A Logic and an Interactive Prover for the Computational Post-Quantum Security of Protocols

Cas Cremers[1], Caroline Fontaine[2], and Charlie Jacomme[1]

[1]CISPA Helmholtz Center for Information Security, Germany
[2]Université Paris-Saclay, CNRS, ENS Paris-Saclay, Laboratoire Méthodes Formelles, 91190, Gif-sur-Yvette, France

March 28, 2022 – v1.0*

Abstract

We provide the first mechanized post-quantum sound security protocol proofs. We achieve this by developing PQ-BC, a computational first-order logic that is sound with respect to quantum attackers, and corresponding mechanization support in the form of the PQ-Squirrel prover.

Our work builds on the classical BC logic [7] and its mechanization in the Squirrel [5] prover. Our development of PQ-BC requires making the BC logic sound for a single interactive quantum attacker. We implement the PQ-Squirrel prover by modifying Squirrel, relying on the soundness results of PQ-BC and enforcing a set of syntactic conditions; additionally, we provide new tactics for the logic that extend the tool's scope.

Using PQ-Squirrel, we perform several case studies, thereby giving the first mechanical proofs of their computational post-quantum security. These include two generic constructions of KEM based key exchange, two sub-protocols from IKEv1 and IKEv2, and a proposed post-quantum variant of Signal's X3DH protocol. Additionally, we use PQ-Squirrel to prove that several classical Squirrel case studies are already post-quantum sound.

1 Introduction

In recent years, multiple highly-successful tools have been developed to analyze and verify cryptographic protocols and primitives [9,17,18,51,60]. They have proven the usefulness and necessity of computer-aided cryptography, both uncovering critical attacks against widely deployed protocols and helping in the design of new standards [2,10,15,25,28–30,32,33,46,47,50].

In anticipation of developments in quantum computing that would break a lot of widely-used cryptographic primitives, the security community has started to develop many new security primitives and protocols, and revisit old protocols. Additionally, an extensive multi-year NIST standardization process is ongoing to develop new primitives and protocols. These efforts aim to establish mechanisms that are provably secure against quantum attackers. At some level of abstraction, this implies (i) designing new primitives and prove (or assume) that they are secure against a quantum attacker, and (ii) proving that a concrete protocol that uses such primitives is indeed secure against a quantum attacker. In this work, we focus on the latter, and in particular how we can mechanize such proofs.

A classical strategy for proving a protocol's security is a so-called *reduction* proof, which yields computational security guarantees against a polynomial-time attacker. This approach is used in most pen-and-paper proofs by cryptographers, and involves constructing a reduction from any attack on the protocol to an attack on the used cryptographic assumptions, and then reasoning by contradiction. This is a well studied approach with respect to classical (non-quantum) attackers: different flavors of such proofs can be mechanized by tools such as CryptoVerif [17] and EasyCrypt [9]. However, some proof steps commonly used in reductions that are valid for a classical attacker, such as *rewinding*, are in general not valid anymore for quantum attackers. This result is similar to the *no-cloning* theorem [65], which implies that one must be careful when talking about the state of a quantum attacker. As a consequence, a

*An extended abstract of this paper appears at IEEE S&P 2022; this is the full version.

classical reduction proof of a protocol (even based on post-quantum sound primitives) may not be valid for quantum attackers. Unfortunately, there exists no formal framework nor mechanization dedicated to computational proofs of a protocol's security versus a quantum attacker.

In this work, we address this problem by developing PQ-BC, a post-quantum sound variant of a computationally sound protocol logic, and a corresponding tool called the PQ-Squirrel prover, by extending the logic's tool support for the post-quantum setting, as well as adding tactics. We use our new tool to provide the first mechanized post-quantum computational security proofs for several protocols.

Concretely, our work builds on the BC logic [7] and its mechanization in the Squirrel prover [5]. The BC logic can be used to construct security proofs that provide computational guarantees against a classical (non-quantum) attacker, while only working inside a logical framework in which many intricate details have been abstracted. It has notably been used for manual proofs of real-world protocols, see e.g., proofs of RFID based protocols [27], AKA [48], e-voting protocols [6], key-wrapping API [56], and SSH through a composition framework [26]. Reasoning in BC was recently mechanized and extended in the Squirrel prover [5], dedicated to the formal proofs of protocols. Notably, reasoning in BC (and therefore Squirrel) is not sound with respect to a quantum attacker, because the framework allows reduction steps that cannot be reproduced with a quantum attacker.

To develop PQ-BC, we have to make the BC logic sound for a single interactive quantum attacker, while it previously relied on a set of deterministic one-shot attackers. This seemingly small change triggers a cascade of technical changes. We design a new term interpretation for the logic and identify three syntactic conditions for proofs that help ensure their post-quantum soundness. We provide mechanization for PQ-BC in the form of the PQ-Squirrel prover. PQ-Squirrel's soundness relies on the soundness results of PQ-BC and implements the syntactic conditions; additionally, we design and implement new tactics that extend the tool's scope.

Contributions. We see our main contributions as the following:

- First, we develop PQ-BC, the first computational first-order logic to prove guarantees of security protocols whose results are provably sound with respect to a quantum attacker.
- Second, we develop the PQ-Squirrel prover, a mechanized tool support for establishing such guarantees.
- Third, we use our tool to provide the first mechanized proofs of the post-quantum computational security of 11 security protocols as case studies. These include two KEM-based key exchanges [21,38], a post-quantum variant of Signal's X3DH [41], and two protocols from the IKE standards [23, 44] – confirming claims in [37].

Overview We provide in Section 2 the necessary background on the BC logic and the Squirrel prover. Then, in Section 3, we give a high-level overview of the design of the PQ-BC logic and how it differs from the BC logic. In Section 4 we formally define PQ-BC, its syntax and semantics, and its rules; in Section 6 we describe PQ-Squirrel and perform case studies. We discuss current limitations and future work in Section 7, and conclude in Section 8.

Upon first reading, the reader may get a high-level understanding of the paper by skipping Section 4 and directly continuing with Section 6.

We provide some details about quantum computing as well as the implementation of the new tactics in the appendix.

Additional related Work Issues regarding the validity of classical cryptographic reductions in the post-quantum setting have mostly started with [64], which identified the "no-cloning theorem" [65] as a key issue, followed, e.g., by [3].

Key details and difficulties when moving to post-quantum security are generally discussed [39, 59]. They provide some guidelines that are suitable for game-based approaches and gave us many insights, but those guidelines are not suited for the BC logic approach.

There exists many tools for security proofs, we only discuss the most widely used. At one extreme of the spectrum are tools like EasyCrypt [9] and CryptoVerif [17], which provide strong computational guarantees for detailed models of cryptographic primitives, but for whom scaling to larger constructs is more challenging; at the other end of the spectrum, we have tools like Tamarin [51] and Proverif [18], which can analyze much larger protocol mechanisms by using a more abstract symbolic model, but cannot provide computational guarantees. Squirrel, and thus PQ-Squirrel, lies in the middle ground between those two ends: on one side, it provides computational guarantees, that are thus stronger than the one

given by Proverif and Tamarin; on the other side, it operates at a higher level of abstraction than EasyCrypt and CryptoVerif. Consequently, Squirrel is less expressive and thus less suited to reason about cryptographic primitives, but tends to scale better to larger construct. However, Squirrel does not provide any concrete security bounds, and in security proofs over unbounded protocols, the number of sessions is arbitrary and not attacker chosen. For a detailed comparison between Squirrel, EasyCrypt and CryptoVerif, we refer the reader to [5, Appendix E].

CryptoVerif does not have any support for quantum attackers yet, it might be possible to make it quantum-sound by using ideas from our work, such as forbidding some manipulations over the attacker state, and ensuring that a unique quantum attacker process can continue without having to alter or inspect its internal state.

The previously mentioned EasyCrypt is a toolset for constructing cryptographic proofs, which currently mainly targets cryptographic primitives. It was first adapted to the quantum setting with qRHL [63], a formal security prover based on a quantum relational Hoare logic, and later (in concurrent work to ours) to the post-quantum setting with EasyPQC [8]. The qRHL approach works on quantum constructions, which substantially complicates proving classical constructions. For example, there is no equivalent to the classical implication operation over quantum predicates (see e.g., [31]). EasyPQC avoids this overhead by only considering classical constructions.

