556. doi:10.1109/CSF61375.2024.00050

[20] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. 2011. Random Oracles in a Quantum World. In *ASIACRYPT (Lecture Notes in Computer Science, Vol. 7073)*. Springer, 41–69.

[21] Colin Boyd, Yvonne Cliff, Juan Manuel González Nieto, and Kenneth G. Paterson. 2009. One-round key exchange in the standard model. *Int. J. Appl. Cryptogr.* 1, 3 (2009), 181–199. doi:10.1504/IJACT.2009.023466

[22] Graeme Connell and Rolfe Schmidt. October 2025. Signal Protocol and Post-Quantum Ratchets. https://signal.org/blog/spqr/.

[23] Ronald Cramer and Victor Shoup. 2003. Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. *SIAM J. Comput.* 33, 1 (2003), 167–226.

[24] Cas Cremers, Caroline Fontaine, and Charlie Jacomme. 2022. A Logic and an Interactive Prover for the Computational Post-Quantum Security of Protocols. In *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*. IEEE, 125–141.

[25] David Deutsch. 1985. Quantum theory, the Church–Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 400 (1985), 117 – 97. https://api.semanticscholar.org/CorpusID:1438116

[26] DGBJ Dieks. 1982. Communication by EPR devices. *Physics Letters A* 92, 6 (1982), 271–272.

[27] Federico Giacon, Felix Heuer, and Bertram Poettering. 2018. KEM combiners. In *IACR International Workshop on Public Key Cryptography*. Springer, 190–218.

[28] Hugo Krawczyk. 2003. SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE-Protocols. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings (Lecture Notes in Computer Science, Vol. 2729)*, Dan Boneh (Ed.). Springer, 400–425. doi:10.1007/978-3-540-45146-4_24

[29] Felix Linker, Ralf Sasse, and David Basin. 2025. A Formal Analysis of Apple's iMessage PQ3 Protocol. In *USENIX Security Symposium*. USENIX Association.

[30] National Institute of Standards and Technology. 2024. Module-Lattice-Based Digital Signature Standard. https://csrc.nist.gov/pubs/fips/204/final.

[31] National Institute of Standards and Technology. 2024. Module-Lattice-Based Key-Encapsulation Mechanism Standard. https://csrc.nist.gov/pubs/fips/203/final.

[32] Signal. [n. d.]. The PQXDH Key Agreement Protocol. https://signal.org/docs/specifications/pqxdh/.

[33] Dominique Unruh. 2019. Quantum relational Hoare logic. *Proc. ACM Program. Lang.* 3, POPL (2019), 33:1–33:31.
[34] Dominique Unruh. 2020. Post-Quantum Verification of Fujisaki-Okamoto. In *ASIACRYPT (1) (Lecture Notes in Computer Science, Vol. 12491)*. Springer, 321–352.
[35] William K Wootters and Wojciech H Zurek. 1982. A single quantum cannot be cloned. *Nature* 299, 5886 (1982), 802–803.

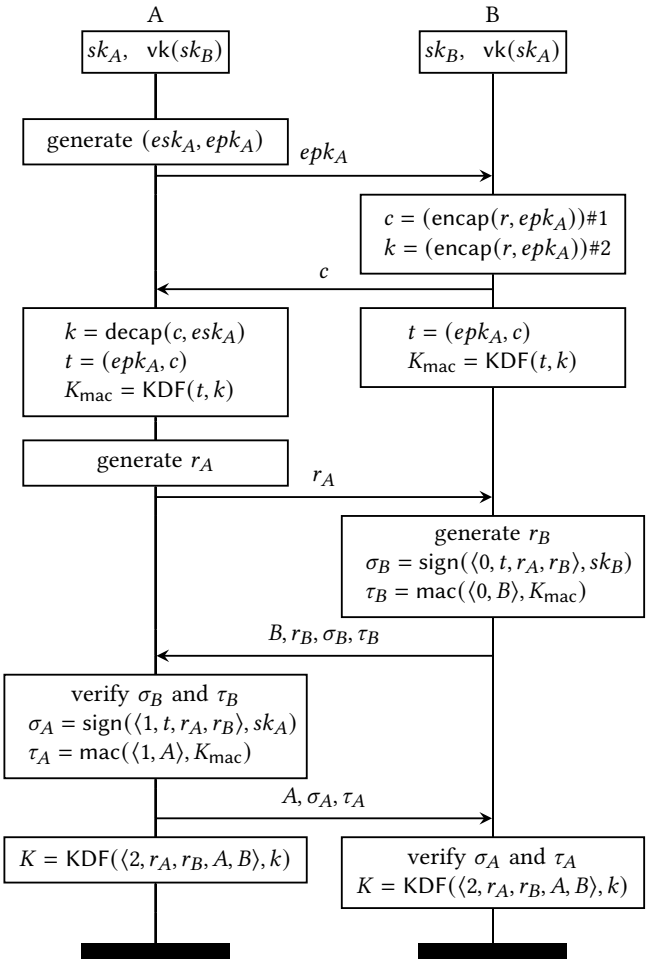


Figure 6: Description of the $C_{\mathtt{SigMA}}$ protocol

Outline. We begin with the mandatory Open Science section. Next, Appendix B provides some background before we formalize our tailored execution model for quantum computation and introduce our novel quantum simulation predicate in Appendix C. In Appendix D, we establish the adequacy of our approximate semantics with respect to its exact counterpart. Finally, Appendix E presents the proof system underlying the quantum extension of the Squirrel prover and proves its soundness.

A Open Science

This paper provides an extension of Squirrel, along with some case studies. The updated Squirrel code and our case studies are provided as supplementary material in an anonymous git repository: https://anonymous.4open.science/r/squirrel-prover-pq-0F42/.

B Some Background

In this section, we recall key aspects of the higher-order probabilistic logic introduced in [10], along with some useful notions from probability theory and quantum computing. We also present a result (Proposition 3) that will be instrumental in relating the exact semantics to its approximate version.

B.1 Details on Higher-Order CCSA Logic

Our approach relies on an instance of the general higher-order probabilistic logic of [10]. We keep the special symbols for names, boolean connectives, and classical cryptography, and the corresponding semantic assumptions. In addition, we consider special symbols for representing quantum computations and reasoning about them, whose semantics is defined in the body of the paper. We recall below some definitions from [10] for convenience.

Definition 1. *Let t be a well-typed term, i.e. $\mathcal{E} \vdash t : \tau$. Let $\mathbb{M}$ be a model wrt. $\mathcal{E}$. We define the interpretation $[\![t]\!]^{\eta,\rho}_{\mathbb{M}} \in \mathbb{RV}_{\mathbb{M}}(\tau)$ as follows:*

$$\begin{aligned} [\![s]\!]^{\eta,\rho}_{\mathbb{M}} &= \mathbb{M}(s)(\eta,\rho) \\ [\![t\ t']\!]^{\eta,\rho}_{\mathbb{M}} &= [\![t]\!]^{\eta,\rho}_{\mathbb{M}}([\![t']\!]^{\eta,\rho}_{\mathbb{M}}) \\ [\![\lambda(x:\tau').t]\!]^{\eta,\rho}_{\mathbb{M}} &= (a \in [\![\tau']\!]^{\eta}_{\mathbb{M}} \mapsto [\![t]\!]^{\eta,\rho}_{\mathbb{M}[x/\mathbb{1}^{\eta}_{a}]}) \end{aligned}$$

where the Dirac $\mathbb{1}^{\eta}_{a} \in \mathbb{RV}_{\mathbb{M}}(\tau')$ is such that $\mathbb{1}^{\eta}_{a}(\eta,\rho) = a$ for all ρ and takes irrelevant values elsewhere.

The full syntax for (global) formulas is given below, using annotated versions of logical connectives to distinguish them from their counterparts used in local formulas:

$$\begin{aligned} F \ ::= \ & F_1 \mathbin{\tilde{\wedge}} F_2 \mid F_1 \mathbin{\tilde{\vee}} F_2 \mid F_1 \mathbin{\tilde{\Rightarrow}} F_2 \mid \tilde{\neg} F \mid \tilde{\forall}(x:\tau).\, F \mid \tilde{\exists}(x:\tau).\, F \\ & \mid [\phi] \mid [\phi]_{\mathsf{e}} \mid \mathsf{const}(t) \mid \mathsf{adv}(t) \mid \vec{u} \sim \vec{v} \end{aligned}$$

The satisfaction relation is defined as usual in first-order logic, and the standard notion of validity (wrt. a class of models) follows. For instance, we have:

$$\mathbb{M} \models \tilde{\forall}(x:\tau).\, F \text{ iff. } \mathbb{M}[x/V] \models F \text{ for all } V \in \mathbb{RV}_{\mathbb{M}}(\tau)$$

B.2 Probability Theory

We recall some useful notions on measure theory, finite distributions and statistical distance.

Measure theory. For any set $\mathbb{S}$, we let $\mathcal{P}(\mathbb{S})$ be the *power-set* of $\mathbb{S}$. A *measurable space* $(\mathbb{S}, \mathcal{F})$ is a set $\mathbb{S}$ equipped with a σ-algebra $\mathcal{F} \subseteq \mathcal{P}(\mathbb{S})$, i.e. $\mathcal{F}$ must be: non-empty, closed under complement, and closed under countable intersection and union. The σ-algebra *generated* by a set $\mathcal{B} \subseteq \mathcal{P}(\mathbb{S})$, which we write $\sigma(\mathcal{B})$, is the smallest σ-algebra over $\mathbb{S}$ containing $\mathcal{B}$. A *measure space* $(\mathbb{S}, \mathcal{F}, \mu)$ is a measurable space $(\mathbb{S}, \mathcal{F})$ together with a measure $\mu : \mathcal{F} \to \mathbb{R}$, which is a function such that: i) $\mu(\emptyset) = 0$; ii) $\mu(E) \geq 0$ for any $E \in \mathcal{F}$; and iii), μ is σ-additive, i.e. for any countable disjoint family $(E_i)_i$ of

events in $\mathcal{F}$, $\mu(\cup_i E_i) = \sum_i \mu(E_i)$. A *probabilistic space* $(\mathbb{S}, \mathcal{F}, \mu)$ is a measure space of total mass one, i.e. such that $\mu(\mathbb{S}) = 1$. For any measurable set $(\mathbb{S}, \mathcal{F})$, we let $\mathrm{Distr}(\mathbb{S})$ be the set of all *distributions* over $\mathbb{S}$, i.e. the set of measure μ such that $(\mathbb{S}, \mathcal{F}, \mu)$ is a probabilistic space.

A *random variable* is a function $X : \Omega \to \mathbb{S}$ from a measure space $(\Omega, \mathcal{F}_\Omega, \mu_\Omega)$ to a measurable space $(\mathbb{S}, \mathcal{F}_\mathbb{S})$ such that $X^{-1}(E) \in \mathcal{F}_\Omega$ for any $E \in \mathcal{F}_\mathbb{S}$. This random variable induces the *pushforward measure* $\mu_\mathbb{S} : \mathcal{F}_\mathbb{S} \mapsto \mathbb{R}$ over $\mathbb{S}$ defined by $\mu_\mathbb{S}(E) \stackrel{\text{def}}{=} \mu_\Omega(X^{-1}(E))$ for any $E \in \mathcal{F}_\mathbb{S}$. Remark that the induced measure $\mu_\mathbb{S}$ is a distribution over $\mathbb{S}$ whenever $(\Omega, \mathcal{F}_\Omega, \mu_\Omega)$ is a probabilistic space.

Discrete distributions. A *discrete distribution* μ over $(\mathbb{S}, \mathcal{F})$ is a distribution for which there exists an at-most countable set $\mathbb{I} \in \mathcal{F}$ such that $\mu(\mathbb{I}) = 1$. A discrete distribution μ is uniquely characterized by the *discrete probability mass* function $\Delta : \mathbb{S} \to [0, 1]$ defined by $\Delta(a) = \mu(\{a\})$ for any $a \in \mathbb{S}$. By notation abuse, we sometimes confuse a discrete distribution μ and its corresponding probability mass function Δ. The *support* of a discrete distribution is $\mathrm{supp}(\mu) \stackrel{\text{def}}{=} \{x \in \mathbb{S} \mid \mu(\{x\}) > 0\}$.

Statistical distance. For any measurable space $(\mathbb{A}, \mathcal{F}_\mathbb{A})$ and distributions μ, μ' over $\mathbb{A}$, we let $\mathrm{d_{stat}}$ be the *total variation distance* (also called *statistical distance*) between μ and μ':

$$\mathrm{d_{stat}}(\mu, \mu') = \sup_{E \in \mathcal{F}_\mathbb{A}} \left|\mu(E) - \mu'(E)\right|.$$

Moreover, if μ, μ' are discrete distributions of support included in a discrete set $\mathbb{S} \subseteq \mathbb{A}$ and characterized by the discrete probability mass function Δ, Δ', then it is well-known that:

$$\mathrm{d_{stat}}(\mu, \mu') = \frac{1}{2} \sum_{a \in \mathbb{S}} \left|\Delta(a) - \Delta'(a)\right|.$$

We lift the notion of statistical distance to random variables through their induced distributions: if $X_1 : \Omega_1 \to \mathbb{A}$ and $X_2 : \Omega_2 \to \mathbb{A}$ are two random variables from the probability spaces, resp., $(\Omega_1, \mathcal{F}_1, \mu_1)$ and $(\Omega_2, \mathcal{F}_2, \mu_2)$, then $\mathrm{d_{stat}}(X_1, X_2) \stackrel{\text{def}}{=} \mathrm{d_{stat}}(\mu_1, \mu_2)$ where μ_1 and μ_2 are the distributions induced over $\mathbb{A}$ by, resp., X_1 and X_2. We define similarly the statistical distance $\mathrm{d_{stat}}(\mu, X)$ between a distribution μ and a random variable X.

B.3 Primer on Quantum Computing

For any set of classical values S, for instance the set of bitstrings, we can define its corresponding quantum domain $\mathcal{H}_S$ built over the orthonormal base $\{|x\rangle \mid x \in S\}$ (sometimes called the computational basis). That is, any element u in $\mathcal{H}_S$ can be written as:

$$u = \textstyle\sum_{x \in S} \alpha_x |x\rangle \quad \text{with } \alpha_x \in \mathbb{C}.$$

We write $\|u\| = \sqrt{\sum_{x \in S} |\alpha_x|^2}$ the norm of a vector u, and say that u is normalized if $\|u\| = 1$.

Physically, the quantum state represented by u is the superposition of all the possible x. If measured, u collapses to some x with probability $\frac{|\alpha_x|^2}{\|u\|^2}$.

Remark 2. *Two vectors u and v represent the same physical quantum state iff. $u = \lambda\, e^{i\theta}\, v$ where $\lambda, \theta \in \mathbb{R}^*$.*

Tensor product. Let $\mathbb{S}_1, \mathbb{S}_2$ be two Hilbert spaces where $\mathbb{S}_i$ has computational basis $\{|x\rangle \mid x \in \mathbb{A}_i\}$ for any $i \in \{1, 2\}$. We let $\mathbb{S}_1 \otimes \mathbb{S}_2$ denote the *tensor product* of $\mathbb{S}_1$ and $\mathbb{S}_2$, which is the Hilbert space with orthonormal basis $\{|x, y\rangle \mid x \in \mathbb{A}_1, y \in \mathbb{A}_2\}$. Let $\mathbf{a} = \sum_{x \in \mathbb{A}_1} a_x |x\rangle$ and $\mathbf{b} = \sum_{y \in \mathbb{A}_2} b_y |y\rangle$ be two elements of, resp., $\mathbb{S}_1$ and $\mathbb{S}_2$. Then we let $\mathbf{a} \otimes \mathbf{b}$ be the element of $\mathbb{S}_1 \otimes \mathbb{S}_2$ defined by: $\sum_{x \in \mathbb{A}_1, y \in \mathbb{A}_2} a_x \cdot b_y \cdot |x, y\rangle$.

Partial measurement. Let $\mathbb{S}_1, \mathbb{S}_2$ be two Hilbert spaces where $\mathbb{S}_i$ has computational basis $\{|x\rangle \mid x \in \mathbb{A}_i\}$ for any $i \in \{1, 2\}$. The partial measurement $\mu : (\mathbb{S}_1 \otimes \mathbb{S}_2) \to \mathrm{Distr}(\mathbb{A}_1 \times \mathbb{S}_2)$ over sub-space $\mathbb{S}_1$ of a quantum state represented by the normalized vector:

$$u = \left(\textstyle\sum_{(x,y) \in \mathbb{A}_1 \times \mathbb{A}_2} \alpha_{x,y} |x, y\rangle\right) \in \mathbb{S}_1 \otimes \mathbb{S}_2$$

is a distribution over $\mathbb{A}_1 \times \mathbb{S}_2$ whose law is given by:

$$\mu(u)(a, \textstyle\sum_{y \in \mathbb{A}_2} \alpha_{a,y} |y\rangle) \stackrel{\text{def}}{=} \sum_{y \in \mathbb{A}_2} |\alpha_{a,y}|^2 \qquad \text{for any } a \in \mathbb{A}_1.$$

Our attackers will always perform partial measurement over their current quantum state, in order to obtain both a new quantum state, as well as a classical output value to send over the network.

B.4 Approximating Distributions

The semantics of our probabilistic logic requires that all distributions are computed using a finite number of random coins. Measuring a quantum state $a \in \mathcal{H}_{\{0,1\}^*}$ yields a discrete distribution Δ over bit-strings in $\{0, 1\}^*$. This distribution may not be *exactly* computable using a finite number of unbiased random coins. E.g. we can have $a \in \mathcal{H}_{\{0,1\}^*}$ such that $\Delta = \mu(a)$ yields 0 with probability $\frac{1}{\sqrt{2}}$ and 1 otherwise.

Therefore, instead of using the discrete distribution $\Delta : \mathbb{A} \to [0, 1]$ corresponding to a quantum measurement of a when defining the semantics of the logic, we *approximate* Δ using a random variable $\mathrm{approx_r}(\Delta) : \{0, 1\}^{2 \cdot \mathsf{r}} \to \mathbb{A}$ which is defined using only a finite number of random bits $2 \cdot \mathsf{r}$. This definition is split in two steps, where a first approximation $\mathrm{approx}^0_\mathsf{r}(\Delta)$ using the first r random bits is modified by a second approximation step using the remaining r random bits to obtain $\mathrm{approx_r}(\Delta)$. This second step tries to ensure that Δ and $\mathrm{approx_r}(\Delta)$ have the same support, which is a useful property of our approximation operator (see Remark 1). Then, we bound the statistical distance between the distribution Δ and (the distribution induced by) the random variable $\mathrm{approx_r}(\Delta)$ as a function of the precision r.

B.4.1 First approximation $\mathrm{approx}^0_\mathsf{r}(\cdot)$. Let $(\mathbb{A}, \mathcal{F}_\mathbb{A})$ be a measurable set, $\Delta : \mathbb{A} \to [0, 1]$ a *discrete distribution* over $\mathbb{A}$, and let $\mathsf{r} \in \mathbb{N}^* \cup \{\omega\}$ (where ω is the first limit ordinal). For any $s \in \mathbb{A}$, let $p_s, k_s \in [0, 1]$ be real numbers s.t. p_s is the real corresponding to the first $\mathsf{r} + 1$ bits of the binary writing of $\Delta(s)$, i.e. if $\sum_{i=0}^{+\infty} a_i \cdot 2^{-i}$ is the binary decomposition of $\Delta(s)$ then:

$$\Delta(s) = p_s + \frac{k_s}{2^\mathsf{r}} \text{ with } p_s = \sum_{i=0}^{r} a_i \cdot 2^{-i} \text{ and } \frac{k_s}{2^\mathsf{r}} = \sum_{i=r+1}^{+\infty} a_i \cdot 2^{-i}$$

with the convention that $p_s = \Delta(s)$ and $k_s = 0$ when $\mathsf{r} = \omega$.

We assume a total ordering $<$ of the elements of $\mathbb{A}$. We equip the set $\{0, 1\}^{2 \cdot \mathsf{r}}$ of sequences of r random bits with the uniform measure when $\mathsf{r} \in \mathbb{N}$, and the Lebesgue measure when $\mathsf{r} = \omega$ (by identifying $\{0, 1\}^\omega = \{0, 1\}^\mathbb{N}$ with the real interval $[0, 1]$). Then

$\mathsf{approx}^0_{\mathsf{r}}(\Delta) : \{0,1\}^{\mathsf{r}} \to \mathbb{A}$ is *the random variable approximating the distribution* Δ *using* $2 \cdot \mathsf{r}$ *random bits* defined as follows:

$$\forall w \in \{0,1\}^{\mathsf{r}}. \quad \mathsf{approx}^0_{\mathsf{r}}(\Delta)(w) \stackrel{\text{def}}{=} \begin{cases} s & \text{if } |0.w|_{\mathbb{R}} \in \left]\sum_{s'<s} p_{s'}; \sum_{s'\leq s} p_{s'}\right] \\ s_{\mathsf{default}} & \text{otherwise} \end{cases}$$

where $|0.w|_{\mathbb{R}} \stackrel{\text{def}}{=} \sum_{1\leq i\leq |w|} w_i \cdot 2^{-i}$ is the binary reading of $0.w$, and s_{default} is some arbitrary element of $\mathbb{A}$ in the support of Δ.

By the construction of this random variable, we have that:

$$\Pr_w\left(\mathsf{approx}^0_{\mathsf{r}}(\Delta)(w) = s\right) = \begin{cases} p_s \text{ if } s \neq s_{\mathsf{default}} \\ p_{s_{\mathsf{default}}} + \sum_{s_0 \in \mathbb{A}} \frac{k_{s_0}}{2^{\mathsf{r}}} \end{cases}.$$

Proposition 1. *If* Δ *is a discrete distribution, then:*

$$\mathsf{d}_{\mathsf{stat}}(\Delta, \mathsf{approx}^0_{\mathsf{r}}(\Delta)) \leq 2 \cdot \frac{|supp(\Delta)|}{2^{\mathsf{r}}}.$$

Proof. For any $s \in \mathbb{A}$, we have that:

$$\left|\Pr(\Delta = s) - \Pr(\mathsf{approx}^0_{\mathsf{r}}(\Delta) = s)\right| = \begin{cases} \frac{k_s}{2^{\mathsf{r}}} & \text{if } s \neq s_{\mathsf{default}} \\ \sum_{s_0 \in \mathbb{A}\setminus\{s_{\mathsf{default}}\}} \frac{k_{s_0}}{2^{\mathsf{r}}} & \text{if } s = s_{\mathsf{default}} \end{cases}$$

Hence, we have that:

$$\begin{aligned} & 2 \cdot \mathsf{d}_{\mathsf{stat}}(\Delta, \mathsf{approx}^0_{\mathsf{r}}(\Delta)) \\ = & \sum_{s\in\mathbb{A}} \left|\Pr(\Delta = s) - \Pr(\mathsf{approx}^0_{\mathsf{r}}(\Delta) = s)\right| \\ = & \sum_{s\in\mathbb{A}\setminus\{s_{\mathsf{default}}\}} \frac{k_s}{2^{\mathsf{r}}} + \sum_{s\in\mathbb{A}\setminus\{s_{\mathsf{default}}\}} \frac{k_s}{2^{\mathsf{r}}} \\ \leq & 2 \cdot \sum_{s\in\mathbb{A}} \frac{k_s}{2^{\mathsf{r}}}. \\ \leq & 2 \cdot \frac{|\mathsf{supp}(\Delta)|}{2^{\mathsf{r}}} \end{aligned}$$

□

B.4.2 Approximation $\mathsf{approx}_{\mathsf{r}}(\cdot)$. We try to modify $\mathsf{approx}^0_{\Delta}()$ by adding a small non-zero probability to all elements in the support of Δ. This is only possible if r is large enough: otherwise, there may be more element in $\mathsf{supp}(\Delta)$ than elements in the partition of $[0,1]$ that we can compute using the fixed number of random bits at our disposal. Concretely, if $|\mathsf{supp}(\Delta)| > 2^{\mathsf{r}}$, we let $\mathsf{approx}_{\mathsf{r}}(\Delta) = \mathsf{approx}^0_{\mathsf{r}}(\Delta)$. Otherwise, we know that $|\mathsf{supp}(\Delta)| \leq 2^{\mathsf{r}}$. Thus, there exists a distribution I that can be computed using r unbiased random bits such that I assigns a strictly positive probability to every element in $\mathsf{supp}(\Delta)$, and a zero probability to any other element:

$$\Pr(I = s) > 0 \text{ if } s \in \mathsf{supp}(\Delta) \qquad \Pr(I = s) = 0 \text{ otherwise}$$

(I can be easily constructed from a subjection from $\{1; \ldots; 2^{\mathsf{r}}\}$ to $\mathsf{supp}(\Delta)$.) Then, we let:

$$\mathsf{approx}_{\mathsf{r}}(\Delta) = (1 - \frac{1}{2^{\mathsf{r}}})\, \mathsf{approx}^0_{\mathsf{r}}(\Delta) + \frac{1}{2^{\mathsf{r}}} I$$

This distribution can be computed using r random bits, by using the first r bits to select the left or right distributions:

$$(1 - \frac{1}{2^{\mathsf{r}}})\, \mathsf{approx}^0_{\mathsf{r}}(\Delta) \quad \text{or} \quad \frac{1}{2^{\mathsf{r}}} I$$

and the following r bits to compute $\mathsf{approx}^0_{\mathsf{r}}(\Delta)$ or I.