Similar to our approach, EasyPQC adds new side conditions to its core logic, such as forbidding case distinctions on the attacker's internal state. It is difficult to compare their side conditions to ours, since the conditions are deeply linked to the underlying logics, which are of a very different nature. Notably, EasyPQC supports reasoning in the Qantum Random Oracle Model (QROM). The BC logic does not yet support the ROM (nor QROM), and hence neither do we. This is not an inherent restriction of the logic and could be future work. For our current case studies, we prefer the use of the PRF assumption over the QROM.

EasyPQC and our approach inherit their focus from their starting points: the EasyCrypt approach is more geared towards cryptographic primitives, while BC is designed for protocols. All current EasyPQC case studies are cryptographic primitives, whereas our case studies are protocols. In particular, our case studies for KEM based key exchanges are the first mechanized proofs with computational guarantees of such protocols.

2 Background: the classical BC logic and Squirrel

Below we first recall the main elements of the original BC logic [7] that are relevant for understanding our work in the following sections. In Section 2.5 we describe the Squirrel prover [5], which mechanizes reasoning in the BC logic.

In the computational model, the security of a protocol is established by showing that the protocol cannot be distinguished from its idealized version by any polynomial-time attacker w.r.t. a security parameter. Such security proofs of protocols rely on two ingredients:

- a base assumption, where we assume that some computational problem cannot be solved efficiently, e.g. integer factoring or discrete logarithm;
- a security reduction, where one shows that if there exists an attacker against the protocol, there exists an attacker against the base assumption.

But the construction of security reductions is difficult and error-prone. To ease this process, the BC logic proves the security of protocols inside a first-order logic. This approach requires that within BC, everything is modeled using only terms, i.e., purely syntactic constructs. This is very different from the game-based modeling, where protocols are expressed as abstract programs with procedure calls, states, and side effects. With terms, all protocol actions become pure functional calls, which tends to ease the formal reasoning. This leads to some core elements in the design of the logic: one needs to

1. define terms, as well as an interpretation from protocols to terms so that the terms syntactically describe all the behaviours of the protocol,
2. define logical predicates and rules (which include axioms) to reason about our terms, and
3. show that the rules are sound, i.e., that the rule applications correspond to correct reductions.

We provide an overview of the first two elements in the following, and refer the reader to [7] for details of the rules and their soundness. However, all three elements will be discussed when we present the modified post-quantum sound logic in the following sections.

2.1 Specifying protocol behaviours using syntactic terms

2.1.1 From protocols to terms

Let us consider a very simple example protocol process P, using an informal syntax.

Example 1 (Protocol)**.**

$$P := \textbf{new } \mathsf{sk}.\textbf{in}(x).\textbf{new } \mathsf{r}.\textbf{out}(\mathit{enc}(x, \mathsf{r}, \mathsf{sk})).\textbf{in}(y).\textbf{new } \mathsf{r}'.\textbf{out}(\mathit{enc}(y, \mathsf{r}', \mathsf{sk})).$$

Process P samples a secret key sk, and uses it to encrypt some attacker input x using the random seed r (explicitly modeling probabilistic encryption). It then encrypts a second input y with random seed r'.

Equivalently, in the game-based notation with a stateful probabilistic attacker $\mathcal{A}$ and security parameter η, the experiment that returns the attacker-observable values is defined as:

Experiment $Exp^{P}_{\mathsf{enc},\mathcal{A}}(\eta)$

$\mathsf{sk} \xleftarrow{\$} \{0,1\}^\eta$
$x \xleftarrow{\$} \mathcal{A}(1^\eta)$
$\mathsf{r} \xleftarrow{\$} \{0,1\}^\eta$
$y \xleftarrow{\$} \mathcal{A}(1^\eta, \mathsf{enc}(x, \mathsf{r}, \mathsf{sk}))$
$\mathsf{r}' \xleftarrow{\$} \{0,1\}^\eta$
return $(\mathsf{enc}(x, \mathsf{r}, \mathsf{sk}), \mathsf{enc}(y, \mathsf{r}', \mathsf{sk}))$

To syntactically represent such observable sequences of values using terms, one can use the following constructions:

- fresh values n sampled from an infinite set $\mathcal{N}$, representing randomly sampled bitstrings, such as r, r', and sk above;
- public function symbols $f \in \Sigma$, to model e.g., encryption functions such as enc; and
- variables such as x and y from the set of variables $\mathcal{X}$, modeling attacker inputs.

Any protocol computation can then be modeled as applications of public functions to either fresh values modeling randomly generated values (such as nonces and secret keys), or variables modeling attacker inputs. This is essentially the Dolev-Yao model [34], where the protocol can be described with the following term sequence:

$$\mathsf{enc}(x, \mathsf{r}, \mathsf{sk}),\ \mathsf{enc}(y, \mathsf{r}', \mathsf{sk}) \tag{1}$$

This sequence of terms, that we will refer to as the *frame* of the protocol, represents the possible messages that an attacker can observe during the protocol's execution.

2.1.2 Modeling attacker computations

The sequence of terms in Eq. (1) is not yet sufficient to reason syntactically about protocols, because it does not capture that y probabilistically depends on the value of $\mathsf{enc}(x, \mathsf{r}, \mathsf{sk})$. From a high-level point of view the logic must satisfy

- locality - a term must explicitly contain all its probabilistic dependencies.

Essentially, it needs to syntactically capture that y is the result of an attacker's unknown computation, which depends on the previous messages, e.g., in our example, y depends on r and sk. The BC logic uses free function symbols $\mathtt{att}_i$ that represent unknown pieces of code, i.e., attacker computations that receive as arguments the previous messages seen by the attacker.

The previous frame can now be expressed as:

$$\mathsf{enc}(\mathtt{att}_0(), \mathsf{r}, \mathsf{sk}),\ \mathsf{enc}(\mathtt{att}_1(\mathsf{enc}(\mathtt{att}_0(), \mathsf{r}, \mathsf{sk})), \mathsf{r}', \mathsf{sk}) \tag{2}$$

$\mathtt{att}_0()$ representing the first message (x) computed by the attacker, when it does not have access to any information from the protocol, and $\mathtt{att}_1(\mathsf{enc}(\mathtt{att}_0(), \mathsf{r}, \mathsf{sk}))$ being its second message (y), which is a function of the protocol's first output.

The term-based notation is more akin to a functional view: one cannot use variables such as x to refer to previously computed values. For this reason, $\mathtt{att}_0()$ occurs twice in Eq. (2). However, attacker

computations such as $\mathtt{att}_0$ are probabilistic algorithms, and hence two different invocations might yield different results. This is not the intended interpretation, and it therefore introduces a new requirement on terms:

- stability - two occurrences of the same term must evaluate to the same value.

When one evaluates the value of the previous frame, this implies that the two occurrences of $\mathtt{att}_0()$ evaluate to the same value. This requirement captures that within a single protocol execution, identical terms refer to the same value and not to separate (probabilistic) attacker calls.

2.1.3 Adding conditionals

A last issue to syntactically capture the frame of a protocol is the fact that protocol outputs often depend on conditionals. Let us consider a new example:

$$Q := \textbf{new } \mathsf{n}.\textbf{new } \mathsf{n}'.out(c, \mathsf{n}').\textbf{in}(c, x).\textbf{if } x = n \textbf{ then out}(c, \mathsf{ok}) \textbf{ else out}(c, \mathsf{ko}).$$

This protocol randomly samples two values, i.e. two names, gives one to the attacker, and finally outputs ok if the attacker can provide the second one. Here, the attacker cannot guess with non-negligible probability the name, so ok should almost never be obtained.

To represent this inside terms, the idea is to add to the public function symbols set Σ, a ternary function symbols $\mathtt{ite}(_, _, _)$, that we will in fact denote for clarity **if** _ **then** _ **else** _. This symbol will always be interpreted so that it matches the computations of a conditional. One also needs to write the condition and the outputs, so the BC logic introduces a function symbol of arity 2 that test equality, $_ \doteq _$, along with two arity 0 symbols ok and ko. The frame of the protocol is then expressed by:

$$\mathsf{n}', \textbf{if } \mathtt{att}_0(\mathsf{n}') \doteq \mathsf{n} \textbf{ then } \mathsf{ok} \textbf{ else } \mathsf{ko}$$

Looking at this frame, we can see one of the first strength of the syntactical approach. If we consider the condition $\mathtt{att}_0(\mathsf{n}') \doteq \mathsf{n}$, it is easy to see that the attacker computation $\mathtt{att}_0(\mathsf{n}')$ is completely independent of n, so we can conclude that this equality test cannot succeed with non-negligible probability. This matches the intuition that Q is in fact indistinguishable from:

$$Q' := \textbf{new } \mathsf{n}.\textbf{new } \mathsf{n}'.out(c, \mathsf{n}').\textbf{in}(c, x).\textbf{out}(c, \mathsf{ko}).$$

2.1.4 Reasoning about terms

The BC logic contains in its syntax a binary predicate $\sim$. This predicate expects two sequences of terms and intuitively represents their indistinguishability. Note that $\sim$ has low operator precedence, but we often add parentheses for readability.

Example 2 (BC indistinguishability formula)**.**

$$\left(\mathsf{n}', \textbf{ if } att_0(\mathsf{n}') \doteq \mathsf{n} \textbf{ then } \mathsf{ko} \textbf{ else } \mathsf{ok}\right) \sim \left(\mathsf{n}', \ \mathsf{ok}\right) \tag{3}$$

The protocol modeled on the left-hand side produces a fresh value n', sends it, and then waits to receive another value (represented by $\mathtt{att}_0(\mathsf{n}')$). If the received value is the same as a freshly generated value n, the protocol outputs ko, otherwise ok. The formula expresses that the attacker cannot distinguish this protocol from the protocol that sends a fresh value n', waits for an input, and then outputs ok. To give a flavor of how proofs are performed in the BC logic, we provide some rule examples and prove that Eq. (3) holds in the logic.

Example 3 (Rules and proofs)**.**

$$\frac{}{(t \doteq \mathsf{n}) \sim \mathit{false}}\ =\text{IND} \quad \textit{when } \mathsf{n} \textit{ does not occur in } t \qquad \frac{\phi \sim \mathit{false} \quad (u, v) \sim w}{(u, \textbf{if } \phi \textbf{ then } s \textbf{ else } v) \sim w}\ \textsc{If-f} \qquad \frac{}{u \sim u}\ \textsc{Refl}$$

Logical rules are read bottom-up, where to prove the formula on the bottom, one can prove the formulas on the top. Rule =IND *means that a term t that does not syntactically contain a fresh value n cannot be equal to it, and* If-f *tells us that if a conditional is always false, we can only consider its negative branch. The* Refl *rule encodes that $\sim$ is reflexive. With those rules, we can prove the formula that represents the indistinguishability of Q and Q':*

$$\dfrac{\dfrac{}{\left(att_0(\mathsf{n}') \doteq \mathsf{n}\right) \sim \mathit{false}}\ =\text{IND} \qquad \dfrac{}{(\mathsf{n}', \mathsf{ok}) \sim (\mathsf{n}', \mathsf{ok})}\ \textsc{Refl}}{\left(\mathsf{n}', \textbf{if } att_0(\mathsf{n}') \doteq \mathsf{n} \textbf{ then } \mathsf{ko} \textbf{ else } \mathsf{ok}\right) \sim (\mathsf{n}', \mathsf{ok})}\ \textsc{If-f}$$

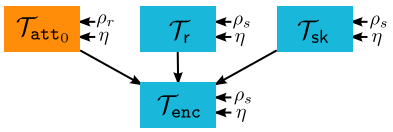

Figure 1: $\mathtt{enc}(\mathtt{att}_0(), \mathsf{r}, \mathsf{sk})$

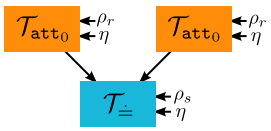

Figure 2: $\mathtt{att}_0() \doteq \mathtt{att}_0()$

2.2 A faithful computational interpretation

We previously described the BC way to syntactically describe the behaviour of a protocol interacting with an attacker. To ultimately get to a logic that provides computational guarantees, those syntactic terms must capture all the behaviours of the protocol. To do so, BC provides a formal way to interpret those terms, so that their possible evaluations match those of the real protocol. Intuitively, given an attacker against the real world protocol, and given the syntactic frame (e.g., Eq. (2)), it should be possible to build a simulator producing the same results as the protocol. If this is possible, then the frame does indeed capture all the possible behaviours of the protocol, and we can thus use it to reason about its security. Recall that we already mentioned two high-level desirable properties of our constructs:

- locality - a term should completely contain all its probabilistic dependencies.
- stability - our interpretation should interpret two occurrences of the same term to the same value.

2.2.1 Interpreting terms

We now describe how BC interprets a term, i.e., computes the probabilistic result of a term, while satisfying both locality and stability. The interpretation has three parameters: the security parameter η and two infinite random bitstrings ρ_s and ρ_r. The interpretation extracts the randomness used in probabilistic protocol functions from ρ_s, and the attacker randomness from ρ_r. By universally quantifying over those parameters and the attacker computations, it captures all the possible executions of a protocol.

The interpretation depends on a set of Polynomial Time Turing Machines (PTTM) that compute the evaluation function $\mathcal{T}$ of a term, i.e.,

- a machine $\mathcal{T}_{\mathsf{n}}$ for n: outputs the value of a given fresh value, depending on η and ρ_s, which may typically extract a sequence of η bits from ρ_s;
- a machine $\mathcal{T}_f$ for f: computes the output of a public function depending on its arguments;
- a machine $\mathcal{T}_{\mathtt{att}_i}$ for $\mathtt{att}_i$: performs some attacker computation, depending on its arguments, η and ρ_r.

To interpret a term, one can consider the term as a tree of (sub)terms, and recursively call the corresponding Turing machine from the leaves to the root. Going back to the term $\mathtt{enc}(\mathtt{att}_0(), \mathsf{r}, \mathsf{sk})$ occurring in Eq. (2), its possible values are obtained by running the simulation described in Fig. 1.

By design, all Turing machines in the interpretation are deterministic: the randomness is explicitly passed to capture probabilistic behaviour (using ρ_s and ρ_r). This ensures that the interpretation satisfies stability: two occurrences of $\mathtt{att}_0()$ in the term imply two computations of $\mathcal{T}_{\mathtt{att}_0}$ that deterministically evaluate to the same value. For example, if we consider the function symbol $\doteq$ that models equality testing, the evaluation of Fig. 2 always returns true, which is the expected behaviour.

Note that $\mathcal{T}_{\mathtt{att}_0}$ and $\mathcal{T}_{\mathtt{att}_1}$ do not share any implicit state and, more generally, nor would $\mathcal{T}_{\mathtt{att}_i}$ and $\mathcal{T}_{\mathtt{att}_{i+1}}$. This may seem counter-intuitive, as $\mathtt{att}_i$ and $\mathtt{att}_{i+1}$ usually represent two unknown computations of the same attacker $\mathcal{A}$, that can in practice maintain a state between successive calls. The BC logic does not explicitly ensure this, but by modeling conventions, we always give all the previous inputs of $\mathtt{att}_i$ also to $\mathtt{att}_{i+1}$. As a result, in the simulations one can make $\mathcal{T}_{\mathtt{att}_{i+1}}$ re-perform all the computations of $\mathcal{T}_{\mathtt{att}_i}$ to recompute its state. Modelling each attacker call by a distinct machine is a crucial design choice of the BC logic that ensures locality.

2.2.2 Protocol interactions

We now focus on protocol interactions, as it will play an important role when moving to the quantum setting. We informally describe an attacker $\mathcal{A}$ in the real world as a sequence of unknown operations $\{\mathcal{A}^i\}$, that produces the message m_i at step i, and implicitly maintains a state ϕ_i between each computation.

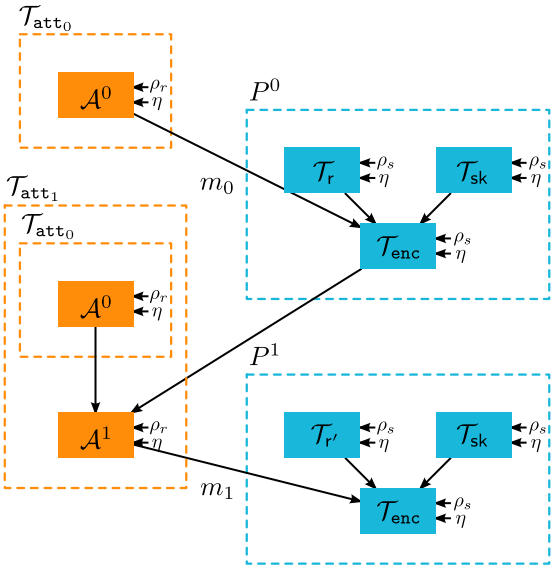

Figure 3: Final BC simulation of attacker/protocol interactions. Note the recomputation of, e.g., $\mathcal{T}_{\mathtt{att}_0}$.