Proposition 2. *If* Δ *is a discrete distribution, then:*

$$\mathsf{d}_{\mathsf{stat}}\left(\mathsf{approx}^0_{\mathsf{r}}(\Delta), \mathsf{approx}_{\mathsf{r}}(\Delta)\right) \leq \frac{|supp(\Delta)|}{2^{\mathsf{r}}}.$$

Proof. If $|\mathsf{supp}(\Delta)| > 2^{\mathsf{r}}$, this is obvious since this distance is 0. Otherwise, let $s \in \mathsf{supp}(\Delta)$ and

$$p = \Pr\left(\mathsf{approx}^0_{\mathsf{r}}(\Delta) = s\right) \quad \text{and} \quad q = \Pr(I = s).$$

Then:

$$\begin{aligned} & \left|\Pr(\mathsf{approx}^0_{\mathsf{r}}(\Delta) = s) - \Pr(\mathsf{approx}_{\mathsf{r}}(\Delta) = s)\right| \\ \leq & \left|p - \left((1 - \frac{1}{2^{\mathsf{r}}})\, p + \frac{q}{2^{\mathsf{r}}}\right)\right| \\ \leq & \frac{1}{2^{\mathsf{r}}}\, |p - q| \\ \leq & \frac{1}{2^{\mathsf{r}}} \qquad \text{(Since } p \leq 1 \text{ and } q \leq 1) \end{aligned}$$

If $s \notin \mathsf{supp}(\Delta)$, then:

$$\left|\Pr(\mathsf{approx}^0_{\mathsf{r}}(\Delta) = s) - \Pr(\mathsf{approx}_{\mathsf{r}}(\Delta) = s)\right| = 0$$

Consequently:

$$\begin{aligned} & 2 \cdot \mathsf{d}_{\mathsf{stat}}(\mathsf{approx}^0_{\mathsf{r}}(\Delta), \mathsf{approx}_{\mathsf{r}}(\Delta)) \\ = & \sum_s \left|\Pr(\mathsf{approx}^0_{\mathsf{r}}(\Delta) = s) - \Pr(\mathsf{approx}_{\mathsf{r}}(\Delta) = s)\right| \\ \leq & \sum_{s\in\mathsf{supp}(\Delta)} \frac{1}{2^{\mathsf{r}}}. \\ \leq & \frac{|\mathsf{supp}(\Delta)|}{2^{\mathsf{r}}} \end{aligned}$$

□

Proposition 3. *If* Δ *is a discrete distribution, then the statistical distance between* Δ *and the distribution induced by* $\mathsf{approx}^0_{\mathsf{r}}(\Delta)$ *can be upper-bounded as follows:*

$$\mathsf{d}_{\mathsf{stat}}(\Delta, \mathsf{approx}^0_{\mathsf{r}}(\Delta)) \leq 3 \cdot \frac{|supp(\Delta)|}{2^{\mathsf{r}}} \leq \frac{|supp(\Delta)|}{2^{\mathsf{r}-2}}$$

Proof. We immediately obtain the wanted result using the triangular inequality, Proposition 1, and Proposition 2. □

B.4.3 Notation. If D is a function from a set $\mathbb{S}_1$ to *distributions* over $\mathbb{S}_2$, i.e. $D : \mathbb{S}_1 \to \mathsf{Distr}(\mathbb{S}_2)$ then we will abuse notation and let, for any $r \in \mathbb{N}^* \cup \{\omega\}$, $a \in \mathbb{S}_1$ and $w \in \{0,1\}^r$:

$$\mathsf{approx}_r(D)(a, w) \stackrel{\text{def}}{=} \mathsf{approx}_r(D(a))(w).$$

If $r = \omega$ then there is no approximation error between Δ and the distribution induced by $\mathsf{approx}_{\mathsf{r}}(\Delta)$, and we write:

$$\mathsf{exact}(D)(a, w) \stackrel{\text{def}}{=} \mathsf{approx}_{\omega}(D)(a, w).$$

C Quantum Computation Execution Model

In this section, we formalize our realistic execution model for quantum computation, which we use to model quantum attacker. As usual in cryptography, we let the attacker have access to oracles, which we model as order one terms. A quantum computation may access an oracle in two different ways, depending on how it can query the oracle. A *classical-access* oracle can only be queried on classical, non-quantum, values. For example, such oracles can be

used to model the interactions of a quantum adversary with a protocol party on the network. By contrast, a *quantum-access* oracle h may be queried on a super-position of inputs $\sum_x \alpha_x|x\rangle$, and answers with the super-positions of outputs $\sum_x \alpha_x|h(x)\rangle$ (see Appendix B.3 below for a quick primer on quantum computing). Such oracles represents sub-procedures of a quantum machines, and allow to model the QROM. To allow both classical-access and quantum-access oracles, we will use hybrid classical-quantum machines, whose executions are composed of interleavings of classical and quantum computations with potentially many quantum measurements within this interleaving. The former can query the classical-access oracles, while the latter can query the quantum-access oracles. Such hybrid classical/quantum computations should faithfully model the real-world capabilities of a post-quantum attackers, which would rely on a combination of a classical and a quantum computer.

Outline. We start with a description of how we interface the abstract types of the logic and the bitstring values manipulated by quantum Turing machine through encoding/decoding in Appendix C.1. Then, Appendix C.2 present our novel quantum execution model. Finally, we recall some well-known quantum simulation of classical computations results in Appendix C.3, and we define quantum simulation predicate in Appendix C.4.

C.1 Type Encodings

The interpretation domain of a base type is required to be bit-string encodable. consequently, for every $\tau_b \in \mathbb{B}$, we let $\mathsf{encode}^{\tau_b}_{\eta,\mathbb{M}} : [\![\tau_b]\!]^\eta_{\mathbb{M}} \mapsto \{0,1\}^*$ be an injective encoding of $[\![\tau_b]\!]^\eta_{\mathbb{M}}$ as bit-strings, and $\mathsf{decode}^{\tau_b}_{\eta,\mathbb{M}} : \{0,1\}^* \mapsto [\![\tau_b]\!]^\eta_{\mathbb{M}}$ a corresponding decoding function (i.e. we must have that $\mathsf{decode}^{\tau_b}_\eta(\mathsf{encode}^{\tau_b}_\eta(a)) = a$ for any $a \in [\![\tau_b]\!]^\eta_{\mathbb{M}}$).

We usually omit the model $\mathbb{M}$ when it is clear from context, e.g. we may write $\mathsf{encode}^{\tau_b}_\eta$ instead of $\mathsf{encode}^{\tau_b}_{\eta,\mathbb{M}}$. Moreover, for any types τ, τ' for which $\mathsf{encode}^{\tau}_\eta$ and $\mathsf{encode}^{\tau'}_\eta$ are defined, we lift the encoding to $\tau \to \tau'$ by having

$$\mathsf{encode}^{\tau\to\tau'}_\eta : [\![\tau \to \tau']\!]^\eta_{\mathbb{M}} \to \{0,1\}^* \to \{0,1\}^*$$

be such that for any $a \in [\![\tau \to \tau']\!]^\eta_{\mathbb{M}}$, $\mathsf{encode}^{\tau\to\tau'}_\eta(a) = \mathsf{encode}^{\tau'}_\eta \circ a \circ \mathsf{decode}^{\tau}_\eta$ where $\circ$ denotes function composition.

We extend encode_η to any quantum type $\tau \stackrel{\text{def}}{=} \mathsf{Hilb}_{\tau_b}$ by having the encoding $\mathsf{encode}^{\tau}_\eta$ of a quantum value $u \stackrel{\text{def}}{=} \sum_{x\in[\![\tau_b]\!]^\eta_{\mathbb{M}}} \alpha_x|x\rangle$ be the superposition $\sum_{x\in[\![\tau_b]\!]^\eta_{\mathbb{M}}} \alpha_x|\mathsf{encode}^{\tau_b}_\eta(x)\rangle$.

Finally, we let $\mathsf{Hilb}_{\mathbb{B}}$ be the set of all quantum base types of the logic:

$$\mathsf{Hilb}_{\mathbb{B}} \stackrel{\text{def}}{=} \{\mathsf{Hilb}_{\tau_b} \mid \tau_b \in \mathbb{B}\}.$$

For any type $\tau \in \mathbb{B} \cup \mathsf{Hilb}_{\mathbb{B}}$, we let $\mathsf{qlift}(\tau)$ be the lifting of τ as a quantum type, if necessary, i.e.:

$$\mathsf{qlift}(\tau) \stackrel{\text{def}}{=} \begin{cases} \tau & \text{if } \tau \in \mathsf{Hilb}_{\mathbb{B}} \\ \mathsf{Hilb}_\tau & \text{if } \tau \in \mathbb{B} \end{cases}.$$

C.2 QTM Execution

We now describe our execution model for *Quantum Turing Machine* (QTM). We split the quantum execution of an machine on some input w in two: first, the **computation phase**, which runs, *without measurement*, the quantum machine $\mathcal{M}$ on w to obtain a pure quantum state $\mathcal{M}(w)$; then, the **measurement phase** computes the partial measurement $\mu(\mathcal{M}(w))$ of the output, which split the output of $\mathcal{M}$ into a classical output and a quantum output. Having both kinds of output is useful: e.g., to model an interactive quantum protocol, the classical output will be forwarded to some honest protocol agent, while the quantum output will serve as a persistent state, which will be passed back to the quantum adversary $\mathcal{M}$ each times it runs.

Our execution model supports quantum *oracle* machine. Essentially, a machine Q can call any oracle it receives as input by writing the input query on a dedicated tape and then going into the query state of the oracle. It then goes, in one step, to a new state with the output of the oracle written on the dedicated oracle output tape. This output can itself be a superposed state, when the oracle is a quantum-access one. We detail this mechanism below (§Appendix C.2.2).

Outline. We explain how we model QTM and their configurations in Appendix C.2.1. Then, we present the evaluation phase of our execution model in Appendix C.2.2 and the measurement phase in Appendix C.2.3. Finally, we discuss aspects related to complexity in Appendix C.2.4.

C.2.1 *Tapes and configurations.*

We rely on a standard presentation of quantum oracle machines, where each machine $\mathcal{M}$ is equipped with a finite set of inputs and output tapes as well as a pair of query and result tapes for each oracle the machine has access to. We do not describe in details the shape of a configuration C of our machine $\mathcal{M}$. During its execution, the state of the quantum machine $\mathcal{M}$ cannot be (usually) represented by a single configuration, but rather by a superpositions of configurations $\sum_x \alpha_x|C_x\rangle$. Each step of $\mathcal{M}$ execution modifies this state by applying an unitary transformation $U_\mathcal{M}$ (called the *time evolution operator* of the machine $\mathcal{M}$) to it. As $U_\mathcal{M}$ is a linear operator:

$$U_\mathcal{M}(\textstyle\sum_x \alpha_x \cdot |C_x\rangle) = \sum_x \alpha_x \cdot U_\mathcal{M}(|C_x\rangle)$$

and thus $U_\mathcal{M}$ can be fully characterized by describing its behavior on each vector of the computational basis $\{|C\rangle \mid C \text{ a configuration}\}$.

Quantum machine operates on bit-strings while the logic operates on typed values. To ease the link with logic, we are going to type the tapes of our quantum machines using base types $\mathbb{B}$, and we implicitly consider that logical values are encoded or decoded to or from bitstring when passed or received to quantum machines. We can type a quantum machine $\mathcal{M}$ with k order 0 inputs of types $\tau^i_1, \ldots, \tau^i_k$, a number l of order 1 inputs (i.e. oracles) of types $\tau^{i,\mathsf{l}}_1 \to \tau^{i,\mathsf{r}}_1, \ldots, \tau^{i,\mathsf{l}}_l \to \tau^{i,\mathsf{r}}_l$; and m output types $\tau^o_1, \ldots, \tau^o_m$, where:

- every input type τ^i_j and oracle result type $\tau^{i,\mathsf{r}}_j$ is an order 0 type in $\mathbb{B} \cup \mathsf{Hilb}_{\mathbb{B}}$, and every oracle query type $\tau^{i,\mathsf{l}}_j$ is an order 0 type $\mathbb{B}$ (i.e. oracle inputs must be classical).

- every output type τ_j^o is an order 0 type in $\mathbb{B} \cup \mathsf{Hilb}_{\mathbb{B}}$, and there is at most one quantum output type, i.e. $|\{\tau_j^o \mid j \leq m\} \cap \mathsf{Hilb}_{\mathbb{B}}| \leq 1$ (we discuss this restriction later).

A configuration for a classical machine can be described as a value inside the product of the domains of the tapes, the head positions and the states. The domain $\mathcal{S}_{\mathcal{M}}$ of configurations of the QTM $\mathcal{M}$ is simply the quantum lifting of this, that is, the tensor product of the Hilbert space associated to the corresponding domains.

C.2.2 Evaluation Phase. A QTM $\mathcal{M}$ of type:

$$(\tau_1^i \star \cdots \star \tau_k^i \star (\tau_1^{i,\mathsf{l}} \to \tau_1^{i,\mathsf{r}}) \star \cdots \star (\tau_l^{i,\mathsf{l}} \to \tau_l^{i,\mathsf{r}})) \to (\tau_1^o \star \cdots \star \tau_m^o)$$

is a machine with k input tapes, one for each order 0 input; one query and one result tape for each of the l order 1 inputs (which yields $2 \cdot l$ oracle tapes); and m output tapes. Moreover, for each $1 \leq j \leq l$, we require that $\mathcal{M}$ has a special oracle query state q_j^l and a corresponding result state q_j^r. The execution of $\mathcal{M}$ w.r.t. the type structure $\mathbb{M}$ (which defines the interpretation of the logic's types) on order 0 inputs

$$(a_1, \ldots, a_k) \in [\![\tau_1^i]\!]_{\mathbb{M}}^{\eta} \times \cdots \times [\![\tau_k^i]\!]_{\mathbb{M}}^{\eta}$$

and order 1 (i.e. oracles) inputs

$$(u_1, \ldots, u_l) \in [\![\tau_1^{i,\mathsf{l}} \to \tau_1^{i,\mathsf{r}}]\!]_{\mathbb{M}}^{\eta} \times \cdots \times [\![\tau_l^{i,\mathsf{l}} \to \tau_l^{i,\mathsf{r}}]\!]_{\mathbb{M}}^{\eta}$$

goes as follows:

- each order 0 input a_j is encoded as a bit-string by encode_{η}; and the inputs $a_1, \ldots, a_k$ mixing classical and non-entangled quantum states are lifted to a single quantum value a in $[\![\mathsf{qlift}(\tau_1^i)]\!]_{\mathbb{M}}^{\eta} \otimes \cdots \otimes [\![\mathsf{qlift}(\tau_k^i)]\!]_{\mathbb{M}}^{\eta}$ as follows:

$$a \stackrel{\text{def}}{=} a'_1 \otimes \cdots \otimes a'_k \quad \text{where } \forall j.\ a'_j = \begin{cases} a_j & \text{if } \tau_j^i \in \mathsf{Hilb}_{\mathbb{B}} \\ |a_j\rangle & \text{if } \tau_j^i \in \mathbb{B} \end{cases}.$$

- then, $\mathcal{M}$ is executed starting from the initial quantum state a on the input tapes and the empty bit-string ϵ on each oracle or output tape. More precisely, the initial value of all of $\mathcal{M}$'s tapes (seen as a quantum tuple value) is:

$$a \otimes \bigotimes_{1 \leq j \leq l} (\epsilon \otimes \epsilon) \otimes \bigotimes_{1 \leq j \leq m} \epsilon$$

- $\mathcal{M}$ starts its execution, modifying its state according to its time evolution operator. In addition to the usual transitions of the QTM, its execution is extended with special transitions for each oracle. More precisely, for any $1 \leq j \leq l$, the machine $\mathcal{M}$ can call the j-th oracle u_j whenever it is in a configuration in the state q_j^l as described next. Let x and y be the values on the j-th oracle query and result tapes, respectively. When u_j returns a quantum value (i.e. the return type $\tau_j^{i,\mathsf{r}}$ of u_j is in $\mathsf{Hilb}_{\mathbb{B}}$), if $u_j(x) = \sum_v \alpha_v |y_v\rangle$, then the machine has a set of transitions, each going with amplitude α_v inside the same configuration but in state q_j^r and with $y + y_v$ on the oracle result tape (the value x on the oracle query tape is left unchanged). When u_j returns a classical value (i.e. if $\tau_j^{i,\mathsf{r}} \in \mathbb{B}$), then there is a single transition going inside the same configuration but in state q_j^r and with $y + u_j(x)$ on the oracle result tape.
- at the end of its execution, $\mathcal{M}$'s tapes yield a quantum value, denoted $\mathcal{M}(a_1, \ldots, a_k, u_1, \ldots, u_l)$, which after each input, output, query or result tape has been decoded from bits and qbits using the corresponding decode_{η}, is a value in the state space $\mathcal{S}_{\mathcal{M}}$ of $\mathcal{M}$:

$$\mathcal{S}_{\mathcal{M}} \stackrel{\text{def}}{=} \bigotimes_{1 \leq j \leq k} [\![\mathsf{qlift}(\tau_j^i)]\!]_{\mathbb{M}}^{\eta} \otimes \bigotimes_{1 \leq j \leq l} ([\![\mathsf{qlift}(\tau_j^{i,\mathsf{l}})]\!]_{\mathbb{M}}^{\eta} \otimes [\![\mathsf{qlift}(\tau_j^{i,\mathsf{r}})]\!]_{\mathbb{M}}^{\eta}) \otimes \bigotimes_{1 \leq j \leq m} [\![\mathsf{qlift}(\tau_j^o)]\!]_{\mathbb{M}}^{\eta}$$

In summary, $\mathcal{M}$ execution w.r.t. security parameter η and type structure $\mathbb{M}$ is a function in:

$$([\![\tau_1^i]\!]_{\mathbb{M}}^{\eta} \times \cdots \times [\![\tau_k^i]\!]_{\mathbb{M}}^{\eta} \times [\![\tau_1^{i,\mathsf{l}} \to \tau_1^{i,\mathsf{r}}]\!]_{\mathbb{M}}^{\eta} \times \cdots \times [\![\tau_l^{i,\mathsf{l}} \to \tau_l^{i,\mathsf{r}}]\!]_{\mathbb{M}}^{\eta}) \to \mathcal{S}_{\mathcal{M}}$$

Note that $\mathcal{M}$ takes as input k classical and non-entangled quantum values, l oracle, and outputs a single entangled quantum state over the tensor products of the $k + (2 \times l) + m$ input, oracle query, oracle result, and output types. The measurement phase, described next, is tasked with measuring $\mathcal{M}$'s output to split it into several classical components and at most a single quantum state.

C.2.3 Measurement phase. Finally, the *final* output of $\mathcal{M}$ is obtained by:

- performing the quantum partial measurement $\mu(\mathcal{M}(a, u))$ which measure the sub-spaces of $\mathcal{M}(a, u)$ corresponding to: the input tapes; the query and result tapes; the *classical* output tapes. Said otherwise, all of $\mathcal{M}(a, u)$'s output is measured, except possibly for the single quantum output.
- throwing away the content of the input tapes using the projection π, which leaves only the outputs.

Taken together, we have that $\pi \circ \mu$ — which we denote by $\mathsf{Measure}^{\mathcal{M}}_{\mathbb{M},\eta}$ — is a function of signature:

$$\mathsf{Measure}^{\mathcal{M}}_{\mathbb{M},\eta} : \mathcal{S}_{\mathcal{M}} \to \mathsf{Distr}([\![\tau_1^o]\!]_{\mathbb{M}}^{\eta} \times \cdots \times [\![\tau_m^o]\!]_{\mathbb{M}}^{\eta}).$$

Remark 3. *We described the execution and measurement phases of an QTM $\mathcal{M}$ where all order 0 inputs precede the order 1 inputs. We naturally lift it to any QTM $\mathcal{M}$ of type*

$$(\tau_1^i \star \cdots \star \tau_n^i) \to (\tau_1^o \star \cdots \star \tau_m^o)$$

where τ_j^i either of order 0 or 1 at any position. For any value $a \in [\![\tau_1^i \star \cdots \star \tau_n^i]\!]_{\mathbb{M}}^{\eta}$, we let $\mathcal{M}(a)$ be the result of executing $\mathcal{M}$ on a, without differentiating the order of the different parts of the input.

C.2.4 Complexity of a QTM. A QTM $\mathcal{M}$ is a *polynomial-time QTM* (PQTM) if $\mathcal{M}$ execution time is at-most polynomial in the length of its order 0 inputs — ignoring the time taken by the measurement and projection phase. Further, we make the following cost model assumptions:

- CM:AdvRO: The tape containing the adversarial randomness ρ_{a} is read-only.
- CM:OracleCost: A call to an oracle f on input x costs the length of the oracle output $f(x)$. That is, the transition $|x, y\rangle \mapsto |x, y + f(x)\rangle$ takes $|f(x)|$ steps.

We stress the fact that order 1 inputs do not participate in the computation of the length of $\mathcal{M}$'s inputs.

Proposition 4. *Let $\mathcal{M}$ be a QTM running in time at-most P. Then, for any order 0 inputs $\vec{a}$ and order 1 inputs $\vec{f}$, we have that at the end of $\mathcal{M}$'s execution on $(\vec{a}, \vec{f})$, the length of $\mathcal{M}(\vec{a}, \vec{f})$'s tapes is at-most $P(|\vec{a}|)$.*

Proof. This is straightforward thanks to our cost model assumption CM:AdvRO. □

C.3 Quantum Simulation of Reversible Classical Computations

All transitions are not valid transformation of the state of a quantum Turing machines: notably, valid transformations must be *unitary* transformation of the Hilbert-space. Fortunately, our approach is mostly agnostic in the precise way this restriction can be enforced: for our purposes, it will suffice to know that any reversible computable function can be implemented by a QTM.

C.3.1 Quantum simulation. It is well-known that any reversible polynomial-time classical function can be exactly simulated by a polynomial-time quantum machine.

Theorem 2 (Bernstein-Vazirani [16])**.** *We have the following two quantum simulation results:*

i) Let $f : \mathbb{A} \to \mathbb{B}$ be a polynomial-time computable function. Then:

$$\hat{f} : \left\{ \begin{array}{ccc} \mathcal{H}_{\mathbb{A}\times\mathbb{B}} & \to & \mathcal{H}_{\mathbb{A}\times\mathbb{B}} \\ \sum_{(x,y)} \alpha_{x,y} \cdot |x,y\rangle & \mapsto & \sum_{(x,y)} \alpha_{x,y} \cdot |x, y + f(x)\rangle \end{array} \right.$$

is computable by a polynomial-time quantum machine.

ii) Let $f : \mathbb{A} \to \mathbb{A}$ be an invertible function such that both f and f^{-1} are polynomial-time computable. Then:

$$\hat{f} : \left\{ \begin{array}{ccc} \mathcal{H}_{\mathbb{A}} & \to & \mathcal{H}_{\mathbb{A}} \\ \sum_{x} \alpha_{x} \cdot |x\rangle & \mapsto & \sum_{x} \alpha_{x} \cdot |f(x)\rangle \end{array} \right.$$

is computable by a polynomial-time quantum machine.

C.3.2 Error-free quantum computations. An error-free quantum machine is a QTM $\mathcal{M}$ whose output distribution is a Dirac. More precisely, $\mathcal{M}$ is an error-free QTM if, for any input w of $\mathcal{M}$, the distribution:

$$\mathsf{Measure}^{\mathcal{M}}_{\mathbb{M},\eta}(\mathcal{M}(w)) : \mathsf{Distr}(\llbracket \vec{\tau_o} \rrbracket^{\eta}_{\mathbb{M}})$$

is a Dirac $\mathbb{1}_{w'}$. In such a case, we will abuse notation and omit the final measurement and projection $\mathsf{Measure}^{\mathcal{M}}_{\mathbb{M},\eta}(\cdot)$, and directly write $\mathcal{M}(w) = w'$. We let $\mathrm{QTM}_{\mathsf{E}}$ stand for the set of error-free quantum machines, and $\mathrm{PQTM}_{\mathsf{E}}$ for the set of polynomial-time error-free quantum machines.

Example 8. *For example, the function* swap *such that*

$$\mathsf{swap}(|x,y\rangle) = |y,x\rangle$$

satisfies the conditions of Theorem 2.ii). Further, this function is error-free. Thus, there exists $\mathcal{M} \in PQTM_{\mathsf{E}}$ such that $\mathcal{M}(|x,y\rangle) = |y,x\rangle$.