We can consider a protocol P as a sequence of operations split by **in** (receive) operations. The attacker's operations interact with the protocol sequence: a protocol operation P^i typically uses **out** to send messages, providing more knowledge to the attacker, which can then use this knowledge to produce a message for P^{i+1}. E.g., considering Example 1, we define $\mathcal{T}_{\mathtt{att}_0}$ s.t. it simulates $\mathcal{A}^0$ and returns m_0, and $\mathcal{T}_{\mathtt{att}_1}$ s.t. on input u it first simulates $\mathcal{A}^0$ to get ϕ_0, and then runs $\mathcal{A}^1$ on ϕ_0 and u.

Still considering P from Example 1, with the corresponding syntactic frame Eq. (2), we depict in Fig. 3 the simulation that produces the same result as the real-world interaction.

2.3 Indistinguishability predicate and logical rules

Once equipped with an interpretation for terms, BC defines the predicate $\sim$ that will return true over two sequence of terms if the advantage of any final distinguisher is negligible in η. As for any other attacker computation step, this final distinguisher does not inherit any states from the previous attacker calls, but it may recompute those states if it is given the same arguments.

Consider two protocols: one can build as previously described two sequences of terms that correctly capture all the behaviours of their interactions with an attacker. If there exists an attacker that can distinguish the two protocols, then there is a set of attacker Turing machines $\{\mathcal{T}_{\mathtt{att}_i}\}$ that shows that the simulation of the two sequences of terms can be distinguished, and thus that $\sim$ does not hold over them. This constitutes the soundness of the BC logic.

Finally, one needs to prove the soundness of the logical reasoning rules by using reductions. For instance, a simple reduction would prove that a rule such as IF-F from Example 3 is indeed a valid rule to reason about $\sim$.

The rules presented in this example can be proven sound for any Turing machine attacker, without limiting its computational power. Such rules form the core of the logic, and are referred to as structural rules. In addition, one also needs to integrate the classical cryptographic assumptions inside the logic.

2.4 Cryptographic assumptions

In BC, cryptographic assumptions are encoded as rules (also referred to as axioms). BC includes provenly-sound rules to encode assumptions such as PRF, IND-CCA, EUF-CMA, ENC-KP, INT-CTXT, OTP, and DDH. Given a cryptographic assumption, one can translate it into a BC rule using terms: proving soundness of such a rule is done by providing a reduction showing that if there exists a model that breaks the BC assumption, then there exists an attacker that breaks the original cryptographic assumption. As the translation is often natural, the soundness proofs of the axioms are relatively straightforward, and are usually fully black-box reductions. We refer the reader to [7, 49] for the soundness proofs of the rules corresponding to the main cryptographic assumptions.

Example 4 (DDH assumption)**.** *In* BC *proofs, the full* DDH *assumption is not usually made: the encoded assumption can be seen as a generic version that is agnostic about the implementation. Concretely, one can see the* DDH *assumption of* BC *as a generic assumption over two function symbols, an extractor* $\mathtt{ext}$ *and a combiner* $\mathtt{comb}$*, such that, for any names* $\mathsf{n}_1, \mathsf{n}_2, \mathsf{n}_3$ *we have*

$$\begin{aligned} &\big(\mathtt{ext}(\mathsf{n}_1),\ \mathtt{ext}(\mathsf{n}_2),\ \mathtt{comb}(\mathtt{ext}(\mathsf{n}_1), \mathsf{n}_2),\ \mathtt{comb}(\mathtt{ext}(\mathsf{n}_2), \mathsf{n}_1)\big) \\ \sim\ &\big(\mathtt{ext}(\mathsf{n}_1),\ \mathtt{ext}(\mathsf{n}_2),\ \mathtt{ext}(\mathsf{n}_3),\ \mathtt{ext}(\mathsf{n}_3)\big). \end{aligned}$$

For a cyclic group with generator g*, the extractor function would be* $\mathtt{ext}(n) := g^n$ *and the combiner* $\mathtt{comb}(t, u) := t^u$*, and the* DDH *over the group assumption implies our generic assumption over the functions.*

Example 5 (OTP assumption)**.** *For the exclusive-or operator* $\oplus$*, we only assume that it is a binary function symbol such that for any term* t *and fresh name* n*, it allows for a one-time pad, i.e.,* $\big((t \oplus \mathsf{n}) \sim \mathsf{n}\big)$*.*

For any security proof of a protocol that uses $\oplus$*, one typically does not need to specify that* $\oplus$ *may be associative and commutative. Such a proof then gives us security guarantees, regardless of whether the concrete implementation of* $\oplus$ *is associative or not.*

To summarize, for proving with BC the full (computational) security of a protocol that uses concretely instantiated primitives, we first require a proof that the primitives instantiate cryptographic assumptions. Then, we use a translation of those assumptions in the BC proof to prove properties of the protocol. For example, to give a computational proof of a signed Diffie-Hellman protocol that uses Ed25519 signatures, we would use the proof that Ed25519 signatures satisfy EUF-CMA, and then use the corresponding EUF-CMA rule in the BC proof.

Note that it is also not uncommon to leave this question unanswered: i.e., to propose a protocol that relies on an assumption (e.g., PRF), for which we do not know an instantiation that provably meets it, and the guarantee is then phrased as "assuming that primitive X is a PRF, ...". Such a proof can still be meaningful since it proves the absence of a class of attacks, or anticipates a future provable instantiation.

2.5 The Squirrel Prover

The Squirrel Prover mechanizes the BC logic, and offers additional features not present in BC. Two downsides of the original BC logic, which are solved by Squirrel, are:

1. to prove the security of a protocol, one must make a proof for each possible action orderings of the protocol, and
2. the logic only considers finite protocols, and therefore notably only a bounded number of sessions.

As the security proof of a protocol may be very similar for many action orderings, which are also called (symbolic) traces, the first point implies a lot of tedious repetitions inside proofs. The second point can be solved by performing induction over the number of sessions, but this cannot be done inside the BC logic, and instead requires external mathematical reasoning.

2.5.1 Protocol Specification

The Squirrel prover's protocol specification language is close to the well-known applied pi-calculus. In Fig. 4 we give a simplified version of a key exchange protocol based on asymmetric encryption. An Initiator with secret and public keys (sk_I, pk_I) sends some encrypted ephemeral secret k_I to a responder with keys (sk_R, pk_R), that answers with its own encrypted ephemeral share k_R. Then, a shared key is derived as $h(s, k_I) \oplus h(s, k_R)$, where h is a PRF hash function and s a public seed. The derived key should only be computable by the intended peer. The protocol does not offer any explicit authentication. A possible implicit authentication guarantee could be: If a party X derives a key k with intended peer Y, then whoever can also compute the key must be Y, and agree on the communication partners.

A basic model of this protocol is expressed in Squirrel as illustrated in Listing 1. The `hash` and `aenc` instructions declare function symbols that respectively satisfy either the PRF assumption or the IND-CCA assumption, and the `name` command declares fresh values. In the process declarations, `out` and `in` specify outputs and inputs, `new` specifies sampling fresh values, and `|` is for parallel composition. In the last step of the Initiator, the instruction `sIR:=` is used to store the derived key in a state. The last line of this example captures a security property, which we discuss in next Section.

Note that the threat model is part of the input file, and Squirrel can be used to verify many scenarios. For instance, more complex versions of this running example that will be discussed in Section 6.3 can

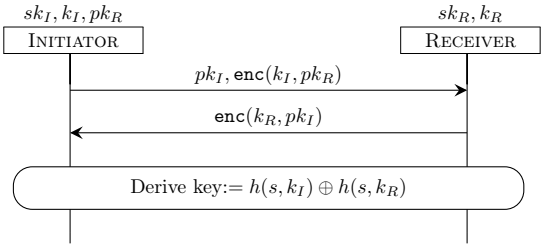

Figure 4: Encryption based key exchange

include an unbounded number of sessions and participants, and $\mathtt{pk}(sk_I)$ may not be hard-coded inside the Responder process but received as an input.

```
hash H
aenc enc,dec,pk
name s,skI,skR,kIR : message

process Initiator =
 new kI; new rI;
 out(cI,  enc(kI, rI,pk(skR)) );
 in(cR,ctR);
 let kR = dec(ctR,skI) in
 sIR :=  H(s,kI) XOR H(s,kR).

process Responder =
   new kR;
   new rR;
   in(cI, ctI);
   out(cR, enc(kR, rR, pk(skI))).

system [KE] out(cI,s); (Responder | Initiator).

equiv [KE] forall t, frame@t,sIR ~ frame@t,kIR.
```

Listing 1: Simplified key exchange in SQUIRREL

2.5.2 The SQUIRREL meta-logic

The SQUIRREL paper [5] introduces a meta-logic that introduces universal quantification over the possible traces of a protocol, as well as its number of sessions. As one can then reason at an abstract level on the traces, a single proof in the meta-logic covers multiple traces of the protocol, and it is now possible to consider unbounded protocols, thus solving both previous issues.