C.4 Quantum Simulation Predicate

We now define our quantum simulation predicate for which we will provide a proof system in Appendix E.

C.4.1 Generalized quantum simulation predicate. First, we generalize the quantum simulation predicate $\mathsf{qadv}(\cdot;\cdot)$ presented in the body with an additional argument X to control the random tapes made available to the quantum simulator. More precisely, our generalized predicate is of the form:

$$\mathsf{qadv}_{\mathsf{X}}(t; t_S)$$

where $\mathsf{X} \subseteq \{\mathsf{a}, \mathsf{h}\}$, t is a term of order at-most one and t_S is a term of type timestamp set. Essentially, $\mathsf{qadv}_{\mathsf{X}}(t; t_S)$ states that t can be simulated in quantum polynomial-time using the randomness $\{\mathsf{qrnd}\ x \mid x \in t_S\}$ for quantum measurements, and with access to the quantum random tapes $\{\rho_x \mid x \in \mathsf{X}\}$.

We use $\{\mathsf{a}\}$ by default when no set of tapes X is specified, i.e. $\mathsf{qadv}(t; t_S)$ stands for $\mathsf{qadv}_{\{\mathsf{a}\}}(t; t_S)$.

C.4.2 Semantics of $\mathsf{qadv}_{\mathsf{X}}(\cdot;\cdot)$. If $\rho = (\rho_{\mathsf{a}}, \rho_{\mathsf{h}})$ is a pair of tapes, we let $\rho_{\mathsf{X}} \stackrel{\text{def}}{=} \{(x, \rho_x) \mid x \in \mathsf{X}\}$ be the subset of ρ's tapes labeled by X.

Let $\tau \stackrel{\text{def}}{=} \tau_c \star \tau_c^{\lambda} \star \tau_q^{\lambda}$ be a type where τ_c are the order 0 inputs, τ_c^{λ} the classical-access oracles of classical output type, and τ_q^{λ} the quantum-access oracles of quantum output type. We consider here for clarity a specific τ with all those components in this order, but the below definition should be understood as allowing any of the components τ_c, τ_c^{λ} and τ_q^{λ} of the type τ to be missing, as well as appearing in any possible ordering and interleaved in τ.

Let t be a term with a type of the form $\tau \to \tau^o$. Let t_S be a term of type timestamp set, then $\mathbb{M} \models \mathsf{qadv}_{\mathsf{X}}(t; t_S)$ iff. there exists:

- an ordering $\prec$ over $\llbracket t_S \rrbracket^{\eta,\rho}_{\mathbb{M}}$,
- and two machines $\mathcal{M}_c$: PPTM and $\mathcal{M}_q$: PQTM

such that for all:

- security parameter $\eta \in \mathbb{N}$ and tape $\rho \in \mathbb{T}_{\mathbb{M},\eta}$,
- and inputs:

$$\mathbf{a} = ((\mathbf{a}_c, \mathbf{a}_{\lambda_c}, \mathbf{a}_{\lambda_q}), \mathbf{a}_q) \in \llbracket \tau \star \mathsf{Hilb}_{\mathsf{msg}} \rrbracket^{\eta}_{\mathbb{M}}.$$

if we let S be the sequence of random values $(\llbracket \mathsf{qrnd}\ x \rrbracket^{\eta,\rho}_{\mathbb{M}} \mid x \in \llbracket t_S \rrbracket^{\eta,\rho}_{\mathbb{M}})$ orderered by $\prec$ and:

$$(w_c, w_q) = \mathsf{fold}_{\tau}(\mathcal{M}_c, \mathcal{M}_q, \mathbf{a}, S, \eta, \rho_{\mathsf{X}})$$

then:

$$\llbracket t \rrbracket^{\eta,\rho}_{\mathbb{M}}(\mathbf{a}) = \begin{cases} w_c & \text{if } \tau^o \in \mathbb{B} \\ w_q & \text{if } \tau^o \in \mathsf{Hilb}_{\mathbb{B}} \\ (w_c, w_q) & \text{if } \tau^o \in \mathbb{B} \star \mathsf{Hilb}_{\mathbb{B}} \end{cases}$$

where $\mathsf{fold}_{\tau}(\cdot)$ is defined inductively as:

- if $S = \emptyset$ then $\mathsf{fold}_{\tau}(\mathcal{M}_c, \mathcal{M}_q, \mathbf{a}, S, \eta, \rho_{\mathsf{X}})$ is equal to $\big(\mathcal{M}_c(1^{\eta}, \mathbf{a}_c, \mathbf{a}_{\lambda_c}, \rho_{\mathsf{X}}), \mathbf{a}_q\big)$.
- otherwise, if $S = (s, S')$ then:

$$\begin{array}{l} \mathsf{fold}_{\tau}(\mathcal{M}_c, \mathcal{M}_q, \mathbf{a}, S, \eta, \rho_{\mathsf{X}}) \stackrel{\text{def}}{=} \\ \text{if we let } w = \mathsf{fold}_{\tau}(\mathcal{M}_c, \mathcal{M}_q, \mathbf{a}, S', \eta, \rho_{\mathsf{X}}) \\ \text{and} (w_c, w_q) = \mathsf{approx}_{\mathsf{r}^{\eta}_{\mathbb{M}}}(\mathsf{Measure}^{\mathcal{M}_q}_{\mathbb{M},\eta})(\mathcal{M}_q(1^{\eta}, w, \mathbf{a}_{\lambda_q}, \rho_{\mathsf{X}}), s) \\ \text{then } \big(\mathcal{M}_c(1^{\eta}, w_c, \mathbf{a}_{\lambda_c}, \rho_{\mathsf{X}}), w_q\big). \end{array}$$

where n is η plus the length of the order 0 inputs in $\mathbf{a}$ — the lengths of the random tapes and of the order 1 arguments are ignored. Further if a is an order 0 input of a quantum type Hilb_τ (where $\tau \in \mathbb{B}$), then its length is the maximal length of any eigenvalue of a, i.e.:

$$\text{if } a = \textstyle\sum_c \alpha_c |c\rangle \text{ then } |a| \stackrel{\text{def}}{=} \max_{\{c \mid \alpha_c \neq 0\}} |c|.$$

Finally, we omit the type τ when it is clear from context, and write $\mathsf{fold}(\dots)$ instead of $\mathsf{fold}_\tau(\dots)$.

Remark 4. *The order 0 inputs, which are resp. $\mathbf{a}_c$ for the classical part and $\mathbf{a}_q$ for the quantum part, are only used once to initialize the computation in the case where $S = \emptyset$. On the other hand, the order 1 inputs are re-used at each iteration of the computation: the classical oracles $\mathbf{a}_{\lambda_c}$ are given to the classical machine $\mathcal{M}_c$, and the quantum oracles $\mathbf{a}_{\lambda_q}$ are given to the quantum machine $\mathcal{M}_q$.*

C.4.3 Error-free quantum simulation predicate. In the case where a term t can be simulated by an *error-free* quantum machine $\mathcal{M}_q$, the definition above can be significantly simplified by getting rid of all the aspect related to randomness. We also restrict this definition to machines that only have access to the adversarial random tape ρ_{a}, as this will be sufficient for our purposes.

More precisely, $\mathbb{M} \models \mathsf{qadv}_{\mathsf{E}}(t)$ iff.

$$\begin{aligned}
&\exists \mathcal{M}_c : \mathrm{PTM},\ \mathcal{M}_q : \mathrm{PQTM}_{\mathsf{E}}.\ P \in \mathbb{N}[X].\ \forall \eta \in \mathbb{N}.\ \forall \rho \in \mathbb{T}_{\mathbb{M},\eta}.\\
&\quad \forall \mathbf{a} = ((\mathbf{a}_c, \mathbf{a}_{\lambda_c}, \mathbf{a}_{\lambda_q}), \mathbf{a}_q) \in [\![\tau \star \mathsf{Hilb}_{\mathsf{msg}}]\!]^{\eta}_{\mathbb{M}}.\\
&\quad \text{if we let } (w_c, w_q) = \mathsf{fold}_{\mathsf{E}}(P(\eta))\\
&\quad \text{then } [\![t]\!]^{\eta,\rho}_{\mathbb{M}}(\mathbf{a}) = \begin{cases} w_c & \text{if } \tau^o \in \mathbb{B} \\ w_q & \text{if } \tau^o \in \mathsf{Hilb}_{\mathbb{B}} \\ (w_c, w_q) & \text{if } \tau^o \in \mathbb{B} \star \mathsf{Hilb}_{\mathbb{B}} \end{cases}
\end{aligned}$$

where $\mathsf{fold}_{\mathsf{E}}(\cdot)$ is defined inductively as:

- if $N = 0$ then $\mathsf{fold}_{\mathsf{E}}(N)$ is $\big(\mathcal{M}_c(1^\eta, \mathbf{a}_c, \mathbf{a}_{\lambda_c}, \rho_a), \mathbf{a}_q\big)$.
- otherwise:

$$\begin{aligned}
\mathsf{fold}_{\mathsf{E}}(N) \stackrel{\text{def}}{=} &\text{ if we let } && w = \mathsf{fold}_{\mathsf{E}}(N-1)\\
&\text{and let } && (w_c, w_q) = \mathcal{M}_q(1^\eta, w, \mathbf{a}_{\lambda_q}, \rho_{\mathsf{X}})\\
&\text{then } && \big(\mathcal{M}_c(1^\eta, w_c, \mathbf{a}_{\lambda_c}, \rho_{\mathsf{X}}), w_q\big).
\end{aligned}$$

D Adequacy of our Approximate Semantics

This section provides details from Section 4.3. We show that, under certain conditions on a term u, the distance between the approximate and the exact semantics of u is negligible. Consequently, proving the indistinguishability of two such terms in Squirrel, which relies on the approximate semantics, also implies their indistinguishability with respect to the exact semantics.

D.1 Preliminaries

We first present some preliminaries regarding statistical distance. The following proposition upper-bounds the statistical distance between the pushforward measures $f(\mu_1)$ and $f(\mu_2)$ by the statistical distance between the original measures μ_1 and μ_2.

Proposition 5. *Let $(\mathbb{A}, \mathcal{F}_{\mathbb{A}})$ and $(\mathbb{B}, \mathcal{F}_{\mathbb{B}})$ be two measurable spaces, μ_1 and μ_2 two distributions over $(\mathbb{A}, \mathcal{F}_{\mathbb{A}})$, and $f : \mathbb{A} \to \mathbb{B}$ a measurable function, then:*

$$\mathrm{d}_{\mathrm{stat}}(f(\mu_1), f(\mu_2)) \leq \mathrm{d}_{\mathrm{stat}}(\mu_1, \mu_2)$$

where, for any $i \in \{1, 2\}$, $f(\mu_i)$ is the pushforward of μ_i by f, i.e. $f(\mu_i)$ is the measure over $(\mathbb{B}, \mathcal{F}_{\mathbb{B}})$ defined by $\Pr(f(\mu_i) \in E) = \Pr(\mu_i \in f^{-1}(E))$ for any event E of $\mathcal{F}_{\mathbb{B}}$.

Proof. Let E be an event of $\mathbb{B}$, then:

$$\begin{aligned}
&\big|\Pr(f(\mu_1) \in E) - \Pr(f(\mu_2) \in E)\big|\\
&= \big|\Pr(\mu_1 \in f^{-1}(E)) - \Pr(\mu_2 \in f^{-1}(E))\big|\\
&\leq \sup_{E' \in \mathcal{F}_{\mathbb{A}}} \big|\Pr(\mu_1 \in E') - \Pr(\mu_2 \in E')\big|\\
&= \mathrm{d}_{\mathrm{stat}}(\mu_1, \mu_2)
\end{aligned}$$

Taking the supremum over all event E, we get that:

$$\mathrm{d}_{\mathrm{stat}}(f(\mu_1), f(\mu_2)) = \sup_{E \in \mathcal{F}_{\mathbb{B}}} \big|\Pr(f(\mu_1) \in E) - \Pr(f(\mu_2) \in E)\big| \leq \mathrm{d}_{\mathrm{stat}}(\mu_1, \mu_2) \qquad \square$$

Definition 2. *Let $(\mathbb{B}, \mathcal{F}_{\mathbb{B}})$ be a measurable space. The* cylinder σ-algebra *on the set of functions $\mathbb{A} \to \mathbb{B}$ is the σ-algebra generated by the sets $\{f \mid f(a_1) \in E_1, \dots, f(a_n) \in E_n\}$ for any integer n, $a_1, \dots, a_n \in \mathbb{A}$ and $E_1, \dots, E_n \in \mathcal{F}_{\mathbb{B}}$.*

Proposition 6. *Let $\mathbb{A}$ be a* countable *set and $(\mathbb{B}, \mathcal{F}_{\mathbb{B}})$ a measurable space. Then the evaluation function:*

$$e : \begin{cases} (\mathbb{A} \to \mathbb{B}) \times \mathbb{A} & \to \mathbb{B} \\ (f, a) & \mapsto f(a) \end{cases}$$

is measurable, where $\mathbb{A}$ is endowed with the discrete σ-algebra and $(\mathbb{A} \to \mathbb{B})$ with the cylinder σ-algebra.

Proof. Let $E \in \mathcal{F}_{\mathbb{B}}$ an event of $\mathbb{B}$. Then:

$$e^{-1}(E) = \{(f, a) \mid f(a) \in E\} = \textstyle\bigcup_{a \in \mathbb{A}} \{f \mid f(a) \in E\} \times \{a\}$$

Thus, the pre-image by e of any measurable set can be seen as a countable union of products of elements in the cylinder σ-algebra and the discrete σ-algebra, and is thus measurable. $\square$

Remark 5. *As an immediate corollary of Proposition 6, we get that e is measurable for any finer σ-algebra on $\mathbb{A} \to \mathbb{B}$ — a σ-algebra $(\mathbb{S}, \mathbb{F}_1)$ is finer that another σ-algebra $(\mathbb{S}, \mathbb{F}_2)$ when $\mathbb{F}_1 \supseteq \mathbb{F}_2$.*

Thus, e is also measurable if $\mathbb{A} \to \mathbb{B}$ is equipped with the discrete σ-algebra, which is the finest σ-algebra.

Proposition 7. *Let $(\mathbb{A}, \mathcal{F}_{\mathbb{A}})$, $(\mathbb{B}, \mathcal{F}_{\mathbb{B}})$ be measurable spaces such that for all $a \in \mathbb{A}$, $\{a\} \in \mathcal{F}_{\mathbb{A}}$ and for all $b \in \mathbb{B}$, $\{b\} \in \mathcal{F}_{\mathbb{B}}$. Let $(\mathbb{D}, \mathcal{F}_{\mathbb{D}})$ a measurable space and $(\Omega, \mathcal{F}_\omega, \mu)$ a probabilistic space. Let:*

$$A_1, A_2 : \Omega \to (\mathbb{A} \to \mathbb{B}) \qquad K : \Omega \to \mathbb{D} \qquad U : \Omega \to \mathbb{A}$$

be random variables, where $\mathbb{A} \to \mathbb{B}$ is endowed with the discrete σ-algebra.[1] If:

- *(K, U) is independent from A_1 and from A_2;*
- *$supp(U)$ is countable;*
- *and $supp(A_i(a))$ is countable for any $a \in \mathbb{A}$ and $i \in \{1, 2\}$;*

[1] We could weaken this to only require that $\mathbb{A} \to \mathbb{B}$ is endowed with the cylinder σ-algebra, but we do not need it.

then:

$$\mathrm{d_{stat}}\big((K,A_1(U)),(K,A_2(U))\big) \leq \sup_{a\in supp(U)} \mathrm{d_{stat}}(A_1(a),A_2(a)).$$

Proof. Let E be an event of $\mathbb{D}\times\mathbb{B}$, then:

$$\begin{aligned}
&\big|\Pr\left((K,A_1(U))\in E\right) - \Pr\left((K,A_2(U))\in E\right)\big| \\
\leq &\sum_{\substack{a\in\mathrm{supp}(U),\\ b\in\mathrm{supp}(A_i(a))}} \Big|\begin{array}{l}\Pr\left((K,b)\in E\wedge U=a\wedge A_1(a)=b\right)\\ -\ \Pr\left((K,b)\in E\wedge U=a\wedge A_2(a)=b\right)\end{array}\Big| \\
= &\sum_{\substack{a\in\mathrm{supp}(U),\\ b\in\mathrm{supp}(A_i(a))}} \Pr\left((K,b)\in E\wedge U=a\right)\cdot \\
&\qquad \big|\Pr\left(A_1(a)=b\right) - \Pr\left(A_2(a)=b\right)\big| \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(by independence)} \\
\leq &\sum_{\substack{a\in\mathrm{supp}(U),\\ b\in\mathrm{supp}(A_i(a))}} \Pr\left((K,b)\in E\wedge U=a\right)\cdot \mathrm{d_{stat}}\big(A_1(a),A_2(a)\big) \\
\leq &\sup_{a\in\mathrm{supp}(U)} \mathrm{d_{stat}}\big(A_1(a),A_2(a)\big)\cdot \\
&\qquad \sum_{\substack{a\in\mathrm{supp}(U),\\ b\in\mathrm{supp}(A_i(a))}} \Pr\left((K,b)\in E\wedge U=a\right) \\
= &\sup_{a\in\mathrm{supp}(U)} \mathrm{d_{stat}}\big(A_1(a),A_2(a)\big)
\end{aligned}$$

We conclude by taking the supremum over all events E. □

Remark 6. *The independence of A_1 and (K,U) (and of A_2 and (K,U)) is necessary in the above lemma.*

Indeed, take U an arbitrary random variable (e.g. $U=0$), and let A_1 and A_2 be random variables ignoring their argument (sampled according to U) such that A_1 is a balanced coin toss T, and $A_2=\neg T$ is the opposite of A_1. Finally, take $K=A_1$. We have that:

$$\mathrm{d_{stat}}\big((K,A_1(U)),(K,A_2(U))\big) = \mathrm{d_{stat}}\big((T,T),(T,\neg T)\big) = 1$$

since the images of (T,T) and $(T,\neg T)$ are disjoint.

D.2 Linking both Semantics for PQTMs

Roughly, the following lemma establishes that for any input $\mathbf{a}$, the statistical distance between the approximated quantum semantics $[\![\mathrm{att}(\mathbf{a})]\!]^{\eta,\rho}_{\mathbb{M}}$ and its exact counter-part is negligible whenever r_η is taken large enough.

Lemma 1. *Let $\mathbb{M}$ be an approximate model and $\eta\in\mathbb{N}$, such that $\mathbb{M}\models \mathrm{qadv}(\lambda(x,x_q).\,\mathrm{att}(\mathrm{qrnd}\ \{i\},x,x_q);\{i\})$. Let $\mathbb{A}$ be the input domain of $\mathrm{att}(\cdot)$ and $\mathbb{B}$ its output domain, i.e.:*

$$\mathbb{A} \stackrel{\text{def}}{=} [\![\tau \star \mathsf{Hilb}_{\mathrm{msg}}]\!]^{\eta}_{\mathbb{M}} \times \mathbb{T}^{\mathsf{a}}_{\mathbb{M},\eta} \qquad \mathbb{B} \stackrel{\text{def}}{=} [\![\mathrm{msg} \star \mathsf{Hilb}_{\mathrm{msg}}]\!]^{\eta}_{\mathbb{M}}$$

Let $\mathcal{M}_c$ be the PPTM and $\mathcal{M}_q$ be the PQTM provided by the quantum simulatability assumption on $\mathrm{att}(\mathrm{qrnd}\ \{i\},\cdot,\cdot)$. We let $\mathcal{M}$ be the hybrid quantum/classical computation defined by:

$$\mathcal{M}(\eta,\boldsymbol{a}) \stackrel{\text{def}}{=} \mathcal{M}_q(1^\eta,\mathcal{M}_c(1^\eta,a_c,a_c^\lambda,\rho_{\mathsf{a}}),a_q^\lambda,a_q,\rho_{\mathsf{a}})$$

for any η and $\boldsymbol{a}=(((a_c,a_c^\lambda,a_q^\lambda),a_q),\rho_{\mathsf{a}})\in\mathbb{A}$. Let $\Delta_{\mathsf{approx}}:\mathbb{A}\mapsto \mathrm{Distr}(\mathbb{B})$ and $\Delta_{\mathsf{exact}}:\mathbb{A}\mapsto\mathrm{Distr}(\mathbb{B})$ be the functions defined by:

$$\Delta_{\mathsf{approx}}(\boldsymbol{a}) \stackrel{\text{def}}{=} p\in\{0,1\}^{r^\eta_{\mathbb{M}}} \mapsto \mathrm{approx}_{r^\eta_{\mathbb{M}}}(\mathrm{Measure}^{\mathcal{M}_q}_{\mathbb{M},\eta})(\mathcal{M}(\eta,\boldsymbol{a}),p)$$

$$\Delta_{\mathsf{exact}}(\boldsymbol{a}) \stackrel{\text{def}}{=} p\in\{0,1\}^{\omega} \mapsto \quad \mathrm{exact}(\mathrm{Measure}^{\mathcal{M}_q}_{\mathbb{M},\eta})(\mathcal{M}(\eta,\boldsymbol{a}),p)$$

(also, recall that $\mathrm{Measure}^{\mathcal{M}_q}_{\mathbb{M},\eta}(x)$ is a distribution over $\mathbb{B}$ for any x produced by $\mathcal{M}_q$).

For any $n\in\mathbb{N}$, we let

$$\mathbb{A}_n \stackrel{\text{def}}{=} \{(((a_c,a_c^\lambda,a_q^\lambda),a_q),\rho_{\mathsf{a}})\in\mathbb{A} \mid |a_c|+|a_q|\leq n\}$$

be the subset of $\mathbb{A}$ of elements of total length at most n,[2] where the length of a pure quantum state $\sum_i c_i|w_i\rangle$ is $\max_i\{|w_i| \mid c_i\neq 0\}$.

Let $P\in\mathbb{Z}[X_\eta,X_c,X_q]$ be a polynomial upper-bounding $\mathcal{M}$'s running time, i.e. $\mathcal{M}(\eta,\boldsymbol{a})$ terminates in at-most $P(\eta,|a_c|,|a_q|)$ steps for any $\boldsymbol{a}=(((a_c,a_c^\lambda,a_q^\lambda),a_q),\rho_{\mathsf{a}})$. Then there exists $k_{\mathcal{M}}\in\mathbb{N}$ that only depends on $\mathcal{M}_q$ such that:

$$\sup_{\boldsymbol{a}\in\mathbb{A}_n}\left(\mathrm{d_{stat}}(\Delta_{\mathsf{approx}}(\boldsymbol{a}),\Delta_{\mathsf{exact}}(\boldsymbol{a}))\right) \leq \frac{1}{2^{r^\eta_{\mathbb{M}}-f(\eta,n)}}.$$
$$\text{(where } f(\eta,n)=k_{\mathcal{M}}\cdot P(\eta,n,n)-2)$$

Proof. $\mathcal{M}$ runs in time at-most P. Hence, on input

$$(((a_c,a_c^\lambda,a_q^\lambda),a_q),\rho_{\mathsf{a}})\in\mathbb{A},$$

we know that the length of $\mathcal{M}$'s tapes (ignoring the tape for the adversarial randomness) at the end of its execution is at-most $P(\eta,|a_c|,|a_q|)$ (as $\mathcal{M}$ cannot write more symbols on its tapes that its running time, see Proposition 4). Further, the tape containing the adversarial randomness still contains ρ_{a}, as it is read-only (assumption CM:AdvRO in Appendix C.2.4). This shows that

$$\mathrm{Measure}^{\mathcal{M}}_{\mathbb{M},\eta}(\mathcal{M}(\eta,\mathbf{a},\rho_{\mathsf{a}}))$$

is a discrete distribution whose support is of size at-most

$$2^{k_{\mathcal{M}}\times P(\eta,|a_c|,|a_q|)} \leq 2^{f(\eta,n)}$$

by taking $k_{\mathcal{M}}$ to be the number of input and output tapes of $\mathcal{M}_q$. Applying Proposition 3 twice, we get that:

$$\mathrm{d_{stat}}\Big(\Delta_{\mathsf{approx}}(\mathbf{a}),\mathrm{Measure}^{\mathcal{M}}_{\mathbb{M},\eta}(\mathcal{M}(\eta,\mathbf{a},\rho_{\mathsf{a}}))\Big) \leq \frac{1}{2^{r^\eta_{\mathbb{M}}-f(\eta)}}$$

$$\mathrm{d_{stat}}\Big(\Delta_{\mathsf{exact}}(\mathbf{a}),\mathrm{Measure}^{\mathcal{M}}_{\mathbb{M},\eta}(\mathcal{M}(\eta,\mathbf{a},\rho_{\mathsf{a}}))\Big) \leq \frac{1}{2^{\omega-f(\eta)}} = 0$$

We conclude immediately using the fact that the statistical distance is transitive. □

We will later use this bound to show that the statistically distance between the approximated and exact semantics of a particular set of terms t is negligible.