To do so, the logic is extended with timestamp variable τ to quantify over possible points in a trace, and macros that depend on a protocol P, such as:

- $\mathsf{input}^P@\tau$, to refer to the input received by the protocol P at timestamp τ;
- $\mathsf{output}^P@\tau$ for the output;
- $\mathsf{frame}^P@\tau$ for the full frame for the protocol up to this point, which is the sequence of all previous outputs.

If we denote by $\mathsf{pre}(\tau)$ the function that points to the previous timestamp in the trace, and abstracting away some details, $\mathsf{frame}^P@\tau$ would expand to $\mathsf{frame}^P@\mathsf{pre}(\tau), \mathsf{output}^P@\tau$.

The indistinguishability of two protocols P and Q can be expressed in a single meta-logic formula $\mathsf{frame}^P@\tau \sim \mathsf{frame}^Q@\tau$. This meta-logic formula then holds if for all possible traces of the protocol, and all possible τ over those traces, the BC frames of the protocol are indistinguishable. In the example of the *KE* protocol of Listing 1, the last line declares the security goal expressing the fact that the key derived by the initiator is indistinguishable from some fresh value k_{IR}, even given all the messages sent over the network. The proof in SQUIRREL of this goal will be described at a high-level in Section 6.3.3.

We remark however that even though this implies a proof of security for each possible number of sessions, it does not imply the classical security for an unbounded number of sessions because no concrete security bounds are obtained. Over this meta-logic, there are two sets of rules. The first allows proving that some probabilistic statement is true with overwhelming probability, and the second is for proving indistinguishability properties. These two sets interact: for instance, one can first prove authentication properties of a key exchange using rules from the first set, and then rely on these properties to prove the secrecy of the key using the second set.

2.5.3 SQUIRREL's mechanization

The SQUIRREL prover is available at [1], along with the code of the running example and the case studies. It is an interactive prover with some built-in tactics that simplify low-level reasoning.

From a high-level point of view, most proofs inside SQUIRREL follow the following schema:

1. prove a set of probabilistic statements inside the dedicated prover, such as matching conversations, or that some bad states cannot be reached;
2. prove the desired indistinguishability, using the probabilistic statements as lemmas.

The second step often relies on an induction over the length of the trace. Concretely, the proof is usually done by assuming that $\mathsf{frame}^P@\mathsf{pre}(\tau) \sim \mathsf{frame}^Q@\mathsf{pre}(\tau)$ is true, and proving that $\mathsf{frame}^P@\tau \sim \mathsf{frame}^Q@\tau$ holds. By definition of the frame, this is equal to proving that

$$\mathsf{frame}^P@\mathsf{pre}(\tau), \mathsf{output}^P@\tau \sim \mathsf{frame}^Q@\mathsf{pre}(\tau), \mathsf{output}^Q@\tau$$

and making a case distinction over possible values of output.

As a concrete example: Fig. 4 in SQUIRREL takes 20 lines to model, 20 lines of proof, and is then proved by SQUIRREL in under a second.

3 Adapting the BC logic and SQUIRREL to the post-quantum world

Proofs in the BC logic as presented in the previous section do not guarantee security against quantum attackers.

Recall that computational proofs rely on (i) computational assumptions, and (ii) reductions. In the context of the BC logic, the computational assumptions are the easiest to deal with when going to the post-quantum world: if we know that no post-quantum secure instantiations exist for a particular assumption, we can no longer use it in proofs. For example, if a previous proof relied on an axiom that states that integer factorization is hard, the proof is unsound as we can no longer use this axiom in the post-quantum setting.

The complexity of adapting the reductions that follow from the logic mostly revolves around revisiting the assumptions around the term interpretation and the proof rules, which implicitly encode attacker manipulations. Where previously the attacker was modeled using polynomial time Turing machines, we now need to change this to a quantum attacker.

It turns out that this modification is less straightforward than we expected. As explained previously, the classical BC logic uses a term representation that essentially corresponds to a tree in which computations, including those of the attacker, are performed in a purely local fashion. In the classical BC setting, this allows for modeling a single attacker using multiple local attackers: instead of explicitly communicating, subsequent local attackers effectively recompute the results of previous ones. This recomputation was possible since we made all operations deterministic and moved the probabilistic aspects (of both protocol and attacker) to two explicit random tapes. This modeling relies on a classical result for modeling probabilistic Turing machines as deterministic machines with a random tape.

However, for a quantum polynomial time attacker, we have no corresponding result. Notably, a quantum attacker may produce internal random state that we cannot influence nor extract. This phenomenon manifests itself in the so-called *no-cloning theorem* [65]: We cannot duplicate the quantum state of a quantum machine, nor can we run a quantum machine twice and expect to obtain the same inner state.

This directly breaks the stability of the previous interpretation, and for instance, the evaluation of Fig. 2 could now return false. Furthermore, in the interpretation of Fig. 3, we would not be able to recompute inside $\mathcal{T}_{\mathtt{att}_1}$ the same attacker state as the one computed during $\mathcal{T}_{\mathtt{att}_0}$, and therefore this

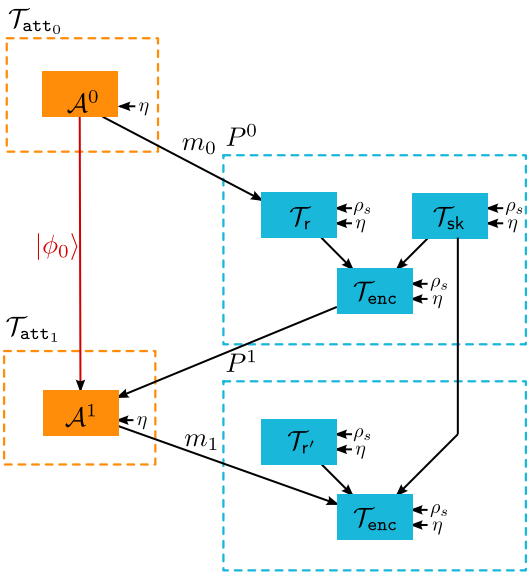

Figure 5: Quantum compatible simulation of attacker/protocol interactions without recomputation in PQ-BC. Compare this to Fig. 3.

evaluation would not correspond to the run of a single interactive quantum attacker, as was the case for classical attacker. Similarly, the classical BC interpretation also allows for *rewinding* the attacker, which is impossible for quantum attackers based on the same no-cloning theorem.

We solve these issues by defining an interpretation where the attacker is a single black-box interactive machine, as opposed to a set of one-shot Turing machines. This allows us to verify a set of conditions that when met, ensure that we can provide a sound interpretation for a quantum attacker. Concretely, to interpret $\mathtt{att}_0$, we query the interactive machine once, and to interpret $\mathtt{att}_1(t)$, we interpret t, and give it as input to the same interactive machine. This effectively changes the term interpretation from a tree to a directed acyclic graph. We will formally define such an interpretation later, but it would intuitively be pictured as in Fig. 5. Changing the term interpretation in this fashion leads to a wide range of subtle changes throughout the logic.

We provide three conditions to ensure that we can give a sound interpretation for a quantum attacker. We provide intuition for them below, and formally define them in Section 4.

Condition 1 - Consistency In the classical BC logic, the attackers need not be consistent. We provide a concrete example, that intuitively corresponds to the simulation of rewinding the attacker. For two distinct terms u, v, consider the sequence of terms $\mathtt{att}_1(\mathtt{att}_0(), u)$, $\mathtt{att}_1(\mathtt{att}_0(), v)$. This corresponds to running $\mathtt{att}_0$, running its continuation with u, and then making it forget that we ever gave it u (rewind the attacker) and calling it on v.

Against a quantum attacker or black-box attacker, we cannot do this operation, as it morally implies that we are duplicating the attacker's inner state at the end of $\mathtt{att}_0$, using each copy to run $\mathtt{att}_1$ once on u and once on v.

We therefore add a syntactic condition on terms to forbid such cases, and force that all occurrences of $\mathtt{att}_i$ are made with the same arguments.

Condition 2 - Monotonicity With the new term interpretation, $\mathtt{att}_0$ and $\mathtt{att}_1$ implicitly share states. This means that when interpreting $\mathtt{att}_1$, we will always get an answer that depends on the argument previously given to $\mathtt{att}_0$.