[2] Note that the length does not account for order 1 oracles and ρ_{a}.

D.3 Linking both Semantics for Terms

Before establishing a lemma to upper-bound the statistical distance between the approximate and exact semantics of *admissible* terms, we first introduce an extended notion of models where a different amount of randomness can be used by each quantum slice.

Definition 3. *An* extended model $\mathbb{M}$ *is a model where the amount of random bits* $r^{\eta}_{\mathbb{M}}(i)$ *used in slice* i *of* $\mathbb{T}^{\mathsf{q}}_{\mathbb{M},\eta}$*, i.e. the length of* $[\![\mathsf{qrnd}\ i]\!]^{\eta,\rho}_{\mathbb{M}'}$*, may depend on* i*, i.e.:*

$$r^{\eta}_{\mathbb{M}'} : [\![\mathsf{timestamp}]\!]^{\eta}_{\mathbb{M}} \to \mathbb{N} \cup \{\omega\}.$$

As already mentioned, the semantics of a global formula may not be well-defined in an exact model. The same issue arises for an extended model $\mathbb{M}$, when there exists some i such that $r^{\eta}_{\mathbb{M}}(i) = \omega$. Indeed, if t is a term with τ, then:

$$[\![t]\!]^{\eta}_{\mathbb{M}} : \mathbb{T}_{\mathbb{M},\eta} \to [\![\tau]\!]^{\eta}_{\mathbb{M}}$$

is a well-defined *function*, but may not be a random variable. Still, we can observe that the following fragment of the global logic remains well-defined, as it only exploits the functional interpretation of terms, and not their probabilistic interpretation.

$$\bar{F} ::= \bar{F} \tilde{\vee} \bar{F} \mid \tilde{\neg} F_{\mathsf{e}} \mid \tilde{\bot} \mid \tilde{\forall} x.\bar{F} \mid [t]_{\mathsf{e}} \mid \det(t) \mid \mathsf{const}(t) \mid \mathsf{qrand}(t;t) \mid \mathsf{qadv}(t;t)$$

Indeed, the semantics of this fragment of the global logic is well-defined in any exact or extended model since:

- $\det(t)$ only requires that t's semantics is independent of ρ, and is thus well-defined even if $[\![t]\!]_{\mathbb{M}}$ is not a random variable.
- similarly, $\mathbb{M} \models [t]_{\mathsf{e}}$ iff. $[\![t]\!]^{\eta,\rho}_{\mathbb{M}} = 1$ for any ρ and η, and does not exploit a probabilistic property of $[\![t]\!]_{\mathbb{M}}$;
- the quantum randomness usage predicate $\mathsf{qrand}(\cdot;\cdot)$ only relies on the generalized sub-terms $\mathcal{ST}_{\mathcal{E}}(\cdot)$ and the exact predicate (c.f. Appendix E.2);
- similarly, the quantum simulation predicate $\mathsf{qadv}(t;t)$ is a functional predicate.

We write $\mathbb{M} \models_{\mathsf{ext}} \bar{F}$ if $\bar{F}$ holds in an extended model $\mathbb{M}$, and write $\models_{\mathsf{ext}} \bar{F}$ if $\bar{F}$ holds in any extended model.

Remark 7. *Any property of terms that only relies on standard computational aspects of the semantics (e.g. β-reduction) is valid in any extended model. E.g.:*

$$\models_{\mathsf{ext}} [(\lambda x.t)\ t' = (t\{x \mapsto t'\})]_{\mathsf{e}}.$$

The following lemma upper-bounds the statistical distance between the approximate and exact semantics of *admissible* terms, where a term t is admissible if, essentially:

1) t never uses a quantum random slice more than once (essentially, if $\mathsf{att}(\mathsf{qrnd}\ i_1, c_1, q_1)$ and $\mathsf{att}(\mathsf{qrnd}\ i_2, c_2, q_2)$ are sub-terms of t, then it must be the case that $[\![i_1 = i_2]\!]$ whenever $[\![(c_1, q_1) = (c_2, q_2)]\!]^{\eta,\rho}_{\mathbb{M}}$);
2) and if the length of its sub-terms is uniformly bounded.

This lemma is shown by applying Lemma 1 once for each random slice i. Condition 1) guarantees that there is indeed at most a single usage of slice i, while condition 2) is used to bound the support of the set of inputs $\mathbb{A}_n$ of Lemma 1.

Proposition 8. *Let t be a well-typed term in $\mathcal{E}$ of type τ_t, where τ_t is a type of order 0. If there exists k and v, v_q such that:*

1) t is equal to $(s\ i) \stackrel{\mathsf{def}}{=} (k\ i)\ (\mathsf{att}(\mathsf{qrnd}\ i, v\ i, v_q\ i))$ for any $(i : \mathsf{timestamp})$

$$\models_{\mathsf{ext}} [\forall i.\ t = s\ i]_{\mathsf{e}} \tag{2}$$

and the quantum randomness is properly partitioned between the continuation $(k\ i)$, the arguments $(v\ i)$ and $(v_q\ i)$, and the adversarial quantum computation $\mathsf{att}(i,\cdot,\cdot)$. More precisely, we assume a relation $<_q$: $(\mathsf{timestamp})^2 \to \mathsf{bool}$ such that $<_q$ is a total deterministic order such that: a) $(k\ i)$ only uses the quantum randomness strictly greater than i; b) $(v\ i, v_q\ i)$ only uses the quantum randomness strictly smaller than i. Formally:

$$\models_{\mathsf{ext}} \tilde{\forall} i.\ \det(i) \tilde{\Rightarrow} \begin{array}{ll} & \mathsf{qrand}(k\ i\ ; \{i' | i <_q i'\}) \\ \tilde{\wedge} & \mathsf{qrand}((v\ i, v_q\ i); \{i' | i >_q i'\}) \end{array} \tag{3}$$

2) $(v\ i, v_q\ i)$ can be simulated by a polynomial-time quantum Turing machine, i.e.:

$$\models_{\mathsf{ext}} \tilde{\forall} i.\ \det(i) \tilde{\Rightarrow} \mathsf{qadv}_{\{\mathsf{a},\mathsf{h}\}}((v\ i, v_q\ i); \{i' \mid i >_q i'\}) \tag{4}$$

If we let $\mathbb{M}_{\mathsf{e}}$ be the exact model corresponding to $\mathbb{M}$ (i.e. $\mathbb{M}_{\mathsf{e}}$ is exactly $\mathbb{M}$, except that $r^{\eta}_{\mathbb{M}_{\mathsf{e}}} = \omega$ for any η), then:

$$\mathsf{d}_{\mathsf{stat}}([\![t]\!]^{\eta}_{\mathbb{M}}, [\![t]\!]^{\eta}_{\mathbb{M}_{\mathsf{e}}}) \leq \frac{|[\![\mathsf{timestamp}]\!]^{\eta}_{\mathbb{M}}|}{2^{r^{\eta}_{\mathbb{M}} - f(\eta)}}$$

where f is a polynomial that only depends on $\mathbb{M}$.

Proof. We start with a first observation before introducing the overall proof structure.

First observation. First, we observe that for any extended model $\mathbb{M}$, the length of $v\ i$ and $v_q\ i$ are uniformly bounded, i.e. there exists a family of length $\mathsf{L} \in \mathbb{N}^{\mathbb{N}}$ such that:

$$\mathbb{M}[x_{\mathsf{L}} \mapsto \mathbb{1}_{\mathsf{L}}] \models [\forall i.\ \mathsf{len}(v\ i) + \mathsf{len}(v_q\ i) \leq x_{\mathsf{L}}]_{\mathsf{e}}. \tag{5}$$

This immediately follows from Eq. (4). Indeed, take a model $\mathbb{M}$. We know that $[\![(v\ i, v_q\ i)]\!]^{\eta}_{\mathbb{M}}$ can be simulated in polynomial-time by a quantum Turing machine. Let $\mathcal{M}_i$ be this machine and P_i a polynomial bounding its execution time. Then, thanks to Proposition 4, we know that $\mathsf{L}^i_{\eta} = P_i(\eta)$ is an upper-bound on $[\![(v\ i, v_q\ i)]\!]^{\eta}_{\mathbb{M}}$'s length. We conclude by taking $\mathsf{L}_{\eta} = \max_{i \in [\![\mathsf{timestamp}]\!]^{\eta}_{\mathbb{M}}} \mathsf{L}^i_{\eta}$.
(Note that we only exploited the fact that timestamp is a finite type, and that we have a bound L^i_{η} for every i. This argument could be generalized to other finite types in the same way.)

Bounding $\mathsf{d}_{\mathsf{stat}}([\![t]\!]^{\eta}_{\mathbb{M}}, [\![t]\!]^{\eta}_{\mathbb{M}_{\mathsf{e}}})$. Let $\eta \in \mathbb{N}$ be a value of the security parameter. For every $k \leq |[\![\mathsf{timestamp}]\!]^{\eta}_{\mathbb{M}}|$, we let $\mathbb{M}_k$ be the extended model where ω bits are used for the first k slices of $[\![\mathsf{timestamp}]\!]^{\eta}_{\mathbb{M}}$ (when ordered by $[\![<_q]\!]^{\eta,\rho}_{\mathbb{M}}$), and $r^{\eta}_{\mathbb{M}}$ bits are used for the remaining slices. More precisely, let $i_1 \ldots, i_{l_{\eta}}$ be the enumeration of $[\![\mathsf{timestamp}]\!]^{\eta}_{\mathbb{M}}$ in *increasing* order w.r.t. $[\![<_q]\!]^{\eta,\rho}_{\mathbb{M}}$, i.e. such that:

$$i_1 <_q \cdots <_q i_{l_{\eta}}.$$

Then:

$$r^{i}_{\mathbb{M}_k} = \begin{cases} \omega & \text{if } i \in \{i_1 \ldots, i_k\} \\ r^{\eta}_{\mathbb{M}} & \text{otherwise} \end{cases}. \tag{6}$$

Observe that we have $\mathbb{M} = \mathbb{M}_0$ and $\mathbb{M}_{\mathsf{e}} = \mathbb{M}_{l_{\eta}}$.

First, using first the triangular inequality and then Eq. (2) for every $i \in [\![\mathsf{timestamp}]\!]^\eta_{\mathbb{M}}$, we know that:

$$\begin{aligned}
&\mathsf{d}_{\mathsf{stat}}([\![t]\!]^\eta_{\mathbb{M}}, [\![t]\!]^\eta_{\mathbb{M}_{\mathsf{e}}}) \\
=\ &\mathsf{d}_{\mathsf{stat}}([\![t]\!]^\eta_{\mathbb{M}_0}, [\![t]\!]^\eta_{\mathbb{M}_{l_\eta}}) \\
\leq\ &\sum_{0 \leq i < l_\eta} \mathsf{d}_{\mathsf{stat}}([\![t]\!]^\eta_{\mathbb{M}_i}, [\![t]\!]^\eta_{\mathbb{M}_{i+1}}) \\
=\ &\sum_{0 \leq i < l_\eta} \mathsf{d}_{\mathsf{stat}}([\![s\ i]\!]^\eta_{\mathbb{M}_i[i \mapsto 1^\eta_i]}, [\![s\ i]\!]^\eta_{\mathbb{M}_{i+1}[i \mapsto 1^\eta_i]})
\end{aligned}$$

To conclude, it only remains to bound, for any $0 \leq i < l_\eta$, the quantity:

$$\mathsf{d}_{\mathsf{stat}}([\![s\ i]\!]^\eta_{\mathbb{M}_i[i \mapsto 1^\eta_i]}, [\![s\ i]\!]^\eta_{\mathbb{M}_{i+1}[i \mapsto 1^\eta_i]}) \tag{7}$$

Take $0 \leq i < l_\eta$, and let $\mathbb{M}_l \stackrel{\text{def}}{=} \mathbb{M}_i[i \mapsto 1^\eta_i]$ and $\mathbb{M}_r \stackrel{\text{def}}{=} \mathbb{M}_{i+1}[i \mapsto 1^\eta_i]$. To upper-bound the statistical distance introduced when switching the i-th adversarial call from an approximated to an exact measurement (i.e. the distance in Eq. (7)), we are going to decompose $[\![s\ i]\!]^\eta_{\mathbb{M}_l}$ and $[\![s\ i]\!]^\eta_{\mathbb{M}_r}$ as an application of the i-th adversarial computation to a common quantity, which will pave the way for the application of Lemma 1.

We first introduce some preliminary definitions before doing the actual decomposition.

Preliminary definitions. Recall that τ_t is the type of t (and is of order 0). Let τ be the type decorating att and let $\tau_i \stackrel{\text{def}}{=} \mathsf{msg}$, and $\tau_o \stackrel{\text{def}}{=} \mathsf{msg}$ (we introduce such generic notations to ease the generalization to an attacker symbol of a more generic type).

Let $x \in \{l, r\}$ be a label denoting whether we are before or after switching the i-th call to $\mathsf{att}(\cdot)$ from an approximated to an exact quantum measurement. Then, consider the following functions:

- A_x is the function applying the evaluation of the i-th adversary computation:

$$\mathsf{A}_x : \left\{ \begin{array}{ccccc} \mathcal{R}_x & \to & [\![\tau_i]\!]^\eta_{\mathbb{M}} \times \mathbb{T}^{\mathsf{a}}_{\mathbb{M},\eta} & \to & [\![\tau_o]\!]^\eta_{\mathbb{M}} \\ p & \mapsto & (w, \rho_{\mathsf{a}}) & \mapsto & [\![t_{\mathsf{att}}]\!]^{\eta,\rho(p,\rho_{\mathsf{a}})}_{\mathbb{M}_x}(w) \end{array} \right.$$

  where $t_{\mathsf{att}} \stackrel{\text{def}}{=} \lambda(x, x_q).\ \mathsf{att}(\mathsf{qrnd}\ i, x, x_q)$, the set $\mathcal{R}_x = \{0,1\}^{\mathsf{r}^\eta_{\mathbb{M}}}$ if $x = l$ and $\{0,1\}^\omega$ if $x = r - \mathcal{R}_x$ is the randomness used for the i-th random slice — and $\rho(p, \rho_{\mathsf{a}})$ is a random tape filled with zero except for the i-th slice (which is filled with p), and for the adversarial randomness which is filled with ρ_{a}.
- K_x and V_x be the functions computing, respectively, $[\![k\ i]\!]_{\mathbb{M}_x}$ and $[\![(v, v_q)\ i]\!]_{\mathbb{M}_x}$:

$$\mathsf{K}_x : \left\{ \begin{array}{ccc} \mathbb{T}_{\mathbb{M},\eta} & \to & ([\![\tau_o]\!]^\eta \to [\![\tau_t]\!]^\eta) \\ \rho & \mapsto & [\![k\ i]\!]^{\eta,\rho}_{\mathbb{M}_x} \end{array} \right.$$

$$\mathsf{V}_x : \left\{ \begin{array}{ccc} \mathbb{T}_{\mathbb{M},\eta} & \to & [\![\tau_i]\!]^\eta \times \mathbb{T}^{\mathsf{a}}_{\mathbb{M},\eta} \\ \rho & \mapsto & ([\![(v\ i, v_q\ i)]\!]^{\eta,\rho}_{\mathbb{M}_x}, \rho_{\mathsf{a}}) \end{array} \right.$$

- e is the evaluation function:

$$e : \left\{ \begin{array}{ccc} ([\![\tau_o]\!]^\eta \to [\![\tau_t]\!]^\eta) \times [\![\tau_o]\!]^\eta & \to & [\![\tau_t]\!]^\eta \\ (f, a) & \mapsto & f(a) \end{array} \right. .$$

e is a deterministic function which does not take as input any random bits. By Proposition 6, we know that e is a measurable function (where the function space is endowed with the discrete σ-algebra, see Remark 5). $\mathsf{A}_x, \mathsf{K}_x$ and V_x will be shown to be random variables from the set of tapes $\mathcal{R}_x$ for A_x and $\mathbb{T}_{\mathbb{M},\eta}$ for K_x and V_x.

Decomposition and $\mathsf{d}_{\mathsf{stat}}$ *upper-bound.* First, remark that $[\![s\ i]\!]^\eta_{\mathbb{M}_x}$ follows the same distribution as $e(\mathsf{K}_x, (\mathsf{A}_x \mathsf{V}_x))$.

Moreover, using Eq. (2), we have that for any tapes $\rho_l \in \mathbb{T}_{\mathbb{M}_l,\eta}$ and $\rho_r \in \mathbb{T}_{\mathbb{M}_r,\eta}$ that coincide on all components except slice i:

$$[\![k\ i]\!]^{\rho_l}_{\mathbb{M}_l} = [\![k\ i]\!]^{\rho_r}_{\mathbb{M}_r} \quad [\![v\ i]\!]^{\rho_l}_{\mathbb{M}_l} = [\![v\ i]\!]^{\rho_r}_{\mathbb{M}_r} \quad [\![v_q\ i]\!]^{\rho_l}_{\mathbb{M}_l} = [\![v_q\ i]\!]^{\rho_r}_{\mathbb{M}_r} \tag{8}$$

Hence we can swap K_l for K_r (resp. V_l for V_r):

$$\begin{aligned}
&\mathsf{d}_{\mathsf{stat}}([\![s\ i]\!]^\eta_{\mathbb{M}_l}, [\![s\ i]\!]^\eta_{\mathbb{M}_r}) \\
=\ &\mathsf{d}_{\mathsf{stat}}\big(e(\mathsf{K}_l, \mathsf{A}_l(\mathsf{V}_l)),\ e(\mathsf{K}_r, \mathsf{A}_r(\mathsf{V}_r))\big) \\
=\ &\mathsf{d}_{\mathsf{stat}}\big(e(\mathsf{K}_l, \mathsf{A}_l(\mathsf{V}_l)),\ e(\mathsf{K}_l, \mathsf{A}_r(\mathsf{V}_l))\big).
\end{aligned}$$

Using the fact that e is a measurable function and Proposition 5, we know that:

$$\mathsf{d}_{\mathsf{stat}}\Big(e(\mathsf{K}_l, \mathsf{A}_l(\mathsf{V}_l)),\ e(\mathsf{K}_l, \mathsf{A}_r(\mathsf{V}_l))\Big) \leq \mathsf{d}_{\mathsf{stat}}\Big((\mathsf{K}_l, \mathsf{A}_l(\mathsf{V}_l)),\ (\mathsf{K}_l, \mathsf{A}_r(\mathsf{V}_l))\Big).$$

Our goal is now to apply Proposition 7. As a first step, we want to see $\mathsf{A}_l, \mathsf{A}_r, \mathsf{V}_l$ and K_l as random variables sharing a common probability space. To that we, we consider the space $\mathbb{T}'_{\mathbb{M}_l,\eta}$ which is the set $\mathbb{T}_{\mathbb{M}_l,\eta} \times \mathcal{R}_r$:

- $\mathcal{R}_r$ provides randomness to A_r;
- $\mathbb{T}_{\mathbb{M}_l,\eta}$, minus the i-th slice of randomness, provides randomness to V_l and K_l;
- the i-th slice of randomness of $\mathbb{T}_{\mathbb{M}_l,\eta}$ provides randomness to A_l.

The functions $\mathsf{A}_l, \mathsf{A}_r, \mathsf{V}_l$ and K_l all ignore the randomness if $\mathbb{T}'_{\mathbb{M}_l,\eta}$ which they do not use.

Then:

- K_l is a discrete random variable, since it only depends on the quantum random slices that have not yet been changed to ω (see Eq. (3) and Eq. (6)), and can thus be seen as a random variable whose probability space is a finite set of tapes.
- V_l is a discrete random variable, since V_l corresponds to the evaluation of a polynomial-time quantum Turing machine (see Eq. (4)). Thus, for any $\eta \in \mathbb{N}$, V_l can thus only take a finite number of values.
- A_l is a discrete random variable since $\mathcal{R}_l$ is finite. We conclude by observing that a discrete random variable is also a random variable for the cylinder σ-algebra (as it is a coarser σ-algebra).
- For A_r, we must exploit the fact that att can be simulated by a polynomial-time quantum Turing machine. We need to show that A_r is a random variable for the cylinder σ-algebra. To that end, it is sufficient to show that:

  $$\mathsf{A}_r^{-1}(\{f \mid f(a_1) \in E_1, \ldots, f(a_n) \in E_n\})$$

  is a measurable event of $\mathcal{R}_r$, for any arbitrary $a_1, \ldots, a_n$ and events $E_1, \ldots, E_n$.

Let $l \in \mathbb{N}$ be the maximum length of the a_i's, and $P(x)$ a polynomial bounding the number of random bits that A_r may read on an input of length at-most x (which exists, since A_r can be simulated by a polynomial-time quantum Turing machine). Let:

$$\overline{\mathsf{A}_r} : \begin{cases} \{0,1\}^{P(l)} & \to & [\![\tau_i]\!]^\eta \times \mathbb{T}^{\mathsf{a}}_{\mathbb{M},\eta} & \to & [\![\tau_o]\!]^\eta \\ p & \mapsto & (w, \rho_{\mathsf{a}}) & \mapsto & \mathsf{A}_r(p')(w, \rho_{\mathsf{a}}) \end{cases}$$

where p' is p extended into an infinite bit-strings using zeros. Clearly, we have that for any $p \in \mathcal{R}_r$:

$$\mathsf{A}_r(p)(a_i) = \overline{\mathsf{A}_r}(p_{|P(l)})(a_i).$$

for any i and ρ_{a}.

Furthermore, $\overline{\mathsf{A}_r}$ is a discrete random variable. Thus,

$$\begin{aligned} &\{p \in \{0,1\}^\omega \mid \textstyle\bigwedge_i \mathsf{A}_r(p)(a_i) \in E_i\} \\ = \ &\{p.p' \in \{0,1\}^\omega \mid p \in \{0,1\}^{P(l)}, \textstyle\bigwedge_i \overline{\mathsf{A}_r}(p)(a_i) \in E_i\} \end{aligned}$$

Since $\overline{\mathsf{A}_r}$ is a discrete random variable, we know that

$$\{p \in \{0,1\}^{P(l)} \mid \textstyle\bigwedge_i \overline{\mathsf{A}_r}(p)(a_i) \in E_i\}$$

is a finite subset of $p \in \{0,1\}^{P(l)}$. Thus, the segment:

$$\{p.p' \in \{0,1\}^\omega \mid p \in \{0,1\}^{P(l)}, \textstyle\bigwedge_i \overline{\mathsf{A}_r}(p)(a_i) \in E_i\}$$

is a Lebesgue subset of $\{0,1\}^\omega$. This concludes the proof that A_r is a random variable when the target function space is equipped with the cylinder σ-algebra.

- for any $w \in [\![\tau_i]\!]^\eta_{\mathbb{M}}$, $\mathrm{supp}(\mathsf{A}_l(w))$ is discrete since $\mathcal{R}_l$ is a finite set of tapes.
- for any $w \in [\![\tau_i]\!]^\eta_{\mathbb{M}}$, $\mathrm{supp}(\mathsf{A}_r(w))$ is discrete since $\mathsf{A}_r(w)$ corresponds to the evaluation of a polynomial-time quantum Turing machine on input w, and can thus only take a finite number of values.

Finally, A_l and $(\mathsf{K}_l, \mathsf{V}_l)$ are clearly independent random variables since they share no randomness. Similarly, A_r and $(\mathsf{K}_l, \mathsf{V}_l)$ are independent random variables. Hence, using Proposition 7, we know that:

$$\mathsf{d}_{\mathrm{stat}}((\mathsf{K}_l, \mathsf{A}_l(\mathsf{V}_l)), (\mathsf{K}_l, \mathsf{A}_r(\mathsf{V}_l))) \leq \sup_{a \in \mathrm{supp}(\mathsf{V}_l)} \mathsf{d}_{\mathrm{stat}}(\mathsf{A}_l(a), \mathsf{A}_r(a)).$$

Recall that $\mathsf{V}_l(\rho) = ([\![(v\ i, v_q\ i)]\!]^{\eta,\rho}_{\mathbb{M}_l}, \rho_{\mathsf{a}})$. Thanks to Eq. (5), we know that:

$$\text{for any } x \in \mathrm{supp}(\mathsf{V}_l).\ |x| \leq 2^{\mathsf{L}_\eta + \mathsf{L}^a_\eta}$$

(where the length of the of the adversarial random tape ρ_{a} is ignored, as in Lemma 1). Hence, using Lemma 1, there exists a polynomial f' (that only depends on the polynomial runtime of the interpretation of att in $\mathbb{M}$) such that:

$$\sup_{a \in \mathrm{supp}(\mathsf{V}_l)} \mathsf{d}_{\mathrm{stat}}(\mathsf{A}_l(a), \mathsf{A}_r(a)) \leq \frac{1}{2^{\mathsf{r}^\eta_{\mathbb{M}} - f'(\eta, \mathsf{L}_\eta)}}.$$

We conclude by taking $f(\eta) = f'(\eta, L_\eta)$. □

Proposition 9. *Let u be a term of order 0 that can be simulated in quantum polynomial-time in any extended model:*

$$\models_{\mathsf{ext}} \mathsf{qadv}_{\{\mathsf{a},\mathsf{h}\}}(u; \{i \mid i : \top\}) \tag{9}$$

Let $\mathbb{M}$ be a extended model, $\mathcal{M}_c$ a PPTM, $\mathcal{M}_q$ a PQTM, and R a polynomial in $\mathbb{N}[X]$ such that $(\mathcal{M}_c, \mathcal{M}_q, R)$ form a PTIME hybrid computation. Then:

$$\mathsf{d}_{\mathrm{stat}}\begin{pmatrix} \mathsf{fold}_\tau(\mathcal{M}_c, \mathcal{M}_q, [\![u]\!]^{\eta,\rho}_{\mathbb{M}}, S, \eta, \rho_{\mathsf{a}}), \\ \mathsf{fold}_\tau(\mathcal{M}_c, \mathcal{M}_q, [\![u]\!]^{\eta,\rho}_{\mathbb{M}}, \hat{S}, \eta, \rho_{\mathsf{a}}) \end{pmatrix} \leq \frac{R(\eta)}{2^{\mathsf{r}^\eta_{\mathbb{M}} - f(\eta)}}$$

where ρ is sampled in $\mathbb{T}_{\mathbb{M},\eta}$, S is sampled in $([\![\mathsf{qrand}]\!]^\eta_{\mathbb{M}})^{R(\eta)}$, and $\hat{S}$ is sampled in $(\{0,1\}^\omega)^{R(\eta)}$.

Further, the function $f(\eta)$ is a polynomial whose degree and coefficient only depends on $\mathcal{M}_c$, $\mathcal{M}_q$, and $\mathcal{U}$ (the machine justifying the quantum simulatability hypothesis of Eq. (9)).

Proof (sketch). The proof is very similar to the proof of Proposition 8: we replace the approximated partial quantum measurements following each call to $\mathcal{M}_q$ by exact partial quantum measurements, one-by-one, starting from the innermost call.

There are $R(\eta)$ swaps to perform, and each swap incurs an error:

$$\frac{1}{2^{\mathsf{r}^\eta_{\mathbb{M}} - f'(\eta, \mathsf{L}_\eta)}}$$

where $f'(\eta, \mathsf{L})$ is a polynomial function whose coefficient only depends on $\mathcal{M}_q$, and where L_η is an upper-bound on the length of the inputs to $\mathcal{M}_q$ in the computation of

$$\mathsf{fold}_\tau(\mathcal{M}_c, \mathcal{M}_q, [\![u]\!]^{\eta,\rho}_{\mathbb{M}}, S, \eta, \rho_{\mathsf{a}}) \tag{10}$$

To bound L_η, we observe that:

- Thanks to Eq. (9), we know that there exists $\mathcal{U}$ a polynomial-time quantum machine justifying the quantum simulatability of $[\![u]\!]^\eta_{\mathbb{M}}$. Thus, we know that there exists a polynomial upper-bound on the length of $[\![u]\!]^\eta_{\mathbb{M}}$.
- Thanks to the fact that $(\mathcal{M}_c, \mathcal{M}_q, R)$ is a PTIME hybrid computation and to the fact that $[\![u]\!]^\eta_{\mathbb{M}}$ is of polynomial length, we know that there exists a polynomial bound L_η on the length of the intermediate inputs to $\mathcal{M}_q$ during the computation in Eq. (10). This polynomial only depends on $\mathcal{M}_q$, $\mathcal{M}_c$ and $\mathcal{U}$.

Thus, $f(\eta) = f'(\eta, L_\eta)$ is a polynomial which only depends on $\mathcal{M}_c$, $\mathcal{M}_q$, and $\mathcal{U}$ (the runtime of the machine justifying Eq. (9)). □

Definition 4. *We say that a term u is well-formed if u is of order at-most one, and:*

1) if u is of order 1 then is does not requires quantum measurements, i.e.:

$$\models_{\mathsf{ext}} \mathsf{qrand}(u; \emptyset) \tag{11}$$

2) if u is of order 0, then it must satisfies the assumptions of Proposition 8 and Proposition 9, i.e. there exists k and v, v_q such that:

- *u is equal to $(s\ i) \overset{\mathrm{def}}{=} (k\ i)\ (\mathsf{att}(\mathsf{qrnd}\ i, v\ i, v_q\ i))$ for any $(i : \mathsf{timestamp})$:*

$$\models_{\mathsf{ext}} [\forall i.\ u = s\ i]_{\mathsf{e}}$$

- *there exists a total deterministic order $<_q$: $(\mathsf{timestamp})^2 \to \mathsf{bool}$ such that*

$$\models_{\mathsf{ext}} \tilde{\forall} i.\ \det(i) \mathrel{\tilde{\Rightarrow}} \begin{array}{ll} & \mathsf{qrand}(k\ i \quad ; \{i' | i <_q i'\}) \\ \tilde{\wedge} & \mathsf{qrand}((v\ i, v_q\ i); \{i' | i >_q i'\}) \end{array}$$

- *$(v\ i, v_q\ i)$ can be simulated by a PQTM:*

$$\models_{\mathsf{ext}} \tilde{\forall} i.\ \det(i) \mathrel{\tilde{\Rightarrow}} \mathsf{qadv}_{\{\mathsf{a},\mathsf{h}\}}((v\ i, v_q\ i); \{i' \mid i >_q i'\})$$

- *u can be simulated by a PQTM:*

$$\models_{\mathsf{ext}} \mathsf{qadv}_{\{\mathsf{a},\mathsf{h}\}}(u; \{i \mid i : \top\})$$

Note that the definition of $\sim$ states that for the top-level distinguisher, the measurement randomness source S is sampled independently at random in $(\llbracket \mathsf{qrand} \rrbracket^{\eta}_{\mathbb{M}})^{R(\eta)}$ (see Section 4.2). So, when considering $\sim$ in an exact model, the top-level distinguisher is also performing exact measurements. We can thus express the adequacy of $\sim$ as follows.

Proposition 10. *Let $u_1, \ldots, u_{2\,n}$ be a sequence of* well-formed *terms. Let $\mathbb{M}$ be a model and $\mathbb{M}_{\mathsf{e}}$ be the corresponding exact model. If $\mathbb{M}$ is such that $r^{\eta}_{\mathbb{M}} \geq \eta$ for any η, then*

$$\mathbb{M} \models u_1, \ldots, u_n \sim u_{n+1}, \ldots, u_{2\,n} \tag{12}$$

if and only if

$$\mathbb{M}_{\mathsf{e}} \models u_1, \ldots, u_n \sim u_{n+1}, \ldots, u_{2\,n}.$$

Proof. Let $\mathcal{M}_q$ be a PQTM, $\mathcal{M}_c$ a PPTM, and R a polynomial. We must prove that

$$\mathsf{d}_{\mathsf{stat}}\begin{pmatrix}\mathsf{fold}_{\tau}(\mathcal{M}_c, \mathcal{M}_q, \llbracket u_1, \ldots, u_n \rrbracket^{\eta,\rho}_{\mathbb{M}_{\mathsf{e}}}, \hat{S}, \eta, \rho_{\mathsf{a}}),\\ \mathsf{fold}_{\tau}(\mathcal{M}_c, \mathcal{M}_q, \llbracket u_{n+1}, \ldots, u_{2\,n} \rrbracket^{\eta,\rho}_{\mathbb{M}_{\mathsf{e}}}, \hat{S}, \eta, \rho_{\mathsf{a}})\end{pmatrix} \in \mathsf{negl}(\eta)$$

if and only if

$$\mathsf{d}_{\mathsf{stat}}\begin{pmatrix}\mathsf{fold}_{\tau}(\mathcal{M}_c, \mathcal{M}_q, \llbracket u_1, \ldots, u_n \rrbracket^{\eta,\rho}_{\mathbb{M}}, S, \eta, \rho_{\mathsf{a}}),\\ \mathsf{fold}_{\tau}(\mathcal{M}_c, \mathcal{M}_q, \llbracket u_{n+1}, \ldots, u_{2\,n} \rrbracket^{\eta,\rho}_{\mathbb{M}}, S, \eta, \rho_{\mathsf{a}})\end{pmatrix} \in \mathsf{negl}(\eta)$$

when ρ is sampled in $\mathbb{T}_{\mathbb{M},\eta}$, S is sampled in $(\llbracket \mathsf{qrand} \rrbracket^{\eta}_{\mathbb{M}})^{R(\eta)}$, and $\hat{S}$ is sampled in $(\{0,1\}^{\omega})^{R(\eta)}$.

Using the triangular inequality twice, it is sufficient to prove that the following quantities are negligible:

$$\mathsf{d}_{\mathsf{stat}}\begin{pmatrix}\mathsf{fold}_{\tau}(\mathcal{M}_c, \mathcal{M}_q, \llbracket u_1, \ldots, u_n \rrbracket^{\eta,\rho}_{\mathbb{M}_{\mathsf{e}}}, \hat{S}, \eta, \rho_{\mathsf{a}}),\\ \mathsf{fold}_{\tau}(\mathcal{M}_c, \mathcal{M}_q, \llbracket u_1, \ldots, u_n \rrbracket^{\eta,\rho}_{\mathbb{M}}, S, \eta, \rho_{\mathsf{a}})\end{pmatrix} \tag{13}$$

$$\mathsf{d}_{\mathsf{stat}}\begin{pmatrix}\mathsf{fold}_{\tau}(\mathcal{M}_c, \mathcal{M}_q, \llbracket u_{n+1}, \ldots, u_{2\,n} \rrbracket^{\eta,\rho}_{\mathbb{M}}, S, \eta, \rho_{\mathsf{a}}),\\ \mathsf{fold}_{\tau}(\mathcal{M}_c, \mathcal{M}_q, \llbracket u_{n+1}, \ldots, u_{2\,n} \rrbracket^{\eta,\rho}_{\mathbb{M}_{\mathsf{e}}}, \hat{S}, \eta, \rho_{\mathsf{a}})\end{pmatrix} \tag{14}$$

Eq. (13) and Eq. (14) have identical proof, we thus focus on the former.

Proof of Eq. (13). First, let $v_1, \ldots, v_l$ be the order 0 terms in $u_1, \ldots, u_n$, and $w_1, \ldots, w_k$ be the order 1 terms. Without loss of generality, we can assume that $\mathcal{E}$ contains two special symbols tape_a and tape_h such that, in any extended model $\mathbb{M}_0 : \mathcal{E}$,

$$\llbracket \mathsf{tape}_a \rrbracket^{\eta,\rho}_{\mathbb{M}_0} = \rho_{\mathsf{a}} \qquad \llbracket \mathsf{tape}_h \rrbracket^{\eta,\rho}_{\mathbb{M}_0} = \rho_{\mathsf{h}}[\mathsf{qrnd} \mapsto 0]$$

where $\rho_{\mathsf{h}}[\mathsf{qrnd} \mapsto 0]$ is the tape ρ_{h} in which the randomness related to qrnd has been masked. Clearly, tape_a and tape_h are terms satisfying the conditions of Proposition 8 (e.g. for the former, we only need the continuation $k\ i$, which we can take to be $k\ i = \mathsf{tape}_a$, and we can set $(v\ i, v_q\ i) = (0, \mathsf{witness})$).

It is easy to check that the tuple $(v_1, \ldots, v_l, \mathsf{tape}_a, \mathsf{tape}_h)$ satisfies the hypotheses of Proposition 8 since each subterm satisfies these hypotheses. Thus, we know that:

$$\mathsf{d}_{\mathsf{stat}}\begin{pmatrix}\llbracket v_1, \ldots, v_l, \mathsf{tape}_a, \mathsf{tape}_h \rrbracket_{\mathbb{M}},\\ \llbracket v_1, \ldots, v_l, \mathsf{tape}_a, \mathsf{tape}_h \rrbracket_{\mathbb{M}_{\mathsf{e}}}\end{pmatrix} \in \mathsf{negl}(\eta)$$

(Using the fact that $r^{\eta}_{\mathbb{M}}$ is larger than η.)
By definitions of tape_a and tape_h, this yields that:

$$\mathsf{d}_{\mathsf{stat}}\begin{pmatrix}(\llbracket v_1, \ldots, v_l \rrbracket^{\rho,\eta}_{\mathbb{M}_{\mathsf{e}}}, \rho_{\mathsf{a}}, \rho_{\mathsf{h}}[\mathsf{qrnd} \mapsto 0]),\\ (\llbracket v_1, \ldots, v_l \rrbracket^{\rho,\eta}_{\mathbb{M}}, \rho_{\mathsf{a}}, \rho_{\mathsf{h}}[\mathsf{qrnd} \mapsto 0])\end{pmatrix} \in \mathsf{negl}(\eta) \tag{15}$$

Then, we observe that thanks to Eq. (11), we have:

$$\llbracket (w_1, \ldots, w_k) \rrbracket^{\rho,\eta}_{\mathbb{M}} = \llbracket (w_1, \ldots, w_k) \rrbracket^{\rho[\mathsf{qrnd} \mapsto 0],\eta}_{\mathbb{M}}$$

where $\rho[\mathsf{qrnd} \mapsto 0] = (\rho_{\mathsf{a}}, \rho_{\mathsf{h}}[\mathsf{qrnd} \mapsto 0])$. Thus:

$$f(\llbracket v_1, \ldots, v_l \rrbracket_{\mathbb{M}}, \rho_{\mathsf{a}}, \rho_{\mathsf{h}}) = \left(\llbracket (u_1, \ldots, u_l, w_1, \ldots, w_k) \rrbracket^{\rho,\eta}_{\mathbb{M}}, \rho_{\mathsf{a}}\right)$$

is well-defined. Further, this is a measurable function (equipping its domain and co-domain with the discrete σ-algebra). Consequently, by applying Proposition 5 to Eq. (15), we get that:

$$\mathsf{d}_{\mathsf{stat}}\begin{pmatrix}(\llbracket v_1, \ldots, v_l, w_1, \ldots, w_k \rrbracket_{\mathbb{M}_{\mathsf{e}}}, \rho_{\mathsf{a}}),\\ (\llbracket v_1, \ldots, v_l, w_1, \ldots, w_k \rrbracket_{\mathbb{M}}, \rho_{\mathsf{a}})\end{pmatrix} \in \mathsf{negl}(\eta).$$

Since $v_1, \ldots, v_l, w_1, \ldots, w_k$ is a permutation of $u_1, \ldots, u_n$, and by adjoining a value S sampled independently in $\llbracket \mathsf{qrand} \rrbracket^{\eta}_{\mathbb{M}})^{R(\eta)}$:

$$\mathsf{d}_{\mathsf{stat}}\begin{pmatrix}(\llbracket u_1, \ldots, u_n \rrbracket_{\mathbb{M}_{\mathsf{e}}}, \rho_{\mathsf{a}}, S),\\ (\llbracket u_1, \ldots, u_n \rrbracket_{\mathbb{M}}, \rho_{\mathsf{a}}, S)\end{pmatrix} \in \mathsf{negl}(\eta).$$

Applying again Proposition 5 to the above:

$$\mathsf{d}_{\mathsf{stat}}\begin{pmatrix}\mathsf{fold}_{\tau}(\mathcal{M}_c, \mathcal{M}_q, \llbracket u_1, \ldots, u_n \rrbracket^{\eta,\rho}_{\mathbb{M}_{\mathsf{e}}}, S, \eta, \rho_{\mathsf{a}}),\\ \mathsf{fold}_{\tau}(\mathcal{M}_c, \mathcal{M}_q, \llbracket u_1, \ldots, u_n \rrbracket^{\eta,\rho}_{\mathbb{M}}, S, \eta, \rho_{\mathsf{a}})\end{pmatrix} \in \mathsf{negl}(\eta). \tag{16}$$

Further, thanks to Proposition 9, we have that:

$$\mathsf{d}_{\mathsf{stat}}\begin{pmatrix}\mathsf{fold}_{\tau}(\mathcal{M}_c, \mathcal{M}_q, \llbracket u_1, \ldots, u_n \rrbracket^{\eta,\rho}_{\mathbb{M}_{\mathsf{e}}}, S, \eta, \rho_{\mathsf{a}}),\\ \mathsf{fold}_{\tau}(\mathcal{M}_c, \mathcal{M}_q, \llbracket u_1, \ldots, u_n \rrbracket^{\eta,\rho}_{\mathbb{M}_{\mathsf{e}}}, \hat{S}, \eta, \rho_{\mathsf{a}})\end{pmatrix} \in \mathsf{negl}(\eta). \tag{17}$$

where $\hat{S}$ is sampled in $(\{0,1\}^{\omega})^{R(\eta)}$.

Finally, using the triangular inequality, Eq. (13) follows from Eq. (16) and Eq. (17), which concludes this proof. □

D.4 Main Adequacy Theorem

We are now ready to state and prove our main adequacy theorem, which states that the approximated and exact semantics coincide on a subset of global formulas, which we call *well-formed*.

Definition 5. *A global formula is said to be well-formed if its lies in the fragment:*

$$\begin{aligned} F_{\mathsf{e}} \ ::= \ & F_{\mathsf{e}} \,\tilde{\vee}\, F_{\mathsf{e}} \mid F_{\mathsf{e}} \,\tilde{\wedge}\, F_{\mathsf{e}} \mid \tilde{\bot} \mid \tilde{\neg} F_{\mathsf{e}} \\ \mid \ & \tilde{\forall} x.\, \mathsf{const}(x) \tilde{\Rightarrow} F_{\mathsf{e}} \mid \tilde{\exists} x.\, \mathsf{const}(x) \tilde{\wedge} F_{\mathsf{e}} \\ \mid \ & [t]_{\mathsf{e}} \mid \mathsf{det}(t) \mid \mathsf{const}(t) \mid \mathsf{qrand}(t;t) \mid \mathsf{qadv}(t;t) \mid [t] \mid \vec{t} \sim \vec{t} \end{aligned}$$

where all the terms appearing in the overwhelming truth $[\cdot]$ and equivalence predicates $\sim$ must be well-formed (c.f. Definition 4), and where all terms appearing in the predicates: $[\cdot]_{\mathsf{e}}$, $\mathsf{det}(\cdot)$ and $\mathsf{const}(\cdot)$ must have a finite type.

Remark that we do not require anything of the terms appearing in the other predicates. E.g. a well-formed global formula F_{e} may contain $\mathsf{qrand}(t;t')$ as sub-formula, for *any* term t. Further, the restrictions requiring that quantification are over variables satisfying $\mathsf{const}(\cdot)$ could be modified to use any other predicate that ensures

that the interpretation of the quantified variable x is independent of the $\mathsf{r}^\eta_{\mathbb{M}}$ component of the random tapes.

Definition 6. *In Theorem 1, we say that a model $\mathbb{M}$ is such that $r^\eta_{\mathbb{M}}$ is large enough for a well-formed global formula F_e if $2^{r^\eta_{\mathbb{M}}}$ asymptotically dominates the cardinal of the interpretation of the type τ of any term appearing at the top-level in the predicates $[\cdot]_e$, $\mathsf{det}(\cdot)$ and $\mathsf{const}(\cdot)$. Furthermore, we require that $r^\eta_{\mathbb{M}}$ is at-least η.*

More formally, for all t such that $[t]_e$, $\mathsf{det}(t)$, or $\mathsf{const}(t)$ appears in F_e, if we let τ be the type of t then we require that for all η:

$$\frac{|[\![\tau]\!]^\eta_{\mathbb{M}}|}{2^{r^\eta_{\mathbb{M}}}} \underset{\eta\to+\infty}{\longrightarrow} 0. \qquad \text{and} \qquad r^\eta_{\mathbb{M}} \geq \eta.$$

We now recall and prove our main adequacy result Theorem 1.

Theorem 1. *Let $\mathbb{M}$ be a model, $\mathbb{M}_e$ the corresponding exact model, and F_e a well-formed global formula. If $r^\eta_{\mathbb{M}}$ is large enough then:*

$$\mathbb{M} \models F_e \quad \textit{iff.} \quad \mathbb{M}_e \models F_e.$$

Proof. We prove this by structural induction on the formula F_e.

- This is trivial for the false atom $\tilde{\bot}$.
- For the boolean connective $\tilde{\vee}$, $\tilde{\wedge}$, $\tilde{\neg}$, this immediately follows from the induction hypothesis. For example,

$$\begin{aligned} \mathbb{M} \models \tilde{\neg} F_e \text{ iff. } & \mathbb{M} \not\models F_e \\ \text{iff. } & \mathbb{M}_e \not\models F_e \qquad \text{(by induction hypothesis)} \\ \text{iff. } & \mathbb{M}_e \models \tilde{\neg} F_e \end{aligned}$$

  Other cases are similar.
- For universal quantification, we observe that:

$$\mathbb{M} \models \tilde{\forall}(x:\tau).\, \mathsf{const}(x) \tilde{\Rightarrow} F_e \tag{18}$$

  if and only if,

$$\text{for any } a \in \textstyle\bigcap_\eta [\![\tau]\!]^\eta_{\mathbb{M}}, \quad \mathbb{M}[x \mapsto \mathbb{1}_a] \models F_e$$

  By induction hypothesis, this holds if and only if:

$$\text{for any } a \in \textstyle\bigcap_\eta [\![\tau]\!]^\eta_{\mathbb{M}}, \quad (\mathbb{M}[x \mapsto \mathbb{1}_a])^e \models F_e \tag{19}$$

  where $(\mathbb{M}[x \mapsto \mathbb{1}_a])^e$ is the exact model associated to the approximation model $\mathbb{M}[x \mapsto \mathbb{1}_a]$, and where $\mathbb{1}_a$ is such that $\mathbb{1}_a(\eta)(\rho) = a$ for any $\eta \in \mathbb{N}$ and $\rho \in \mathbb{T}_{\mathbb{M},\eta}$.
  Recall that an approximation model and its corresponding exact model only differ in: 1) their set of random tapes; 2) the interpretation of the att declared symbols; 3) the interpretation of defined symbols.
  Let $\mathbb{M}^e$ be the exact model associated to $\mathbb{M}$. We want that:

$$(\mathbb{M}[x \mapsto \mathbb{1}_a])^e = \mathbb{M}^e[x \mapsto \mathbb{1}_a], \tag{20}$$

  First, we observe that both models have the same underlying type structures, since type structures are identical in an approximation model and its corresponding exact model. Then, we observe that:
  - The factor $\mathsf{r}^\eta_{\mathbb{M}}$, and the interpretation of att, are identical in the exact models $(\mathbb{M}[x \mapsto \mathbb{1}_a])^e$ and $\mathbb{M}^e[x \mapsto \mathbb{1}_a]$, since switching to an exact model from $\mathbb{M}$ or $\mathbb{M}[x \mapsto \mathbb{1}_a]$ has the same impact on both $\mathsf{r}^\eta_{\mathbb{M}}$ and att.
  - All *declared* symbols different from x and att have identical interpretation in both models, as they are left invariant by an exact/approximation model switch, and by the extension of a model with a new binding $[x \mapsto \cdot]$.
  - x has the same interpretation $\mathbb{1}_a$ in both models. Actually, this would hold for any binding $x \mapsto A$ where A is such that its interpretation does not depend on the quantum component of the tapes (i.e. the part of the tape corresponding to $\mathsf{r}^\eta_{\mathbb{M}}$).
  - All *defined* symbols have identical interpretation too, since their semantics only depend on the existing declared symbols and the set of tapes, which we show identical in the points above.