Consider for example the terms:

$$\mathtt{att}_0(\mathsf{n}),\ \mathtt{att}_1() \doteq \mathsf{n}$$

In the classical BC logic, the second term would evaluate to false, as $\mathtt{att}_1()$ does not depend on the distribution of n. But if we are forced to interpret the symbols using a black box attacker, the computation of $\mathtt{att}_1()$ does depend on n. Thus, we lose the locality of our logic on some terms, that we will need to restore by further reducing the set of terms. This result in a second syntactic condition, where the arguments of the successive $\mathtt{att}_i$ must form a growing sequence of terms.

Condition 3 - Balance In the classical BC logic, we can write formulas that do not model interactions between a single attacker with a protocol, and for instance prove indistinguishability formulas that would be trivially false against a single attacker, but true for an independent set of attackers. For instance, consider the formula $\left(\mathtt{att}_0() \doteq \mathsf{n}\right) \sim \left(\mathtt{att}_1(\mathtt{att}_0(), \mathsf{n}') \doteq \mathsf{n}\right)$. Both sides evaluate to false, because the attacker is completely independent of n (this is the =IND rule). In the classical BC logic, the final distinguisher is also a disjoint machine, which typically does not know how many other attacker machines were executed: therefore, the formula is true in classical BC. This reflects that classical BC allows modeling a weaker threat model with independent attackers. In the PQ-BC interpretation, the final attacker will be a final call to the interactive machine, which of course knows how many times it was previously called.

Thus, under the new and strictly stronger interpretation with a single attacker, some formulas that held under the classical BC logic might no longer hold. We therefore have to ensure that none of our logical rules allows for deriving such formulas. We implement this by introducing a syntactic condition for PQ-BC that essentially requires that the number of attacker calls is balanced, i.e., the number of calls is equal on both sides of an indistinguishability operator, and which will be a side condition of all our logical rules.

Design choices for the conditions While the above three conditions solve issues in designing a post-quantum sound BC logic, they were additionally chosen because they also form a small sufficient set of conditions to derive a usable PQ-BC logic, as we prove in the following section. It would have been possible, for instance, to replace the balance condition by a specific and more refined condition for each logical rule, that as a set would have been equivalent to the balance condition. However, the balance condition is both necessary and more generic, and hence we decide to use this one within PQ-BC. Overall, the three conditions allow for a generic implementation in the SQUIRREL prover, and yield a logic usable in practice.

4 PQ-BC: A Post-Quantum BC logic

In this section we first provide the core of the formal definition of the PQ-BC logic: its term interpretation that is suitable for post-quantum attackers, as well as the consistency and monotonicity conditions needed to ensure the stability and locality of the logic. This interpretation allows to consider a single interactive attacker, rather than a set of single-one shot attackers. We then prove the computational soundness of the logic with the new interpretation. This implies that the logic can be used to obtained computational guarantees against quantum attackers. Second, we provide the structural rules of the logic, as well as the balance condition needed to prove the rules sound in the post-quantum setting. Finally, we discuss which cryptographic axioms - and thus which corresponding BC axioms - can be used inside the logic to get post-quantum soundness of a protocol analysis.

While we directly refer to a quantum attacker in definitions and proofs, we actually design a logic with an interpretation and rules that are computationally sound for any interactive black-box attacker. The attacker can be instantiated as a Polynomial Time attacker, or a Quantum Polynomial Time attacker, or even an unbounded Turing machine attacker. It is only the cryptographic assumptions used inside a given proof that restrict the attacker's computational power.

4.1 Syntax and Semantics

We use terms to model random samplings, public function computations by honest parties, and black-box attacker computations. For random samplings, the BC logic inherits some conventions from the Pi calculus: notably, fresh values (such as nonces) are called *names* (and have nothing to do with identities); by convention, variables called $\mathsf{n}, \mathsf{n}', \ldots$ are names and hence freshly generated values. In Example 1 r, r', and sk would also be modeled as names. Let $\mathcal{N}$ be a set of names. Names can be seen as fixed identifiers, where each is a pointer to a uniformly sampled bitstring. Let Σ be a set of function symbols, the set used for public functions and primitives. Let $\{\mathtt{att}_i \mid i \in \mathbb{N}\}$ be a set of function symbols such that $\mathtt{att}_i$ is of arity i for each $i \in \mathbb{N}$.

Definition 1. *We consider terms built according to the syntax:*

$$
\begin{array}{rcll}
t & ::= & \mathsf{n} \in \mathcal{N} & \textit{name (fresh value)} \\
 & | & f(t_1, \ldots, t_k) & \textit{function symbol } f \in \Sigma \\
 & | & \mathtt{att}_i(t_1, \ldots, t_i) & \textit{i-th attacker call}
\end{array}
$$

We write $\vec{t}$ or $t_1, \ldots, t_n$ for sequences of terms. This notation is not a construction inside the terms, and is thus different to defining a function symbol to model tuples inside terms.

4.1.1 Functional model

Recall that while we consider a quantum attacker, we model honest protocol participants as classical Polynomial Time Turing Machines (PTTMs). To interpret terms, we introduce the notion of a *functional model* $\mathcal{M}_f$, a library implementing the public function symbols and names that are used in the protocol: for each function symbol f (encryptions, signatures,...), $\mathcal{T}_f$ is a PTTM, which we view as a deterministic machine with an infinite random tape and taking the security parameter as input. The functional model also contains a PTTM $\mathcal{T}_n$ for each $n \in \mathcal{N}$, which will extract from the random tape a bitstring of length η. We give η in unary to the PTTMs, as they are expected to be polynomial time w.r.t. η in the computational model.

Definition 2. *A* functional model $\mathcal{M}_f$ *is a set of PTTMs, one for each name and symbol function, such that:*

1. *if* $\mathsf{n} \in \mathcal{N}$ *(i.e.,* n *is a name),* n *is associated to the machine* $\mathcal{T}_n$ *that on input* $(1^\eta, \rho_s)$ *extracts a word of length* η *from the tape* ρ_s*. Different names extract disjoint parts of the random tape.*
2. *if* $f \in \Sigma$ *is of arity* n*,* f *is associated to a machine* $\mathcal{T}_f$ *which, on input* 1^η*, expects* n *more bitstrings, and does not use* ρ_s*.*

We model public functions as deterministic functions: if randomness is required, it should be given explicitly as an argument to the function symbol. This modeling is needed for the stability of the interpretation.

We can now define the basic interpretation of terms, assuming that we have been given the output bitstring corresponding to each attacker call. Based on this first interpretation, we will then define the one where there is an actual attacker.

Definition 3. *Given a functional model* $\mathcal{M}_f$*, the security parameter* η*, a mapping* σ *from terms* $\mathtt{att}_i(\phi)$ *to bitstrings, and an infinite sequence of bitstrings* ρ_s*, we define the interpretation of terms such that all occurrences of* $\mathtt{att}_i(\phi_i)$ *are in the domain of* σ *as:*

$$\begin{array}{lll} [\![\mathsf{n}]\!]^{\eta,\sigma}_{\mathcal{M}_f,\rho_s} & := \mathcal{T}_n(1^\eta, \rho_s) & \text{, if } n \in \mathcal{N} \\ [\![f(\vec{u})]\!]^{\eta,\sigma}_{\mathcal{M}_f,\rho_s} & := \mathcal{T}_f([\![\vec{u}]\!]^{\eta,\sigma}_{\mathcal{M}_f,\rho_s}) & \text{, if } f \in \Sigma \\ [\![\mathtt{att}_i(\vec{u})]\!]^{\eta,\sigma}_{\mathcal{M}_f,\rho_s} & := \mathtt{att}_i(\vec{u})\sigma & \text{, for all } i \end{array}$$

We assume that the functional model contains function symbols that expresses propositional formulas, which are interpreted as expected. We denote those connectives by the $\dot{=}, \dot{\wedge}, \dot{\Rightarrow}, \ldots$ – note these are marked with a dot.

We will ultimately use two different sets of logical connectives: (1) the dot variants, used in terms, and (2) the variants without a dot that are part of the logic we are building. We will illustrate their combination in Example 7.

4.1.2 Computational Model

To define the interpretation of terms with attacker function symbols, we view terms as directed acyclic graphs from leaves to their root.