  Consequently, we proved Eq. (20), and we know that $(\mathbb{M}[x \mapsto \mathbb{1}_a])^e$ is equal to the model $\mathbb{M}^e[x \mapsto \mathbb{1}_a]$. Thus, continuing from Eq. (19), we know that Eq. (18) holds if and only if

$$\text{for any } a \in \textstyle\bigcap_\eta [\![\tau]\!]^\eta_{\mathbb{M}}, \quad \mathbb{M}^e[x \mapsto \mathbb{1}_a] \models F_e.$$

  By definition, this holds if and only if:

$$\mathbb{M}^e \models \tilde{\forall}(x:\tau).\, \mathsf{const}(x) \tilde{\Rightarrow} F_e,$$

  which concludes this case.
- For existential quantification, we use the fact that $\tilde{\exists}$ is equivalent to $\tilde{\neg}\tilde{\forall}$.
- For $[t]_e$, $\mathsf{det}(t)$ and $\mathsf{const}(t)$, we observe that since $\mathsf{r}^\eta_{\mathbb{M}}$ is large enough for F_e (c.f. Definition 6), we know thanks to Remark 1 that:

$$\mathsf{supp}([\![t]\!]_{\mathbb{M}}) = \mathsf{supp}([\![t]\!]_{\mathbb{M}_e}).$$

  Thus, the family of random variables $[\![t]\!]_{\mathbb{M}_e}$ is, resp., always true, deterministic or constant iff. the family of random variables $[\![t]\!]_{\mathbb{M}}$ is, resp., always true, deterministic or constant.
- For $\mathsf{qrand}(t;t)$, we observe that $[\![t]\!]_{\mathbb{M}}$ and $[\![t]\!]_{\mathbb{M}_e}$ access exactly the same part of the random tapes.
- For $\mathsf{qadv}(t;t_s)$, this follows from the fact that the finalization function Measure is changed in the same way in the semantics of a term and the semantics of the predicate $\mathsf{qadv}(\cdot;\cdot)$. Thus, we can reuse the machine provided by $\mathsf{qadv}(\cdot;\cdot)$ in $\mathbb{M}$ to show that the predicate also holds in $\mathbb{M}_e$, and conversely.
- For $\vec{u} \sim \vec{v}$, we use Proposition 10.
- For $[t]$, we directly use Proposition 8. □

Corollary 1. *Let F_e be a well-formed global formula. If:*

$$\models F_e \quad \textit{then} \quad \mathbb{M}_e \models F_e \quad \textit{for any exact model } \mathbb{M}_e.$$

Proof. Let F_e be a well-formed global formula. By hypothesis, we have $\models F_e$, i.e. F_e is satisfied by any approximation model. Take an exact model $\mathbb{M}_e$, and let $\mathbb{M}$ be an approximation model corresponding to $\mathbb{M}_e$ such that $\mathsf{r}^\eta_{\mathbb{M}}$ is large enough for Theorem 1 to apply. Thus, $\mathbb{M}_e \models F_e$. □

E Leveraging Squirrel's Proof System

We start by providing in Figure 7 a proof system for our quantum simulation predicate $\mathsf{qadv}(\cdot;\cdot)$ which plays a crucial role in our proof system.

$$\textsc{Ext}\ \frac{\mathrm{qadv}_X(t;t_S) \quad [t_S \cap t'_S = \emptyset]_{\mathrm{e}}}{\mathrm{qadv}_X(t;t_S \cup t'_S)} \qquad \textsc{Ext'}\ \frac{\mathrm{qadv}_X(t;t_S) \quad X \subseteq X'}{\mathrm{qadv}_{X'}(t;t_S)}$$

$$\textsc{Q-Att}\ \frac{}{\mathrm{qadv}_{\{\mathrm{a}\}}(\lambda x.\ \mathrm{att}(\mathrm{qrnd}\ t, x); t)} \qquad \textsc{FA:App:Indep}\ \frac{\mathrm{qadv}_X(t;t_S) \qquad \mathrm{qadv}_X(t';t'_S) \quad [t_S \cap t'_S = \emptyset]_{\mathrm{e}}}{\mathrm{qadv}_X(t\ t';t'_S \cup t_S).}$$

$$\textsc{Adv}\ \frac{\mathrm{adv}(t)}{\mathrm{qadv}_{\{\mathrm{a}\}}(t;\emptyset)}$$

Figure 7: Our quantum simulation rules.

We denote $\subseteq, \cap, \cup$, and $\emptyset$ the function symbols on the type timestamp set representing, resp., set inclusion, intersection, union, and the empty set.

Theorem 3. *The rules given in Figure 7 are sound.*

Proof. We prove each rule one by one:

- Rule Ext: Let $\mathcal{M}_c, \mathcal{M}_q$ be the machines coming from the validity of $\mathrm{qadv}(t;t_S)$ with some ordering $\prec$ over $[\![t_S]\!]^{\eta,\rho}_{\mathbb{M}}$, we need to build $\mathcal{M}'_c, \mathcal{M}'_q$ and $\prec'$ for $\mathrm{qadv}(t;t_S \cup t'_S)$. We set $\prec'$ as $\prec$, but which extends to $[\![t'_s]\!]^{\eta,\rho}_{\mathbb{M}}$ by putting all its elements after the ones $[\![t_s]\!]^{\eta,\rho}_{\mathbb{M}}$ (in any arbitrary order). $\mathcal{M}'_x$ simulates $\mathcal{M}_x$, but during the fold_τ operation, but once all elements of $[\![t_s]\!]^{\eta,\rho}_{\mathbb{M}}$ have been consumed, it instantly returns.
- Rule Q-Att: This is by hypothesis, as we always require att to be computable by a polynomial-time quantum machine.
- Rule Ext': If we can simulate with some randomness, we can simulate with more randomness.
- Rule Adv: This is by the axiom of Section 4.1.4.
- Rule FA:App:Indep: Let $\mathcal{M}_c, \mathcal{M}_q$ be the machines coming from $\mathrm{qadv}(t;t_S)$ with order $\prec$. Let $\mathcal{M}'_c, \mathcal{M}'_q$ be the machines coming from $\mathrm{qadv}(t';t'_S)$ with order $\prec'$. We simply first simulate the second one, and then the first one, by constructing $\prec''$ as the union of $\prec$ and $\prec'$, with in addition the fact that all elements in $[\![t_S]\!]^{\eta,\rho}_{\mathbb{M}}$ comes before the ones in $[\![t'_S]\!]^{\eta,\rho}_{\mathbb{M}}$. □

E.1 Core Logical Rules

As already said, thanks to our general approach which fully embeds the quantum values inside the logic, we directly inherited these rules from [10].

E.2 Quantum Indistinguishability Rules

We first define the $\mathrm{qrand}(\cdot;\cdot)$ and $\mathrm{polylen}(\cdot)$ predicates and present our proof system for $\sim$.

Quantum randomness usage predicate. Re-using the notion of generalized subterms $\mathcal{ST}_{\mathcal{E}}(u)$ from [9, Figure 8], and if $\mathbb{M}$ is a model of $\mathcal{E}$, we let $\mathbb{M} \models \mathrm{qrand}(u;t_S)$ iff. for all $(\vec{\alpha}, \phi, \mathrm{att}(\mathrm{r}, v)) \in \mathcal{ST}_{\mathcal{E}}(u)$, r is of the form $(\mathrm{qrnd}\ t)$ and $\mathbb{M} \models [\forall \vec{\alpha}.\ \phi \longrightarrow t \subseteq t_S]_{\mathrm{e}}$

Poly-length predicate. For any term t, we define the global predicate $\mathrm{polylen}(t)$ such that for any model $\mathbb{M}$:

- if t is order 0, $\mathbb{M} \models \mathrm{polylen}(t)$ iff. $\exists P$ polynomial. $\forall \eta \in \mathbb{N}. \forall \rho \in \mathbb{T}_{\mathbb{M},\eta}. |[\![t]\!]^{\eta,\rho}_{\mathbb{M}}| \leq P(\eta)$
- if t is order 1, $\mathbb{M} \models \mathrm{polylen}(t)$ iff. $\exists P$ polynomial. $\forall \eta \in \mathbb{N}. \forall \rho \in \mathbb{T}_{\mathbb{M},\eta}. \forall \mathbf{a}, |[\![t]\!]^{\eta,\rho}_{\mathbb{M}}(\mathbf{a})| \leq P(\eta + |\mathbf{a}|)$

Rules for quantum indistinguishability. We now present our proof system for $\sim$.

Theorem 4. *The rules of Figure 8 are sound.*

Proof. We prove all rules of Figure 8:

- Rule FA:App: In the classical setting, we simply follow the proof of [10]. That is, given a distinguisher $\mathcal{D}$ over the conclusion, we build a distinguisher over the premise that receives $\vec{w}, f, t$, calls the oracle for f over t, then simulates $\mathcal{D}$ over $\vec{w}, u$ where u is the results. We know that u is of polynomial length and can safely be copied from the oracle output tape to the working tape thanks to the polylen premises.
- Rule FA:App-Q: Consider the previous proof. It directly maps to a similar construction when the result of f is quantum.
- Rule FA:Adv-0: Given a distinguisher $\mathcal{D}$ against the conclusion, we build a distinguisher over the premise that only receives $\vec{w}$. We rely on the two machines $\mathcal{M}_c\ \mathcal{M}_q$ that allow to simulate t with the computation
$$C_{t_S} := \mathrm{fold}_\tau\left(\mathcal{M}_c, \mathcal{M}_q, 0, ([\![\mathrm{qrnd}\ \tau]\!]^{\eta,\rho}_{\mathbb{M}} \mid \tau \in [\![t_S]\!]^{\eta,\rho}_{\mathbb{M}}), \eta, \rho_{\mathrm{a}}\right).$$
Based on the definition of fold_τ, note that for any $t_{\mathcal{D}}$ such that it has the same length as t_S, the distribution produced by C_{t_S} and $C_{t_{\mathcal{D}}}$ are equal. Further, as we know that $\vec{w}$ does not have any probabilistic dependencies with the lists $([\![\mathrm{qrnd}\ \tau]\!]^{\eta,\rho}_{\mathbb{M}} \mid \tau \in [\![t_S]\!]^{\eta,\rho}_{\mathbb{M}})$ thanks to the additional premises, we even have that the joint distributions corresponding to $[\![\vec{w}]\!]^{\eta,\rho}_{\mathbb{M}}, C_{t_S}$ and $[\![\vec{w}]\!]^{\eta,\rho}_{\mathbb{M}}, C_{t_{\mathcal{D}}}$ are equal. As we can trivially construct a $t_{\mathcal{D}}$ that has the same length as t_S and only uses the randomness accessible to $\mathcal{D}$, we can simulate $C_{t_{\mathcal{D}}}$ and then pass $\vec{w}$ and its result to $\mathcal{D}$, the whole simulation yielding a valid simulator against $\vec{u} \sim \vec{v}$.
- Rule FA:Adv-1: Given a distinguisher on the conclusion, we use the simulator for t to answer all its oracle queries for t. As the simulator does not need any measurements, it is trivially added to the computations.
- Rule FA:Single: Similar to FA:Adv-0.
- Rule FA:Dup: The new distinguisher duplicates the last input. Note that this is not possible for a quantum value.
- Rule FA:Witness: The top distinguisher simulates the bottom distinguisher, by extending its inputs with ϵ (whose quantum lifting $|\epsilon\rangle$ is exactly the value of witness). □

E.3 Bi-Deduction

We now formally define our novel notion of quantum bi-deduction. The inputs $\vec{u}_0$ and $\vec{u}_1$ are two sequences of terms of the same length with compatible types, i.e. such that for all i, the i-th terms in $\vec{u}_0$ and $\vec{u}_1$ have the same type. We require that $\vec{u}_0, \vec{u}_1$ only contain terms of type τ where τ is either of classical order at-most one, or of quantum order zero. We make the same assumptions for the

FA:App

$$\frac{\mathcal{E};\Theta \vdash \mathsf{polylen}(f) \mathbin{\tilde{\wedge}} \mathsf{polylen}(f') \qquad \mathcal{E};\Theta \vdash \vec{u}, f, t \sim \vec{v}, f', t'}{\mathcal{E};\Theta \vdash \vec{u}, (f\ t) \sim \vec{v}, (f'\ t')} \text{ for } f \text{ of classical order 1}$$

FA:App-Q

$$\frac{\mathcal{E};\Theta \vdash \mathsf{polylen}(f) \mathbin{\tilde{\wedge}} \mathsf{polylen}(f') \qquad \mathcal{E};\Theta \vdash \vec{u}, \mathfrak{q}(f), t \sim \vec{v}, \mathfrak{q}(f'), t'}{\mathcal{E};\Theta \vdash \vec{u}, (f\ t) \sim \vec{v}, (f'\ t')} \text{ for } f \text{ of order 1 with a quantum output type and classical input type}$$

FA:Adv-0

$$\frac{t \text{ of order } 0 \qquad \mathcal{E};\Theta \vdash [t_S \cap t_{S'} = \emptyset]_{\mathsf{e}} \qquad \mathcal{E};\Theta \vdash \mathsf{qadv}_{\mathsf{P}}([;\{\}\mathsf{a}\}]tt_S \qquad \mathcal{E};\Theta \vdash \vec{u} \sim \vec{v} \qquad \mathcal{E};\Theta \vdash \mathsf{qrand}(\vec{u}; t_{S'}) \mathbin{\tilde{\wedge}} \mathsf{qrand}(\vec{v}; t_{S'})}{\mathcal{E};\Theta \vdash \vec{u}, t \sim \vec{v}, t}$$

FA:Adv-1

$$\frac{t \text{ of order } 1 \qquad \mathcal{E};\Theta \vdash \mathsf{qadv}_{\mathsf{P}}([;\{\}\mathsf{a}\}]t\emptyset \qquad \mathcal{E};\Theta \vdash \vec{u} \sim \vec{v}}{\mathcal{E};\Theta \vdash \vec{u}, t \sim \vec{v}, t}$$

FA:Single

$$\frac{\mathcal{E};\Theta \vdash [t_S \cap t_{S'} = \emptyset]_{\mathsf{e}} \qquad \mathcal{E};\Theta \vdash \mathsf{qadv}_{\mathsf{P}}([;\{\}\mathsf{a}\}]ft_S \qquad \mathcal{E};\Theta \vdash \vec{u}, t_0 \sim \vec{v}, t_1 \qquad \mathcal{E};\Theta \vdash \mathsf{qrand}(\vec{u}, t_0; t_{S'}) \mathbin{\tilde{\wedge}} \mathsf{qrand}(\vec{v}, t_1; t_{S'})}{\mathcal{E};\Theta \vdash \vec{u}, f\ t_0 \sim \vec{v}, f\ t_1} \text{ for } f \text{ a quantum or classical oracle}$$

FA:Dup

$$\frac{\mathcal{E};\Theta \vdash \vec{u}, t_0 \sim \vec{v}, t_1}{\mathcal{E};\Theta \vdash \vec{u}, t_0, t_0 \sim \vec{v}, t_1, t_1} \text{ for } t_0, t_1 \text{ of classical types and of any order}$$

FA:Witness

$$\frac{\mathcal{E};\Theta \vdash \vec{u} \sim \vec{v}}{\mathcal{E};\Theta \vdash \vec{u}, \mathsf{witness} \sim \vec{v}, \mathsf{witness}}$$

witness **is a symbol of type** $\mathsf{Hilb}_{\mathsf{msg}}$ **that is always equal to** $|\epsilon\rangle$**, where** ϵ **is the empty bit-string.**

Figure 8: Reasoning rules over $\sim$

outputs $\vec{v}_0, \vec{v}_1$. Further, we require that $\vec{v}_i$ contains *at-most one* term of quantum type. The semantics of $\triangleright$ is:

$$\mathbb{M} \models \vec{x} : \#(\vec{u}_0; \vec{u}_1) \triangleright \#(\vec{v}_0; \vec{v}_1) \qquad \text{iff.}$$

$$\mathbb{M} \models \tilde{\exists} f.\ \mathsf{qadv}_{\mathsf{E}}(f) \mathbin{\tilde{\wedge}} \tilde{\forall} \vec{x}.\ [f(\vec{x}, \vec{u}_0) = \vec{v}_0]_{\mathsf{e}} \mathbin{\tilde{\wedge}} [f(\vec{x}, \vec{u}_1) = \vec{v}_1]_{\mathsf{e}}.$$

In a predicate $\vec{x} : \vec{u} \triangleright \vec{v}$, the inputs $\vec{x}$ are *uniform* argument of the simulator, i.e. there must exists a *single* simulator that can compute $\vec{v}$ from the inputs $\vec{u}, \vec{x}$ for any value of $\vec{x}$.

Remark 8. *The classical bi-deduction predicate of [7] is of the form $\vec{u} \triangleright \vec{v}$, with several restrictions. First, the terms $\vec{u}$ and $\vec{v}$ must be of order-0, as this work predates the higher-order extension [10] that added supports for λ. Second, quantification is restricted to be over the types* timestamp *and* index*, which are finite and fixed types. Third, they require that $\vec{u}, \vec{v}$ are classical terms, since they do not target post-quantum cryptography. For the first two reasons mentioned above (excluding the restriction to classical terms), the predicate in [7] does not need to support uniform argument as we do. Remark that the need for uniform argument is unrelated to the extension to quantum values: it is needed because we allow order-1 classical term.*

As it is the case for classical bi-deduction, we have a rule:

Bideduce-$\triangleright$

$$\frac{\vec{u}_0 \sim \vec{u}_1 \qquad \emptyset : \#(\vec{u}_0; \vec{u}_1) \triangleright \#(\vec{v}_0; \vec{v}_1)}{\vec{v}_0 \sim \vec{v}_1}$$

linking quantum bi-deduction and quantum indistinguishability.

Theorem 5. *The rule* Bideduce-$\triangleright$ *is sound.*

Proof. Consider a winning distinguisher $\mathcal{B}$ against $\vec{v}_0 \sim \vec{v}_1$ (we denote as a single computation $\mathcal{B}$ a folding of classical and quantum Turing Machines). Given a model $\mathbb{M}$, $\mathcal{B}$ can distinguish whether it receives $[\![\vec{v}_0]\!]^{\eta,\rho}_{\mathbb{M}}$ or $[\![\vec{v}_1]\!]^{\eta,\rho}_{\mathbb{M}}$. Further, we know that we have a function f such that:

$$\mathbb{M} \models \mathsf{qadv}_{\mathsf{E}}(f) \mathbin{\tilde{\wedge}} [f(\vec{u}_0) = \vec{v}_0]_{\mathsf{e}} \mathbin{\tilde{\wedge}} [f(\vec{u}_1) = \vec{v}_1]_{\mathsf{e}}.$$

Then, on input $[\![\vec{u}_0]\!]^{\eta,\rho}_{\mathbb{M}}$ or $[\![\vec{u}_1]\!]^{\eta,\rho}_{\mathbb{M}}$, by first calling f, we get $[\![\vec{v}_0]\!]^{\eta,\rho}_{\mathbb{M}}$ or $[\![\vec{v}_1]\!]^{\eta,\rho}_{\mathbb{M}}$, that $\mathcal{B}$ can distinguish. Thus, the composition of f and $\mathcal{B}$ is a distinguisher against $\vec{u}_0 \sim \vec{u}_1$. Thanks to our definition of distinguishers using folding classical and quantum computations, the resulting machine is clearly quantum polynomial-time as it is the composition of $\mathcal{B}$, which satisfies an instance of $\mathsf{qadv}(t; t_s)$, with f, which is such that $\mathsf{qadv}_{\mathsf{E}}(f)$. □

We present a proof-system for quantum bi-deduction in Figure 9, which is an adaptation of the existing proof-systems for classical bi-deduction [5, 7].

Remark 9. *The rules implemented in* Squirrel *bi-deduction engine are slightly more complicated than the rules in Figure 9, as they support* conditional *inputs and outputs. A conditional term $(t \mid f)$ is a term that is guarded by a boolean condition f: formally, $(t \mid f)$ is syntactic sugar for $(f, \text{if } f \text{ then } t \text{ else } \bot)$ where $\bot$ is an arbitrary symbol of the same type as t.*

Theorem 6. *The rules of Figure 9 are sound.*

Proof. For each rule, we build the simulator justifying the conclusion from the simulator justifying the premise.

- BD.Split - Given functions $f_i(\vec{u}_c, \vec{u}^i_q)$, we directly define $f(\vec{u}_c, \vec{u}^1_q, ..., \vec{u}^1_q) = f_1(\vec{u}_c, \vec{u}^1_q), ..., f_n(\vec{u}_c, \vec{u}^n_q)$.
- BD.FA - Direct, as the set of functions such that $\mathsf{qadv}_{\mathsf{E}}(f)$ is closed under composition (thanks to the exact notion).

$$\text{BD.Split}\ \frac{\vec{x} : \vec{u_c}, \vec{u_q^1} \triangleright \vec{v_1} \quad \dots \quad \vec{x} : \vec{u_c}, \vec{u_q^n} \triangleright \vec{v_n}}{\vec{x} : \vec{u_c}, \vec{u_q^1}, \dots, \vec{x} : \vec{u_q^n}, \triangleright \vec{v_1}, \dots, \vec{v_n}} \qquad \text{BD.FA}\ \frac{\vec{x} : \vec{u} \triangleright \vec{v} \quad \mathsf{qadv}_{\mathsf{E}}(f)}{\vec{x} : \vec{u} \triangleright f(\vec{v})}$$

$$\text{BD.Trans}\ \frac{\vec{x} : \vec{u_c}, \vec{u_q^1} \triangleright \vec{w} \quad \vec{x} : \vec{u_c}, \vec{u_q^2}, \vec{w} \triangleright \vec{v}}{\vec{x} : \vec{u_c}, \vec{u_q^1}, \vec{u_q^2} \triangleright \vec{v}} \qquad \text{BD.Weak}\ \frac{\vec{x} : \vec{u} \triangleright \vec{w}}{\vec{x} : \vec{u}, \vec{v} \triangleright \vec{w}}$$

$$\text{BD.Rewrite}\ \frac{\vec{x} : \vec{u_0} \triangleright \vec{v} \quad \vec{x} \vdash [\vec{u_0} = \vec{u_1}]_{\mathsf{e}}}{\vec{x} : \vec{u_1} \triangleright v} \qquad \text{BD.Quant}\ \frac{\vec{x}, y : \vec{u_c} \triangleright v \quad Q \in \{\exists; \forall\} \quad \mathsf{enum}(\tau)}{\vec{x} : \vec{u_c} \triangleright Q(y : \tau).\, v}$$

$$\text{BD.Lambda}\ \frac{\vec{x}, y : \vec{u_c} \triangleright v}{\vec{x} : \vec{u_c} \triangleright \lambda(y : \tau).\, v} \qquad \text{BD.Ded}\ \frac{\vec{x} \vdash [w\{\vec{y} \mapsto \vec{a}\} = v]_{\mathsf{e}} \quad \vec{x} : \vec{u}, \lambda \vec{y}.w \triangleright \vec{a}}{\vec{x} : \vec{u}, \lambda \vec{y}.w \triangleright v}$$

Bold terms $\vec{u}, \vec{v}, \dots$ denotes pairs of sequences of terms $\#(\vec{u_0}; \vec{u_1}), \#(\vec{v_0}; \vec{v_1}), \dots$. In all the rules, $\vec{u_c}$ must be a sequence of *purely classical* terms. We factorize common behavior, e.g. we write $f(\#(\vec{u_0}; \vec{u_1}))$ instead of $\#(f(\vec{u_0}); f(\vec{u_1}))$. In BD.Quant and BD.Lambda, we require that y has been properly renamed, i.e. that $y \notin \vec{x}$. Recall that all terms are of order at-most one, and that order one terms are classical.

Figure 9: Proof-system for quantum bi-deduction.

- BD.Trans - Given functions $f_1(\vec{u_c}, \vec{u_q^1})$ and $f_2(\vec{u_c}, \vec{u_q^2}, \vec{w})$, we define $f(\vec{u_c}, \vec{u_q^1}, \vec{u_q^1}) = f_2(\vec{u_c}, \vec{u_q^2}, f_1(\vec{u_c}, \vec{u_q^1}))$.
- BD.Weak - This simply drops parts of its input.
- BD.Rewrite - By rewriting inside the definitions.
- BD.Quant - The simulator directly enumerates the possible values, and in essence replaces existential or universal quantification by a disjunction or a conjunction.
- BD.Lambda - The equality in the conclusion holds by functional extentionality. The quantum polynomial-time condition follows from the definition of $\mathsf{qadv}_{\mathsf{E}}(\cdot)$.
- BD.Ded - This rule follows from the purely classical semantics. □

We can easily generalize this system to allow tuples in the sequence of terms, and keeping the restriction that at most one element on the right-hand-side is quantum, to be understood as at most one element either in the sequence of terms or the tuple components is quantum.

E.4 Cryptographic Bi-Deduction

We are now ready to define how we do quantum cryptographic bi-deductions. We rely on the programming language of [11] to describe adversary, simulator and cryptographic games, that we only adapt in minor ways to make it capable of capturing the quantum simulators we are interested in. Our changes are only superficial (e.g. we allow variables to have quantum base types, when [11] only allow variables with base types). Crucially, we do not modify the semantics of this programming language. Thus, this language suffers from the issues of the logic: it allows to write programs representing unreasonable computations. E.g. the function

$$\begin{aligned} &f(q : \mathsf{Hilb}_{\mathsf{msg}}) = \{ \\ &\quad r_0 \xleftarrow{\$} \mathsf{qrand}; \\ &\quad r_1 \xleftarrow{\$} \mathsf{qrand}; \\ &\quad y_0 \leftarrow \mathsf{att}(r_0, q); \\ &\quad y_1 \leftarrow \mathsf{att}(r_1, q); \\ &\quad \textbf{return}\ (y_0, y_1) \\ &\} \end{aligned}$$

uses the quantum value q twice. Thus, it is clearly not quantum polynomial-time, even if $\mathsf{att}(\cdot, \cdot)$ is quantum polynomial-time. Yet, this function is a valid syntactic program with a well-defined semantics, even though this semantics cannot be computed by an efficient quantum machine. We address this issue as we did for the logic, by restricting ourselves to quantum polynomial-time program whenever needed.