Example 6. *Consider a variant of Example 1, where we denote tuples using* $\langle \ldots \rangle$*:*

$$\begin{array}{l} P := \mathbf{new}\ \mathsf{sk}.\ \mathbf{in}(c, x).\ \mathbf{new}\ \mathsf{r}.\ \mathbf{out}(c, \mathtt{enc}(x, \mathsf{r}, \mathsf{sk})). \\ \qquad \mathbf{in}(c, y).\ \mathbf{new}\ \mathsf{r}'.\ \mathbf{out}(c, \mathtt{enc}(\langle y, x, y\rangle, \mathsf{r}', \mathsf{sk})). \end{array}$$

In the second step, the protocol encrypts the tuple made with twice the second protocol input on and once the first input.

The frame corresponding to this protocol would be t_0, t_1*:*

- $t_0 := \mathtt{enc}(\mathtt{att}_0(), \mathsf{r}, \mathsf{sk})$
- $t_1 := \mathtt{enc}(\langle \mathtt{att}_1(t_0), \mathtt{att}_0(), \mathtt{att}_1(t_0)\rangle, \mathsf{r}', \mathsf{sk})$

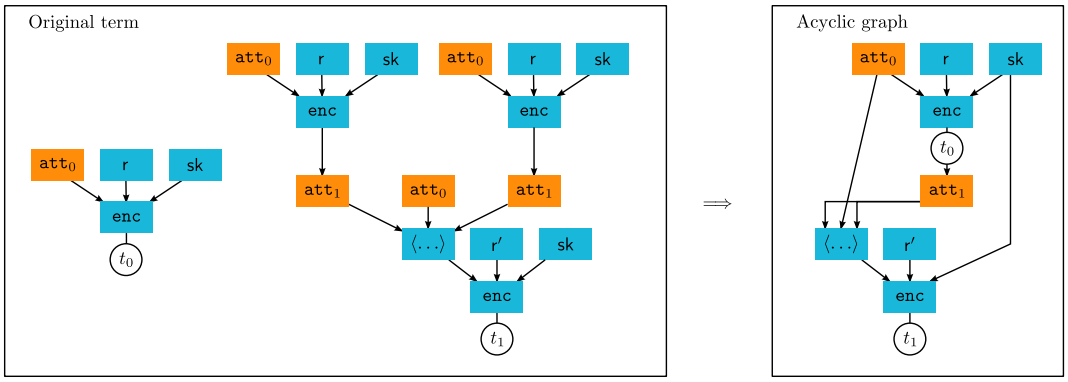

Figure 6: From the classical BC to the acyclic graph representation in PQ-BC

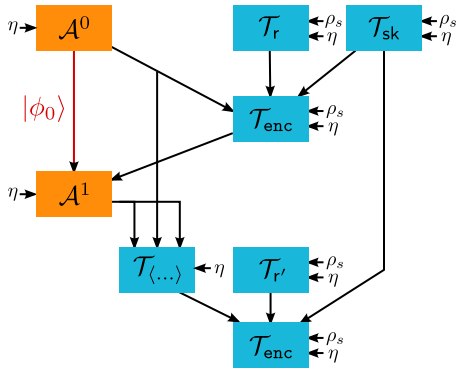

Figure 7: Quantum compatible interpretation in PQ-BC

We give the original terms and the acyclic graph variant for this frame in Fig. 6.

The acyclic representation leads to a natural interpretation that we can execute even when we are only given access to a black-box straight line attacker, for instance a sequence of $\{C^i\}$ as in Fig. 12. We illustrate this interpretation in Fig. 7. An interesting side effect of this be interpretation, is that the computation time of the simulator actually matches the original execution time of the attacker and the protocol in the real-world. This was not at all the case with the original interpretation of the BC logic, where for instance the interpretation t_0 and t_1 of Example 6 would have executed four times $\mathcal{A}^0$ and two times $\mathcal{A}^1$. When we model the interactions with an interactive attacker, this corresponds to the high-level change between Fig. 3 and Fig. 5.

We assume that the attacker is an oracle Quantum Turing Machine (QTM) to obtain post-quantum soundness. We provide the formal definition of such machines in Appendix A For our purpose here, it suffices to know that such a machine behaves as a quantum computer that interactively performs oracle queries and receives the answers. Importantly, the oracle queries and answers are classical bitstrings, and do not contain any quantum state. This models a quantum computer interacting over a network with a classical protocol. One can also abstract such an attacker as a straight-line black-box interactive process, which is what we do in most of our proofs.

Definition 4. *A* computational model $\mathcal{M}^{\mathcal{A}}$ *is an extension of a functional model* $\mathcal{M}_f$*, which provides an additional oracle QTM* $\mathcal{A}$ *that takes as input a security parameter* 1^η*.*

Given $\mathcal{M}^{\mathcal{A}}$*,* η*,* σ*,* ρ_s *and* ρ_r*, we define the interpretation* $[\![t]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s}$ *of a term* t *as:*

1. *First, evaluate* $\mathtt{att}_0$*, by running* $\mathcal{A}$ *on input* 1^η *until the first oracle query, and store the content of the oracle query tape* o *inside the substitution* $\sigma_0 : \{\mathtt{att}_0() \mapsto o\}$*.*

2. *Then, we assume that we have a substitution* σ_i *mapping all occurrences of* $\mathtt{att}_j$*,* $j < i$ *to a bitstring. We find the smallest subterm occurrence of a* $\mathtt{att}_l(t_1, \ldots, t_l)$ *in* t*. Then, for* k *from* i *to* l*, we write on* $\mathcal{A}$*'s oracle answer tape the value* $[\![t_k]\!]^{\eta,\sigma_{k-1}}_{\mathcal{M}_f,\rho_s}$*, then continue* $\mathcal{A}$ *and wait for the next oracle query, and store the content of the oracle query tape* o *inside the substitution* $\sigma_k = \sigma_{k-1} \cup \{\mathtt{att}_k(t_1, \ldots, t_k) \mapsto o\}$*.*

3. Finally, given σ_l, where l corresponds to the biggest occurrence of a $\mathtt{att}_l$, return $[\![t]\!]^{\eta,\sigma_l}_{\mathcal{M}_f,\rho_s}$.

This interpretation does not require that a term contains all intermediate calls to the attacker. Thus, interpreting the term $\mathtt{att}_1(t)$ or the sequence $\mathtt{att}_0(), \mathtt{att}_1(t)$ leads to the same interactions with the attacker.

4.1.3 Well-defined interpretation

Our previous definition is not defined over all possible terms: there may not be a unique smallest occurrence of a $\mathtt{att}_l$. This is expected, as there exist terms that correspond to experiments that are not realisable with a quantum attacker. Notably, recall that in this context, we cannot interpret the sequence $\mathtt{att}_0(), \mathtt{att}_1(u), \mathtt{att}_1(v)$. Indeed, if the attacker is straight-line and black-box, it means that we can only get one attacker answer corresponding to $\mathtt{att}_1$. Therefore, to ensure that a term corresponds to a valid experiment with respect to a quantum attacker, we require consistency: $\mathtt{att}_i$ should always occur with the same arguments in the terms, i.e., corresponds to the same unique call of this attacker's interactive step.

Definition 5 (Consistency)**.** *A sequence of terms $\vec{t}$ is consistent if all function symbol $\mathtt{att}_i$ occurs with the same arguments.*

Lemma 1. *$[\![\vec{t}]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s}$ is well-defined if $\vec{t}$ is consistent.*

Proof. Let $\mathcal{M}^{\mathcal{A}}$ be a computational model and $\vec{t}$ a consistent sequence of terms. We show that $[\![t]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s}$ is well-defined, i.e., that it corresponds to a possible evaluation given the interactive attacker $\mathcal{A}$ defined by $\mathcal{M}^{\mathcal{A}}$. Recall that $\mathcal{A}$ is seen as interacting with an oracle, where intuitively the oracle corresponds to the protocol execution, a query to the oracle is thus an input given by the attacker to the protocol, and the oracle answer is the protocol output.

1. σ_0 simply corresponds to executing $\mathcal{A}$ in a black-box fashion until it makes an oracle query. The content of the query is classical (non quantum), and we can thus store this inside σ_0.
2. We assume that σ_i has been computed, that $\mathcal{A}$ has made i oracles queries and is waiting for an answer, and that for all subterms t of $\vec{t}$ that only contain occurrences of $\mathtt{att}_k$ with $k \leq i$, $[\![t]\!]^{\eta,\sigma_i}_{\mathcal{M}_f,\rho_s}$ is well-defined. As $\vec{t}$ is consistent, there exist a unique smallest subterm occurrence of a $\mathtt{att}_l(t_1,\ldots,t_l)$ in $\vec{t}$ such that l is the smallest new index bigger than i. Then, $t_1,\ldots,t_l$ only contains occurrences of att_k with $k \leq i$ (otherwise l is not the smallest index bigger than i). By the induction hypothesis, we can thus compute the value of all $[\![t_k]\!]^{\eta,\sigma_i}_{\mathcal{M}_f,\rho_s}$, $1 \leq k \leq l$. We can thus continue the execution of $\mathcal{A}$, giving him as answer to its i-th oracle query $[\![t_{i+1}]\!]^{\eta,\sigma_i}_{\mathcal{M}_f,\rho_s}$, waiting for its next query and storing it inside σ_{i+1}, and so on until we submit $[\![t_l]\!]^{\eta,\sigma_i}_{\mathcal{M}_f,\rho_s}$ as answer, and it continues up to its next query, storing the content of the query in σ_l.