Quantum imperative programming language. We do not describe the language syntax or semantics in detail here, and only present its main features (we refer the reader to [11] for details). We assume given a library $\mathcal{L}$ of function symbols, where each symbol $l \in \mathcal{L}$ comes with an associated type $\mathsf{type}(s)$. The library represent functions that are shared between the logic and our imperative programming language: we will only consider (logical) environment $\mathcal{E}$ and base axioms Θ that are *compatible* with the library $\mathcal{L}$, i.e. that are such that for any symbol $l \in \mathcal{L}$ with type τ, we have $\mathcal{E} \vdash l : \tau$ and $\mathcal{E}, \Theta \models \mathsf{adv}(l)$, or l is the att symbol. In [11], all library symbols $l \in \mathcal{L}$ are required to be classical polynomial-time: thus, we loosen this condition and also allow for the quantum polynomial-time function symbol att.

We assume a set of *program variables* $\mathcal{X}_p$, where each variable x comes with a type $\mathsf{type}(x) \in \mathbb{B} \cup \mathsf{Hilb}_{\mathbb{B}}$, which restrict program variables to base types or quantum base types, ensuring that they can be encoded as bit-strings — here again, [11] only allowed for classical base types. We consider well-typed expressions built on top of $\mathcal{X}_p$, $\mathcal{L}$, and two special symbols: b of type bool used to indicate if we are considering the left or right variant of a game; η of type int which always contain the security parameter. We use a standard imperative first-order programming language, whose syntax is given below:

$$\begin{aligned} \mathsf{p} \overset{\text{def}}{=}\ & v \leftarrow e && \textit{(assignment)} \\ \mid\ & \mathsf{p}_1; \mathsf{p}_2 && \textit{(sequence)} \\ \mid\ & \textbf{if}\ e\ \textbf{then}\ \mathsf{p}_1\ \textbf{else}\ \mathsf{p}_2 && \textit{(conditional branching)} \\ \mid\ & \textbf{while}(e)\{\mathsf{p}\} && \textit{(while loop)} \\ \mid\ & \textbf{abort} && \textit{(abort)} \\ \mid\ & \textbf{skip} && \textit{(no operation)} \\ \mid\ & v \xleftarrow{\$} \mathsf{T}[e] && \textit{(random sampling)} \\ \mid\ & v \leftarrow O_f(\vec{e})[\vec{e_g}; \vec{e_l}] && \textit{(oracle call)} \end{aligned}$$

Most of the constructs are standard, but for the last two cases which deal with random samplings and oracle calls. Before describing them, it will be useful to explain how probabilities are handled in this language. This language relies on an *eager probabilistic semantics*, in the sens that all randomness is sampled *before* the program starts its execution and stored into a program random tape $\mathfrak{p}$ — concretely, $\mathfrak{p} = (\mathfrak{p}_{|\mathsf{T}})_{\mathsf{T}\in\mathcal{T}}$ is a finite collection indexed by $\mathsf{T} \in \mathcal{T}$ of sub-tapes $\mathfrak{p}_{|\mathsf{T}}$, where each sub-tape is a finite array of random bits (all sampled independently uniformly at random). Then, whenever a program wants to do a random sampling, it simply reads some random bits from $\mathfrak{p}$.

More precisely, if v is an expression with base type τ, then $v \xleftarrow{\$} \mathsf{T}[e]$ sampled a value of type τ (using the same distribution than for names of type τ), where the necessary randomness is extracted from the sub-tape with tag T at offset e. The statement $v \leftarrow O_f(\vec{e})[\vec{e_g}; \vec{e_l}]$ calls the oracle f of the game $\mathcal{G}$ on inputs $\vec{e}$, using $\vec{e_g}$ as offsets for the global randomness, and $\vec{e_l}$ as offset for the local randomness. We refer the reader to [11] for a detailed description and the semantics of random samplings and oracle calls. As mention earlier, we make no modifications to the semantics of [11].

Cryptographic games. A *game* $\mathcal{G}$ is of the form:

```
game G = {
 var r_0 <-$ τ_0; ... var r_n <-$ τ_n;
 var x_0 <- e_1; ... var x_m <- e_m;
 oracle f_1 (...) = body_1
 ...
 oracle f_l (...) = body_l
}
```

where $\mathbf{var}\ \mathrm{r}_i \xleftarrow{\$} \tau_i$ declares a new randomly sampled variable of type τ_i, $\mathbf{var}\ \mathrm{x}_i \leftarrow \mathrm{e}_i$ declares a new variable initialized to e_i, and each oracle body body_i is of the form:

$$\mathbf{var}\ \mathrm{r'}_0 \xleftarrow{\$} \tau_0; \ldots\ \mathbf{var}\ \mathrm{r'}_k \xleftarrow{\$} \tau_n; \mathrm{p}; \mathbf{return}\ \mathrm{e}$$

and consists in a sequence of local random variable declarations, a body p, and a returned value e.

For any game $\mathcal{G}$, we let $\mathcal{G}.\mathrm{glob}$, $\mathcal{G}.\mathrm{glob}_\$$ and $\mathcal{G}.\mathrm{oracles}$ be, resp., the set of global variables, global randomness and oracles of $\mathcal{G}$.

An *adversary* against a game $\mathcal{G}$ is a well-behaved programs that

i) does not access $\mathcal{G}$'s internal variables or randomness,
ii) does not directly read the special bit b indicating which side of the game is being executed,
iii) properly feed randomness offsets to oracle calls (i.e. local offsets are fresh, and global offsets are consistent across oracle calls),
iv) linearly use quantum variables.

Only condition iv) is novel, and needed to deal with our extension to quantum adversaries. We refer the reader to [11, Appendix B.6] for more details on conditions i) to iii).

Safe quantum API. We are now ready to introduce our novel notion of safe quantum API.

Definition 7. *A* safe quantum API $\mathcal{Q}$ *is a cryptographic game such that* $\mathcal{S}^{\mathcal{Q}}$ *is a quantum polynomial-time program for any classical polynomial-time program* $\mathcal{S}$.

Formally designing an effective syntactic characterization of this property is out-of-scope of this paper, but the informal characterization below will be sufficient for our needs. To check that a game is a safe quantum API, it is sufficient to verify that: i) all its oracle have only classical inputs and outputs; ii) it stores a finite number of quantum states, which are subject to a linear usage condition ensuring that a state is never used more than once; iii) all its oracle uses only quantum polynomial-time library functions.

Proposition 11. *The API in Figure 5 is a safe quantum API.*

Quantum cryptographic bi-deduction. We recall that $\mu^{i\ \eta,\mathfrak{p}}_{\mathrm{init}\,\mathbb{M}}$ is the initial memory of the game $\mathcal{G}$ (see [11]) on side $i \in \{0,1\}$ using the model $\mathbb{M}$ to interpret library function symbols and the tape $\mathfrak{p}$ to provide randomness to the adversarial function symbols.

Definition 8. *A bi-assertion* $\boldsymbol{\phi}$ *is a* valid initial memory *for a game* $\mathcal{G}$ *w.r.t. an environment* $\mathcal{E}$ *and a set of global formulas* Θ *if for any* $\mathbb{M} : \mathcal{E}$ *satisfying* Θ*, for any* $i \in \{0,1\}$*,* η *and* ρ*, we have:*

$$\mathbb{M}, \eta, \rho, \mu^{i\ \eta,\mathfrak{p}}_{init\mathbb{M}} \models^{A} \boldsymbol{\phi}$$

(where $\mathfrak{p}$ *is the tape* ρ *filled with zero but for the sub-tape* $(\mathrm{tag}_A, \mathrm{bool})$ *which is set to* ρ_{a}*).*

We use bi-deduction judgements of the following shape:

$$\mathcal{E}; \Theta; \boldsymbol{C}; (\boldsymbol{\phi}, \boldsymbol{\psi}) \vdash \vec{\boldsymbol{u}} \triangleright_{\mathcal{G}} \vec{\boldsymbol{v}}$$

We make no changes to the syntax or semantics of bi-deduction judgement w.r.t. [11], except that the game $\mathcal{G}$ uses our slightly modified programming language that can describe (some) quantum computations. Crucially, we do *not* allow the simulator $\mathcal{S}$ witnessing the bi-deduction judgement to be a quantum polynomial-time adversary: as in [11], the simulator must be a classical adversary. Instead, we synthesize quantum simulators by wrapping all quantum computations in a safe quantum API $\mathcal{Q}$, allowing the classical simulator $\mathcal{S}$ to manipulate quantum values safely (e.g. $\mathcal{S}$ cannot clone a quantum value). To do so, we will need to extend an existing game with additional variables and oracles.

Definition 9. *The* composition $\mathcal{G}_0 \cdot \mathcal{G}_1$ *of two games* $\mathcal{G}_0$ *and* $\mathcal{G}_1$ *is the game obtained by concatenating all declarations of* $\mathcal{G}_0$ *and* $\mathcal{G}_1$*. It is* well-defined *whenever the two games share no oracle names, no global variable nor global random variables:*

$$\mathcal{G}_0.\mathrm{oracles} \cap \mathcal{G}_1.\mathrm{oracles} = \emptyset$$
$$(\mathcal{G}_0.\mathrm{glob} \cup \mathcal{G}_0.\mathrm{glob}_\$) \cap (\mathcal{G}_1.\mathrm{glob} \cup \mathcal{G}_1.\mathrm{glob}_\$) = \emptyset$$

Example 9. *For example, consider the games below:*

```
game G_0 = {
 var x <-$ τ;
 oracle f (...) = { p_f }
}

game G_2 = {
 var y <-$;
 oracle f (...) = { }
}
```

```
game G_1 = {
 var y = 0;
 oracle g (...) = { p_g }
}

game G_01 = {
 var x <-$ τ;
 var y = 0;
 oracle f (...) = { p_f }
 oracle g (...) = { p_g }
}
```

The game $\mathcal{G}_{01}$ *is the composition of* $\mathcal{G}_0$ *and* $\mathcal{G}_1$*, but* $\mathcal{G}_0$ *cannot be composed with* $\mathcal{G}_2$ *(because of f), and* $\mathcal{G}_1$ *cannot be composed with* $\mathcal{G}_2$ *(because of y).*

Obviously, it is always possible to rename the oracles or variables of a game to obtain an equivalent game for which composition is possible.

We can now state our rule for quantum cryptographic reductions.

Lemma 2. *Let $\mathcal{G}$ be a secure game against any quantum polynomial-time adversary. Let $\mathcal{Q}$ be a safe quantum API such that the composition $\mathcal{G} \cdot \mathcal{Q}$ is well-defined. Then, the following rule is sound:*

$$\textsc{Reduction}\quad \frac{\mathcal{E};\Theta \vdash \mathit{Valid}(\boldsymbol{C}) \qquad \mathcal{E};\Theta;\boldsymbol{C};(\boldsymbol{\phi},\boldsymbol{\psi}) \vdash \emptyset \rhd_{\mathcal{G}\cdot\mathcal{Q}} \#(\vec{u}_0;\vec{u}_1)}{\mathcal{E};\Theta \vdash \vec{u}_0 \sim \vec{u}_1}$$

assuming that $\boldsymbol{\phi}$ is a valid initial memory for $\mathcal{G}$ w.r.t. $\mathcal{E}$ and Θ.

Furthermore, if $\mathcal{Q}$ is the safe quantum API of Figure 5, and if $\boldsymbol{\psi}$ is such that $\mathcal{E};\Theta \models_{\mathsf{A}} \mathcal{Q}.\mathsf{state} = \mathsf{state}\ \boldsymbol{t}$, then the rule above can be strengthened by replacing its conclusion by:

$$\mathcal{E};\Theta \vdash \vec{u}_0, \mathsf{state}\ t_0 \sim \vec{u}_1, \mathsf{state}\ t_1.$$

Proof (sketch). The bi-deduction premise ensures the existence of a *classical* polynomial-time adversary $\mathcal{S}$ against the game $\mathcal{G} \cdot \mathcal{Q}$ such that:

$$\forall i \in \{0,1\},\ \mathcal{S}^{\mathcal{G}\cdot\mathcal{Q}}() = [\![\vec{u}_i]\!].$$

Since $\mathcal{Q}$ is a safe quantum API and since $\mathcal{S}$ is *classical*, we know that $\mathcal{S}_0^{\mathcal{G}_i} \stackrel{\text{def}}{=} \mathcal{S}^{\mathcal{G}_i\cdot\mathcal{Q}}$ is a *quantum* polynomial-time adversary against the game $\mathcal{G}$. Since

$$\forall i \in \{0,1\},\ \mathcal{S}_0^{\mathcal{G}_i}() = [\![\vec{u}_i]\!],$$

$\mathcal{S}_0$ is a valid quantum simulator against the game $\mathcal{G}$, and the indistinguishability $\vec{u}_0 \sim \vec{u}_1$ is valid assuming $\mathcal{G}$ is secure against quantum polynomial-time adversaries.

Further, if $\mathcal{Q}$ is the safe quantum API of Figure 5, then the simulator:

$$\mathcal{S}_1^{\mathcal{G}_i}() \stackrel{\text{def}}{=} \left\{\vec{u} \leftarrow \mathcal{S}_0^{\mathcal{G}_i}();\ \ \mathbf{return}\ (\vec{u}, \mathcal{Q}.\mathsf{state})\right\}$$

is a valid quantum polynomial-time adversary against the game $\mathcal{G}$, Indeed, the quantum variable $\mathcal{Q}.\mathsf{state}$ is known to to be live at that program point, and since $\mathcal{S}_1$ is an adversary against the game $\mathcal{G}$, it is allowed to access the variable $\mathcal{Q}.\mathsf{state}$. We conclude by observing that since $\boldsymbol{\psi}$ is such that $\mathcal{E};\Theta \models_{\mathsf{A}} \mathcal{Q}.\mathsf{state} = \mathsf{state}\ \boldsymbol{t}$, we have:

$$\forall i \in \{0,1\},\ \mathcal{S}_1^{\mathcal{G}_i}() = [\![(\vec{u}_i, \mathsf{state}\ t_i)]\!] \qquad \square$$

Inductive quantum simulator synthesis. The proof-system of [11] for cryptographic bi-deduction features an induction rule allowing for the synthesis of recursive simulators. This inductive rule is used to synthesize simulators for terms with (mutual) recursion, and [11] provides a non-trivial fully-automated inductive proof-search procedure allowing to automatically apply the inductive rule. Example 10 below shows that inferring quantum simulators for terms with mutual recursion in full generality introduces additional difficulties. Instead, we will present how to adapt the procedure of [11] the post-quantum execution model we presented in Figure 2.

Example 10. *We show in Figure 10 two examples of (mutually) recursive functions that cannot be simulated in quantum polynomial-time: on the left, because the quantum value produced by* (f n) *is consumed twice, once in* (g_1 n) *and once in* (g_2 n)*; on the right, because* (u n) *is consumed by both* (v (2 n)) *and* (v (2 n + 1)).

```
inductive nat = Z : nat | S : nat → nat.

let f (n : nat) =
  match n with
  | Z → (witness, ϵ)
  | S p → att0(g1 p, g2 p)

and g1 (n : nat) =
  let st, _ = att1(f n) in
  st

and g2 (n : nat) =
  let _, inp = att2(f n) in
  inp

let u (n : nat) =
  match n with
  | Z → (witness, ϵ)
  | S p → att0(u p, v p)

and v (n : nat) =
  let _, inp = att0(u ⌊n/2⌋) in
  inp
```

We assume given three symbols att_0, att_1 and att_2, representing quantum polynomial-time functions. We assume that att_0 is of type $(\mathsf{Hilb}_{\mathsf{msg}} * \mathsf{msg}) \to (\mathsf{Hilb}_{\mathsf{msg}} * \mathsf{msg})$, and att_1 and att_2 are of type $\mathsf{Hilb}_{\mathsf{msg}} \to (\mathsf{Hilb}_{\mathsf{msg}} * \mathsf{msg})$. For the sake of simplicity, we omit the quantum randomness argument for all quantum functions. Here, $\lfloor x \rfloor$ rounds x to the greatest integer smaller or equal to x, ϵ denotes the empty bit-string and witness is an arbitrary constant of type $\mathsf{Hilb}_{\mathsf{msg}}$.

Figure 10: Example of a set of mutually recursive functions that cannot be simulated in quantum polynomial-time.

The example above shows that, when building a simulator for terms computed by recurrence, we need to make sure that quantum values produced by a recursive function are not consumed multiple times, either by distinct functions or by distinct calls to the same function.

We present in Figure 11 an induction rule specialized for the quantum execution model of Figure 2 and the safe quantum API of Figure 5. This induction rule handles the quantum state in the background, and only asks the user the prove that the output of the protocol can be simulated when they are provided with the input (more precisely, the rule provides them with all the past inputs). Notably, the bi-deduction premise asks that the pre-condition $\boldsymbol{\phi}_q$ on the state of the quantum API $\mathcal{Q}$ is left unchanged: in practice, this pre-condition is supposed to be discharged without calling $\mathcal{Q}$.

Lemma 3. *The rule in Figure 11 is sound.*

Proof. We want to prove that:

$$\mathcal{E};\Theta;\Pi_x \boldsymbol{C}_q \cdot \Pi_x \boldsymbol{C};\left(\boldsymbol{\phi}_0 \wedge (\boldsymbol{\phi}_q\ \mathsf{init});\boldsymbol{\phi}_0 \wedge (\boldsymbol{\phi}_q\ \boldsymbol{t})\right) \vdash \vec{\boldsymbol{u}} \rhd_{\mathcal{G}\cdot\mathcal{Q}} (\boldsymbol{v}\ \boldsymbol{t} \mid \boldsymbol{f}) \tag{21}$$

knowing that:

$$\mathcal{E}_0;\Theta,[x \neq \mathsf{init}]_{\mathsf{e}};\boldsymbol{C};\left(\boldsymbol{\phi}_0 \wedge (\boldsymbol{\phi}_q\ x);\boldsymbol{\phi}_0 \wedge (\boldsymbol{\phi}_q\ x)\right) \vdash \vec{\boldsymbol{in}} \rhd_{\mathcal{G}\cdot\mathcal{Q}} (\mathsf{output}\ x \mid \boldsymbol{f} \wedge x \preceq \boldsymbol{t}). \tag{22}$$

First, we apply a time-sensitive induction rule (our rule can be found in Figure 12, and is a simplification of the rule of [12]), to (21), using:

- The well-founded total ordering $x \prec y \stackrel{\text{def}}{=} \neg\mathsf{happens}(y) \vee x < y$ where $<$ is the usual order over for timestamps. Note that the minimal element for $\prec$ is init. Further, we let $\preceq$ be the reflexive closure of $\prec$, and $\mathsf{pred}_\prec$ the predecessor for the ordering $\prec$ (we choose that $\mathsf{pred}_\prec\ \mathsf{init} = \mathsf{init}$).

$$\textsc{Induction}_{\mathrm{q}}^{\mathrm{tr}}$$

$$\frac{\begin{array}{c} f \in \vec{u} \qquad t \in \vec{u} \qquad \mathcal{E};\Theta \models [\mathsf{happens}(t)]_{\mathrm{e}} \\ \mathcal{E}_0;\Theta,[x \neq \mathsf{init}]_{\mathrm{e}};C;\left(\phi_0 \wedge (\phi_q\ x);\phi_0 \wedge (\phi_q\ x)\right) \vdash \vec{in} \rhd_{\mathcal{G}\cdot\mathcal{Q}} (\mathsf{output}\ x \mid f \wedge x \leq t) \end{array}}{\mathcal{E};\Theta;\Pi_x C_q \cdot \Pi_x C;\left(\phi_0 \wedge (\phi_q\ \mathsf{init});\phi_0 \wedge (\phi_q\ t)\right) \vdash \vec{u} \rhd_{\mathcal{G}\cdot\mathcal{Q}} (v\ t \mid f)}$$

where

$$\vec{in} = \vec{u}, \lambda y.\ (v\ y \mid y < x \wedge x \leq t \wedge f), (\mathsf{input}\ x \mid x \leq t \wedge f), x \qquad v\ y \stackrel{\text{def}}{=} (\mathsf{input}\ y, \mathsf{output}\ y, \mathsf{transcript}\ y)$$

$$\phi_q\ x \stackrel{\text{def}}{=} (\mathcal{Q}.\mathsf{state} = \mathsf{state}\ x \wedge \mathcal{Q}.\mathsf{input} = \mathsf{input}\ x) \qquad C_q \stackrel{\text{def}}{=} (\emptyset;\mathsf{qrnd};x;\mathsf{T}_{\mathrm{G}}^{\mathsf{loc}};x \leq t) \qquad \mathcal{E}_0 \stackrel{\text{def}}{=} (\mathcal{E}, x : \mathsf{timestamp})$$

and where $\mathsf{vars}(\phi_0) \cap \mathcal{Q}.\mathsf{glob} = \emptyset$, and $\mathcal{Q}$ is the quantum safe API of Figure 5.

Figure 11: Specialized induction rule for the quantum execution model.

$$\textsc{Induction}$$

$$\frac{\begin{array}{c} t \in in \qquad f \in in \qquad \mathcal{E};\Theta \models \mathsf{adv}(<) \\ (\mathcal{E}, x : \tau);\Theta;C;(\mathcal{I}_<(x);\mathcal{I}_\leq(x)) \vdash in_{\mathrm{rec}} \rhd \left(u\ x \mid f \wedge x \leq t\right) \\ \mathcal{E};\Theta \models_{\mathsf{A}} \phi_0 \sqsubseteq \mathcal{I}_<(x_0) \qquad (\mathcal{E}, x : \tau);\Theta,[x_0 < x \leq t]_{\mathrm{e}} \models_{\mathsf{A}} \mathcal{I}_\leq(\mathsf{pred}\ x) \sqsubseteq \mathcal{I}_<(x) \end{array}}{\mathcal{E};\Theta;\Pi_x C;(\phi_0;\mathcal{I}_\leq(t)) \vdash in \rhd_{\mathcal{G}} (u\ t \mid f)}$$

We require that $u \equiv \lambda(x : \tau).\ u_0$ where τ is a `fixed` and `finite` base-type. We assume a total order $<$ over τ. We let $in_{\mathrm{rec}} \stackrel{\text{def}}{=} (in, x, \lambda y.(u\ y \mid y < x \wedge x \leq t))$. Let x_0 be the minimal element for $<$, and $\mathsf{pred}\ x$ the predecessor of x w.r.t. $<$ for any x (we leave the predecessor of x_0 unspecified).

Figure 12: Time-sensitive induction rule inspired from the rule of [12].

- The memory invariants:

$$\mathcal{I}_<\ x \stackrel{\text{def}}{=} \phi_0 \wedge (\phi_q\ (\mathsf{pred}_\prec\ x)). \qquad \mathcal{I}_\leq\ x \stackrel{\text{def}}{=} \phi_0 \wedge (\phi_q\ x)$$

We must prove that:

$$\mathcal{E}_0;\Theta;C_q \cdot C;\left(\phi_0 \wedge (\phi_q\ (\mathsf{pred}_\prec\ x));\phi_0 \wedge (\phi_q\ x)\right) \vdash \vec{u}, \lambda y.\ (v\ y \mid y \prec x \wedge x \preceq t \wedge f), x \rhd_{\mathcal{G}\cdot\mathcal{Q}} (v\ x \mid f \wedge x \preceq t) \quad (23)$$

and that:

$$\mathcal{E};\Theta \models_{\mathsf{A}} \phi_0 \wedge (\phi_q\ \mathsf{init}) \sqsubseteq \mathcal{I}_<(\mathsf{init}) \quad (24)$$

$$\mathcal{E}_0;\Theta,[\mathsf{init} \prec x \preceq t]_{\mathrm{e}} \models_{\mathsf{A}} \mathcal{I}_\leq(\mathsf{pred}_\prec\ x) \sqsubseteq \mathcal{I}_<(x) \quad (25)$$

$$\mathcal{E};\Theta \models \mathsf{adv}(\prec) \quad (26)$$

First, we show the side-conditions above:

- Eq. (24) is straightforward, since $\mathcal{I}_<(\mathsf{init})$ is the formula $\phi_0 \wedge \mathsf{pred}_q(\mathsf{pred}_\prec\ \mathsf{init})$, which is equivalent to $\phi_0 \wedge \mathsf{pred}_q(\mathsf{init})$ (as we chose to have $\mathsf{pred}_\prec\ \mathsf{init} = \mathsf{init}$).
- Condition Eq. (25) is immediate as both formulas are identical, and Eq. (26) follows from the fact that the builtin ordering $<$ over timestamp is adversarial, i.e. $\mathcal{E};\Theta \models \mathsf{adv}(<)$.

Second, using that $\mathcal{E};\Theta \models [\mathsf{happens}(t)]_{\mathrm{e}}$, we replace the orderings $\prec$ and $\preceq$ by, resp., $<$ and $\leq$ in Eq. (23) when possible. Thus, we are left to prove that:

$$\mathcal{E}_0;\Theta;C_q \cdot C;\left(\phi_0 \wedge (\phi_q\ (\mathsf{pred}_\prec\ x));\phi_0 \wedge (\phi_q\ x)\right) \vdash \vec{u}, \lambda y.\ (v\ y \mid y < x \leq t \wedge f), x \rhd_{\mathcal{G}\cdot\mathcal{Q}} (v\ x \mid f \wedge x \leq t) \quad (27)$$

We apply the case disjunction rule of [12] on the condition $t = \mathsf{init}$. First, we can check that this condition is bi-deducible, since x appears in the inputs and we have the builtin assumption $\mathcal{E};\Theta \models \mathsf{adv}(\mathsf{init})$. Then, we are left to prove that (after simplifications):

$$\mathcal{E}_0;\Theta,[x = \mathsf{init}]_{\mathrm{e}};C_0;\left(\phi_0 \wedge (\phi_q\ (\mathsf{pred}_\prec\ x));\phi_0 \wedge (\phi_q\ x)\right) \vdash \vec{u} \rhd_{\mathcal{G}\cdot\mathcal{Q}} (v\ x \mid f \wedge x \leq t) \quad (28)$$

$$\mathcal{E}_0;\Theta,[x \neq \mathsf{init}]_{\mathrm{e}};C_1;\left(\phi_0 \wedge (\phi_q\ (\mathsf{pred}_\prec\ x));\phi_0 \wedge (\phi_q\ x)\right) \vdash \vec{u}, \lambda y.\ (v\ y \mid y < x \leq t \wedge f), x \rhd_{\mathcal{G}\cdot\mathcal{Q}} (v\ x \mid f \wedge x \leq t) \quad (29)$$

where C_0 and C_1 must be such that $C_0 \cdot C_1 = C_q \cdot C$ (as we will see, we take $C_0 = \emptyset$ and $C_1 = C_q \cdot C$).

The first bi-deduction (28) goal is easy to prove, since $v\ \mathsf{init}$ is $(\epsilon, \epsilon, \epsilon)$, which is clearly bi-deducible by taking $C_0 = \emptyset$. This leaves the game's state unchanged, which satisfies the memory side-conditions since $\phi_0 \wedge (\phi_q\ (\mathsf{pred}_\prec\ \mathsf{init}))$ and $\phi_0 \wedge (\phi_q\ \mathsf{init})$ are equivalent (as $\mathsf{pred}_\prec\ \mathsf{init} = \mathsf{init}$).

For the second bi-deduction Eq. (29) goal, we start by replacing $\mathsf{pred}_\prec$ by pred since it occurs in a position where the timestamps is known to happen and to be different from init. Thus, we need to show that:

$$\mathcal{E}_0;\Theta_{\neq};C_1;\left(\phi_0 \wedge (\phi_q\ (\mathsf{pred}\ x));\phi_0 \wedge (\phi_q\ x)\right) \vdash \vec{u}, \lambda y.\ (v\ y \mid y < x \leq t \wedge f), x \rhd_{\mathcal{G}\cdot\mathcal{Q}} (v\ x \mid f \wedge x \leq t) \quad (30)$$

where $\Theta_{\neq} \stackrel{\text{def}}{=} \Theta, [x \neq \mathsf{init}]_{\mathrm{e}}$.

We conclude by computing $v\ x = (\mathsf{input}\ x, \mathsf{output}\ x, \mathsf{transcript}\ x)$ component by component using the transitivity rule.

$$\mathcal{E}_0;\Theta_{\neq};C_q;\left(\phi_0 \wedge (\phi_q\ (\mathsf{pred}\ x));\phi_0 \wedge (\phi_q\ x)\right) \vdash \vec{in_0} \rhd_{\mathcal{G}\cdot\mathcal{Q}} (\mathsf{input}\ x \mid f \wedge x \leq t)$$

$$\mathcal{E}_0;\Theta_{\neq};C;\left(\phi_0 \wedge (\phi_q\ x);\phi_0 \wedge (\phi_q\ x)\right) \vdash \vec{in} \rhd_{\mathcal{G}\cdot\mathcal{Q}} (\mathsf{output}\ x \mid f \wedge x \leq t)$$

$$\mathcal{E}_0;\Theta_{\neq};\emptyset;(\boldsymbol{\phi}_0 \wedge (\boldsymbol{\phi}_q\ x);\boldsymbol{\phi}_0 \wedge (\boldsymbol{\phi}_q\ x)) \vdash \vec{\boldsymbol{in}}_1 \rhd_{\mathcal{G}\cdot\mathcal{Q}} (\mathsf{transcript}\ x \mid \boldsymbol{f} \wedge x \leq \boldsymbol{t})$$

where:

$$\vec{\boldsymbol{in}}_0 \stackrel{\text{def}}{=} \vec{\boldsymbol{u}}, \lambda y.(\boldsymbol{v}\ y \mid y < x \leq \boldsymbol{t} \wedge \boldsymbol{f}), x$$

$$\vec{\boldsymbol{in}} \stackrel{\text{def}}{=} \vec{\boldsymbol{u}}, \lambda y.(\boldsymbol{v}\ y \mid y < x \leq \boldsymbol{t} \wedge \boldsymbol{f}), x, (\mathsf{input}\ x \mid \boldsymbol{f} \wedge x \leq \boldsymbol{t})$$

$$\vec{\boldsymbol{in}}_1 \stackrel{\text{def}}{=} \vec{\boldsymbol{u}}, \lambda y.(\boldsymbol{v}\ y \mid y < x \leq \boldsymbol{t} \wedge \boldsymbol{f}), x, (\mathsf{input}\ x \mid \boldsymbol{f} \wedge x \leq \boldsymbol{t}), (\mathsf{output}\ x \mid \boldsymbol{f} \wedge x \leq \boldsymbol{t})$$

The second and third bi-deduction goals, which must respectively compute output x and transcript x, are easy to establish: the former is exactly the hypothesis Eq. (22), while the latter is a straightforward computation. For the first bi-deduction goal, which must computes input x, we use the API $\mathcal{Q}$. Indeed, using the fact that $\boldsymbol{\phi}_q$ (pred x), we know that $\mathcal{Q}.\mathsf{state} = \mathsf{state}\ (\mathsf{pred}\ x)$. Thus, calling the oracle $\mathcal{Q}.\mathsf{step}(x, \mathsf{transcript}\ (\mathsf{pred}\ x))$ yields a memory satisfying $\mathcal{Q}.\mathsf{state} = \mathsf{state}\ (x)$ — notice that transcript (pred x) can be computed using $\boldsymbol{v}\ x$. We conclude by calling $\mathcal{Q}.\mathsf{get\_input}$ which yields input x. □

E.4.1 Implementation. A few observations allow to implement the rule of Figure 11 with minimal changes to the cryptographic bi-deduction engine. When solving the bi-deduction premise:

$$\mathcal{E}_0;\Theta_{\neq};\boldsymbol{C};(\boldsymbol{\phi}_0 \wedge (\boldsymbol{\phi}_q\ x);\boldsymbol{\phi}_0 \wedge (\boldsymbol{\phi}_q\ x)) \vdash \vec{\boldsymbol{in}} \rhd_{\mathcal{G}\cdot\mathcal{Q}} (\mathsf{output}\ x \mid \boldsymbol{f} \wedge x \leq \boldsymbol{t}) \tag{31}$$

the oracles of the game $\mathcal{Q}$ must not be called by the simulator being synthesized (all necessary calls to the $\mathcal{Q}$'s oracles have already been made). Thus, there is no need to implement $\mathcal{Q}$ explicitly, and the bi-deduction goal can be simply solved using the oracles of $\mathcal{G}$. Formally, we can then add back $\mathcal{Q}$ through the framing result below.

Lemma 4 (Framing). *For any games $\mathcal{G}_0$ and $\mathcal{G}_1$ such that the composition $\mathcal{G}_0 \cdot \mathcal{G}_1$ is well-defined, for any assertion formulas $\boldsymbol{\phi}_0, \boldsymbol{\phi}_1, \boldsymbol{\psi}_1$ such that $\boldsymbol{\phi}_0$ does not refer to the global variables of $\mathcal{G}_1$, and $\boldsymbol{\phi}_1, \boldsymbol{\psi}_1$ do not refer to the global variables of $\mathcal{G}_0$, i.e.:*

$$\mathsf{vars}(\boldsymbol{\phi}_0) \cap (\mathcal{G}_1.\mathsf{glob} \cup \mathcal{G}_1.\mathsf{glob}_\$) = \emptyset$$
$$\mathsf{vars}(\boldsymbol{\phi}_1) \cap (\mathcal{G}_0.\mathsf{glob} \cup \mathcal{G}_0.\mathsf{glob}_\$) = \emptyset$$
$$\mathsf{vars}(\boldsymbol{\psi}_1) \cap (\mathcal{G}_0.\mathsf{glob} \cup \mathcal{G}_0.\mathsf{glob}_\$) = \emptyset$$

If $\boldsymbol{\phi}_0$ and $(\boldsymbol{\phi}_1, \boldsymbol{\psi}_1)$ do not share program variables, i.e.:

$$\mathsf{vars}(\boldsymbol{\phi}_0) \cap \big(\mathsf{vars}(\boldsymbol{\phi}_1) \cup \mathcal{X}(\boldsymbol{\psi}_1)\big) \cap \mathcal{X}_p = \emptyset$$

and if:

$$\mathcal{E};\Theta;\boldsymbol{C};(\boldsymbol{\phi}_1;\boldsymbol{\psi}_1) \vdash \vec{\boldsymbol{in}} \rhd_{\mathcal{G}_1} \boldsymbol{v} \tag{32}$$

then:

$$\mathcal{E};\Theta;\boldsymbol{C};(\boldsymbol{\phi}_0 \wedge \boldsymbol{\phi}_1;\boldsymbol{\phi}_0 \wedge \boldsymbol{\psi}_1) \vdash \vec{\boldsymbol{in}} \rhd_{\mathcal{G}_0\cdot\mathcal{G}_1} \boldsymbol{v}$$

Proof. We rename all the variables declared in the simulator witnessing the premise Eq. (32) such that they do not overlap with the variables of $\mathcal{G}_0$ nor $\boldsymbol{\phi}_0$. We let $\mathcal{S}$ be the simulator post-renaming. Clearly, $\mathcal{S}$ still witnesses Eq. (32). We conclude by observing that $\mathcal{S}$ preserves $\boldsymbol{\phi}_0$, as it does not modify any of $\boldsymbol{\phi}_0$'s variables. □

Thus, we can replace the bi-deduction goal of (31) by

$$\mathcal{E}_0;\Theta_{\neq};\boldsymbol{C};(\boldsymbol{\phi}_0;\boldsymbol{\phi}_0) \vdash \vec{\boldsymbol{in}} \rhd_{\mathcal{G}} (\mathsf{output}\ x \mid \boldsymbol{f} \wedge x \leq \boldsymbol{t})$$

The syntactic disjointness conditions of Lemma 4 are automatically checked by the implementation (more precisely, since $\mathcal{Q}$ is not concretely represented, no assertion can refer to its variable, which makes the checks trivially true).

Further, the conclusion of the induction rule is the union of the generalization $\Pi_x \boldsymbol{C}$ of the premise's constraints $\boldsymbol{C}$ and the generalization $\Pi_x \boldsymbol{C}_q$ of the constraints $\boldsymbol{C}_q$ — intuitively, the latter comes from the implicit computation of input in the inductive part of the simulation. The constraints resulting from the quantum computation $\Pi_x \boldsymbol{C}_q$ only refer to the name qrnd. This name is not supposed to be used anywhere in the protocol but for the quantum measurement following the execution of the quantum adversary. This let us exploit the two results presented below, which we use to automatically discharge the validity check for $\mathcal{Q}$.

First, for any constraint system $\boldsymbol{C}$, we let $\mathsf{names}(\boldsymbol{C})$ be the set of name symbols constrained by $\boldsymbol{C}$, i.e.:

$$\mathsf{names}(\boldsymbol{C}) \stackrel{\text{def}}{=} \{\mathsf{n} \mid (\vec{\alpha}, \mathsf{n}, t, f, \mathsf{T}) \in \boldsymbol{C}\}.$$

and $\mathsf{globs}(\boldsymbol{C})$ be the set of global random variables constrained by $\boldsymbol{C}$, i.e.:

$$\mathsf{globs}(\boldsymbol{C}) \stackrel{\text{def}}{=} \{v \mid (\vec{\alpha}, \mathsf{n}, t, f, \mathsf{T}^{\mathsf{glob}}_{\mathsf{G},v}) \in \boldsymbol{C}\}.$$

The following lemma allow to decompose the validity of a concatenation of constraint systems $\boldsymbol{C}_0 \cdot \boldsymbol{C}_1$ into the conjunction of the validity of $\boldsymbol{C}_0$ and $\boldsymbol{C}_1$.

Lemma 5 (Framing constraints). *For any constraint systems $\boldsymbol{C}_0$ and $\boldsymbol{C}_1$ such that:*

$$\mathsf{names}(\boldsymbol{C}_0) \cap \mathsf{names}(\boldsymbol{C}_1) = \emptyset \tag{33}$$
$$\mathsf{globs}(\boldsymbol{C}_0) \cap \mathsf{globs}(\boldsymbol{C}_1) = \emptyset \tag{34}$$

Then:

$$\mathcal{E};\Theta \models \mathsf{Valid}(\boldsymbol{C}_0 \cdot \boldsymbol{C}_1) \quad \textit{iff.} \quad \mathcal{E};\Theta \models \mathsf{Valid}(\boldsymbol{C}_0) \wedge \mathsf{Valid}(\boldsymbol{C}_1)$$

Proof (sketch). Valid is defined as:

$$\mathsf{Valid}(\boldsymbol{C}) \stackrel{\text{def}}{=} [\bigwedge_{c_1,c_2 \in \boldsymbol{C}} \mathsf{Fun}(c_1,c_2) \wedge \mathsf{Fresh}(c_1,c_2) \wedge \mathsf{Unique}(c_1,c_2)\ ]_{\mathsf{e}}$$

where the components Fun, Fresh, and Unique are defined in [11, Figure 5]. We quickly argue why our property holds for each component:

- For the Fun and Fresh components of Valid, this is immediate since the names of $\boldsymbol{C}_0$ and $\boldsymbol{C}_1$ are disjoint (Eq. (34)).
- For the Unique component of Valid, we use the fact that the global samplings constrained by $\boldsymbol{C}_0$ and $\boldsymbol{C}_1$ are disjoint (Eq. (34)). □

Remark that the safe quantum API of Figure 5 uses no random variables, so condition Eq. (34) trivially holds.

Condition Eq. (33) is checked by the following lemma, which states that $\Pi_x \boldsymbol{C}_q$ is always valid.

Lemma 6 (Framing constraints). *For any $\mathcal{E}$ and Θ, we have that:*

$$\mathcal{E};\Theta \models \mathsf{Valid}(\Pi_x \boldsymbol{C}_q) \qquad \textit{where } \boldsymbol{C}_q = (\emptyset;\mathsf{qrnd};x;\mathsf{T}^{\mathsf{loc}}_{\mathsf{G}};x \leq \boldsymbol{t})$$

Proof. We simply observe that $\Pi_x \boldsymbol{C}_q = (x; \mathsf{qrnd}; x; \mathsf{T}^{\mathsf{loc}}_{\mathsf{G}}; x \leq \boldsymbol{t})$, which is trivially valid since it represents a multi-set containing no duplicated names and always uses the same tag. □

Using both lemmas above, the validity of $\Pi_x \boldsymbol{C}_q \cdot \Pi_x \boldsymbol{C}$ is equivalent to the validity of $\Pi_x \boldsymbol{C}$, which simplifies the proof-obligation discharged to the user — the syntactic checks on name symbols of Lemma 5 is automatically checked by the tool.

E.5 Dedicated Cryptographic Tactics

Squirrel has tactics dedicated to specific cryptographic axioms, which allow to go beyond the generic and heuristic treatment of the **crypto** tactic.

E.5.1 Existing tactics and soundness proofs. Consider the case of the PRF axiom, which states that a keyed hash function hash produces values that are indistinguishable from fresh random values. Over a sequence of term $\vec{u}$, hash m k, with k a name, the PRF axiom intuitively enables us to conclude that $\vec{u}$, hash m k $\sim \vec{u}, \mathsf{n}_{\mathsf{fresh}}$ with $\mathsf{n}_{\mathsf{fresh}}$ a fresh name, under the condition that we can see $\vec{u}$, hash(m, k) as the result of a PPTM interacting with the hash oracle provided by the PRF assumption. Squirrel thus has a logical rule, which allows to use the PRF axiom, with a set of premises that guarantees the previous condition.

The PRF rule is formally expressed as follows.

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

where, only recalled here informally,

- $\mathcal{E}; \Theta; \emptyset \vdash_{\mathsf{pptm}} \vec{u}, m$ states that the terms can be simulated step by step and incrementally by a PPTM;
- $\psi^{\mathsf{k}}_{\mathsf{key}}(\vec{u}, m)$ states that k only occurs as the key to hash;
- $\psi^{\mathsf{k}}_{\mathsf{msg}}(\vec{u}, m)$ states that all the other messages hashed in $\vec{u}$ are distinct from m.

The proof showing that if the interpretation of hash satisfies the cryptographic PRF assumption then PRF is a sound rule builds explicitly a reduction, with a PPTM which computes incrementally the subterms in $\mathcal{E}; \Theta; \emptyset \vdash_{\mathsf{pptm}} \vec{u}, m$, calling the hash oracle of the PRF assumption whenever it needs to compute the value of some hash x k. And the side conditions imply the validity of the reduction.

Generic soundness proofs. Squirrel's logic has a set of such tactics, covering notably PRF, EUF, ENC − KP, CCA$_1$, DDH, XOR, ... The soundness of all the corresponding tactics follow the same pattern as the one presented for PRF:

- a pptm judgement states that the targeted terms can be simulated by a PPTM in an incremental fashion, subterm after subterm;
- some side conditions ensure that, e.g., the secret keys, are only used in key position, and thus that we can indeed build a reduction by using the oracles provided by the corresponding assumption.

E.5.2 Post-quantum sound tactics. Adapting those existing tactics can thus be done in two steps. First, we define an equivalent pqtm judgement, which also allows to extract a simulator incrementally for every subterm. Such a judgment then allows us to literally use the existing reduction to build our quantum polynomial time simulator. However, it is well known that all PPTM reductions don't instantly yield valid reductions for PQTM, one must for instance check that all reductions treat the adversary in a black-box fashion, and for instance do not rely on rewinding techniques. This check was in fact already performed in [24], we do not recover it here. However, we do provide formally below the definition of the PQTM judgement, and provide an efficient way to check it in the Squirrel implementation, building on the previous notion of API.

Capturing incremental PQTM simulation. We adapt the pptm judgment from [10] to our setting. Intuitively, the pqtm judgment $\mathcal{E}; \Theta; \vec{\alpha} \vdash_{\mathsf{pqtm}} t$ states that there exists a folding of PQTMs and PPTMs that computes $[\![t]\!]$ by following the *same computation steps* as the recursive definition of the semantics $[\![\cdot]\!]$. This intuition is formalized by once again relying on the $\mathsf{qadv}_{\mathsf{X}}$ predicate.

Definition 10. *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 of order 0 or 1 well-typed in $\mathcal{E}, \vec{\alpha}$. The judgement*

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

is valid iff. for any model $\mathbb{M} : \mathcal{E}$ such that $\mathbb{M} \models \Theta$, then there exists t_S such that:

- *if t has order 0, then $\mathbb{M} \models \mathsf{qadv}_{\{\mathsf{a},\mathsf{h}\}}(\lambda\, \vec{\alpha}.\, t;\ t_S)$.*
- *if t has order 1 and is of the form $\lambda \vec{x}.t'$, then $\mathbb{M} \models \mathsf{qadv}_{\{\mathsf{a},\mathsf{h}\}}(\lambda\, \vec{\alpha} \cdot \vec{x}.\, t;\ t_S)$.*

We note that by definition of the predicate $\mathsf{qadv}_{\mathsf{X}}$, the statement $\mathbb{M} \models \mathsf{qadv}_{\{\mathsf{a},\mathsf{h}\}}(\lambda\, \vec{\alpha} \cdot \vec{x}.\, t;\ t_S)$ can only be true if $\vec{\alpha} \cdot \vec{x}$ contains at most one quantum element, and note that this condition propagates to every subterm.

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 the simulator to follow the same steps as $[\![\cdot]\!]$).

We assume that all judgments must be well-typed.

Handling recursive definitions. As we consider classical protocol facing a quantum adversary, we note that in the terms we will consider, only the attacker manipulates quantum value, and we in fact know precisely how it is manipulating it. Thus, rather than trying to come up with a general proof system for the judgment, it is sufficient to design a single rule that matches our execution model, where we check that everything but the attacker is PPTM, and that the attacker is only used following the execution model. We thus adapt the polynomial time criterion for terms with recursive function from [10] in order to manage our builtin quantum execution model.

Proposition 12. *Let $\mathcal{E}$ be an environment such that*

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

where $\mathcal{E}_0$ contains only variables declarations and $\mathcal{E}_Q$ defines frame, transcript, state and input according to the execution model of Figure 2. Also denoting output *by x_{n+1} for uniformity, we assume that for every $i \in [1; n+1]$, 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 $(x'_i)_{i\in[1;n+1]}, \mathsf{input}', \mathsf{transcript}', \mathsf{frame}'$ *be fresh variable names and* $\sigma \stackrel{\text{def}}{=} \{x_i \mapsto x'_i \mid i \in [1;n+1]\}\{\mathsf{input} \mapsto \mathsf{input}'\}\{\mathsf{transcript} \mapsto \mathsf{transcript}'\}\{\mathsf{frame} \mapsto \mathsf{frame}'\}$ *be a substitution. If, given input* y_i*, the bodies of the defined variable* $(x_i)_{i\in[1;n+1]}$ *are all computable by a PPTM, assuming (by recurrence) that other defined variables are computable,* i.e.*, for all* $i \in [1; n+1]$*,*

$$\mathcal{E}_0; \Theta; \vec{\alpha}, (x'_i)_{i\in[1;n+1]}, y_i, \mathsf{input}', \mathsf{transcript}' \vdash_{pptm} t_i\sigma$$

then the following rule is a sound rule

QPT.Def:Rec-Finite

$$\frac{\mathcal{E}; \Theta; \vec{\alpha}, (x'_i)_{i\in[1;n+1]}, \mathsf{input}', \mathsf{transcript}' \vdash_{pptm} s\sigma}{\mathcal{E}; \Theta; \vec{\alpha} \vdash_{pqtm} s, \mathsf{state}\ \mathsf{ts}_{max}}$$

where ts_{max} *is any term such that:*

$$\mathcal{E}, \Theta \models [\bigwedge_{\{\mathsf{ts}|x\ \mathsf{ts}\in\mathcal{ST}^0_{\mathcal{E}}(t)|x\ \textit{defined in}\ \mathcal{E}\}} \mathsf{ts}_{max} \geq \mathsf{ts}\})]_{\mathsf{e}}$$

This proposition essentially states that if a term uses our quantum execution model, all other recursive functions are PPTM, and if the top level term is PQTM when ignoring the recusrive calls, then the whole term is. This notably imply that we can only prove PQTM for terms that do not call att outside of the way prescribed by the execution model, and where att is the only quantum function symbol. Note that we must do the substitution of the recursive variables in order to remove them and reduce ourselves to finite terms that can indeed be proved to be simulatable by PPTMs.

Proof (sketch). Take a model $\mathbb{M}$, then recursion depth is finite and independent of η.

Thanks to the premise, all the bodies of user defined recursive functions are computable in polynomial-time assuming that recursive calls have been computed. Further, recall that we restrict ourselves to $\mathbb{M}$ such that

$$\mathbb{M} \models \tilde{\forall} t \in \mathsf{timestamp}.\ \mathsf{qadv}(\lambda x.\mathsf{att}(\mathsf{qrnd}\ t, x); t).$$

We thus have a folding of a classical and quantum machine to compute the classical and quantum output of att.

To simulate s, we simply compose the pptm simulator for s with the ones for the recursive functions, and use the folding for the attacker to compute input and state values when needed. As the only quantum function is att and all occurences of att are hidden inside the macros, the global composition can be seen as an attacker following the quantum safe API of Figure 5, and it does yield a valid folding of a finite number of quantum and classical machines overall.

This is enough to argue that $\mathcal{E}; \Theta; \vec{\alpha} \vdash_{\mathsf{pqtm}} s$. In order to also add the state, a tricky point is that the simulator for s without the recursive call might call its order one input (the oracles for recursive calls) in an arbitrary fashion, and might notably make unnecessary calls to state with timestamps bigger than ts_{max}.

To solve this, we notice that we can build a term s' where the macros from $\mathcal{E}_Q$ in s' have been replaced by variants that are equal except that they return $\bot$ whenever called over a timestamp bigger than ts_{max}. By the definition of ts_{max}, we have that the local formula $s = s'$ is valid. We can thus replay the previous argument but over s'. This yield an overall folding simulating s' (and in fact s), in which we are sure that at the end of its execution, the final quantum value it computed is indeed state ts_{max}. We can then at the end return both the value of s' and the corresponding state. □

Note that this rule trivially generalizes to the case where state is any position of a potentially nested tuple of values.