With 1) and 2), we thus have by induction that σ_l can be computed given an interactive quantum attacker, and we can then finally evaluate $[\![\vec{t}]\!]^{\eta,\sigma_l}_{\mathcal{M}_f,\rho_s}$, which is simply a classical computation (recall that all values in σ_l correspond to classical oracle queries made by $\mathcal{A}$. The interpretation is then well-defined. □

4.1.4 Restoring locality

In the evaluation of the sequence $\mathtt{att}_0(), \mathtt{att}_1(u), \mathtt{att}_2(v,v')$, the value of $\mathtt{att}_2(v,v')$ depends on u in the new interpretation. Yet, this is not reflected inside the term and introduces an implicit dependency that breaks some BC rules as it breaks locality. In particular, it breaks the =IND rule presented in Section 2. To repair it, we introduce the second term restriction.

Definition 6 (Monotonicity)**.** *A sequence of terms $\vec{t}$ satisfies the monotonicity restriction if for all occurrences of $\mathtt{att}_i(t_1,\ldots,t_i)$ and $\mathtt{att}_j(t'_1,\ldots,t'_j)$ with $i < j$, we have for all $1 \leq k \leq i$ that $t_i = t'_i$.*

Essentially, to restore locality, we require that attacker function symbols must always be called on at least the same set of argument as the previous attacker calls.

Note that in the definition of monotonicity, we do not specify that the sequence of attacker calls must be continuous w.r.t. to the indices. For instance, a sequence may contain $\mathtt{att}_1$ but no $\mathtt{att}_0$. It is essential for the later proofs of protocol, because we sometimes remove some attacker calls in the proof process, replacing them by some other values.

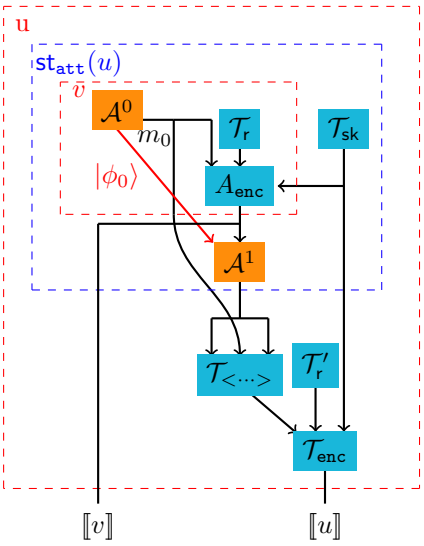

Figure 8:

Those restrictions notably allow to show a lemma that concretely express the locality of the interpretation. The probability that some u evaluates to a value is independent of what we put around of it.

Lemma 2. *Let $\vec{u}$ be a sequence of term. We have for any $\vec{v}$ such that $\vec{u}, \vec{v}$ is a well-formed sequence that for any $c \in \{0,1\}^*$, any attacker $\mathcal{A}$ and computational model $\mathcal{M}^{\mathcal{A}}$,*

$$\mathbf{Pr}\{[\![\vec{u}]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} = c\} = \mathbf{Pr}\{[\![\vec{u},\vec{v}]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} = c, _\}$$

Proof. In this proof, we rely on a notation defined slightly later in Definition 15.

To provide some intuition, we first provide an example based on the interpretation previously presented in Fig. 7. With the notations of Example 6, when interpreting either the sequence t_1, t_0 or the single element t_1, the distribution corresponding to t_1 will be the same. This is because t_0 can only reuse some computations of t_1. We illustrate this in Fig. 8. In general, we could also add a t_2 that would perform further attacker calls, but that would not influence the values returned by the previous attacker calls.

Formally, this holds by Definition 4. Indeed, recall that this Definition of interpretation can intuitively be seen as interpreting bottom-up the acyclic graph, and reusing the previously computed values when needed through the mapping σ. In this definition, we see that the value of $[\![\vec{u}]\!]$ is determined by the value given to in the interpretation $\sigma_{\mathsf{Max}_{\mathsf{att}}(\vec{u})}$. This value is fully determined by the attacker $\mathcal{A}$ and the set of syntactical attacker calls of $\vec{u}$. As $\vec{u}, \vec{v}$ is well-formed, it is notably a consistent, and thus the set of syntactical attacker calls of $\vec{u}, \vec{v}$ up to $\mathsf{Max}_{\mathsf{att}}(\vec{u})$ is equal to the one of $\vec{u}$. Thus, the distribution of $\sigma_{\mathsf{Max}_{\mathsf{att}}(\vec{u})}$ when interpreting $\vec{u}, \vec{v}$ is the same as the one of $\sigma_{\mathsf{Max}_{\mathsf{att}}(\vec{u})}$ when interpreting $\vec{u}$.

□

Interestingly, recall that in Section 2.1, when we described how to construct a syntactic frame that capture syntactically the attacker interactions with a protocol so that we can see all probabilistic dependencies syntactically, we concluded that we in fact needed to satisfy the monotonicity and consistency restrictions. At the time, we did not express it as a general restriction on terms, but only as a specification of the translation from protocols to terms. Our terms were more expressive, and it was not an issue because we could indeed give a meaning to all sort of terms, either corresponding to a frame or to anything else. In the quantum world, we cannot do so anymore, and we thus directly consider the set of terms that we can consider. But as we want to perform proofs of security protocols, we only care about writing terms that correspond to protocols. And thus, our restrictions will not restrict the proving capabilities of the previous BC logic.

From now on, we only allow in the logic terms that satisfy the consistency and the monotonicity properties.

4.1.5 Interpretation of formulas

Atomic formulas of the logic are built using a set of predicate symbols $\sim_n$ of arity $2n$. Given terms $t_1, \ldots, t_n, s_1, \ldots, s_n$, the predicate $\sim_n (t_1, \ldots, t_n, s_1, \ldots, s_n)$ will be interpreted as computational indistinguishability between the two sequences of terms. We use infix notation, and always omit n as it is clear from the context, thus denoting the previous equivalence by $(t_1, \ldots, t_n) \sim (s_1, \ldots, s_n)$. The first order formulas are then built using the usual logical connectives $\vee, \wedge, \top, \bot, \Rightarrow, \exists, \forall, \neg$. Note that these connectives are not marked with a dot, and are part of the logic, not the terms.

Example 7. *Given terms u, v, t, we can write the formula:*

$$\big((u \mathrel{\dot{\Rightarrow}} v) \sim t\big) \Leftrightarrow \big((\dot{\neg}\, u \mathrel{\dot{\vee}} v) \sim t\big)$$

This formula holds because we assume that for all functional models, $\dot{\Rightarrow}, \dot{\neg}$ and $\dot{\vee}$ are classically interpreted.

To define a distinguisher between sequences of terms, we must define this distinguisher as a continuation of the interactive attacker $\mathcal{A}$, and thus pass to it the quantum state of $\mathcal{A}$. This is only a technicality, and in practice we simply consider that the continuation is the last stage of the interactive attacker.

Definition 7. *Let $\vec{v}$ be a sequence of term. For any computational model $\mathcal{M}^{\mathcal{A}}$, we denote by $\phi^{\vec{v}}_{\mathcal{M}^{\mathcal{A}},\rho_s,\eta}$ the final (quantum) configuration reached by $\mathcal{A}$ during a computation of $[\![\vec{v}]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s}$.*

Definition 8. *Given a computational model $\mathcal{M}^{\mathcal{A}}$, and two sequences of ground terms $\vec{t}$, $\vec{u}$, the formula $\vec{t} \sim \vec{u}$ is satisfied by $\mathcal{M}^{\mathcal{A}}$, denoted by $\mathcal{M}^{\mathcal{A}} \models \vec{t} \sim \vec{u}$, if, for every polynomial time QTM $\mathcal{B}$,*

$$\begin{aligned}|&\mathbf{Pr}_{\rho_s}\{\mathcal{B}([\![\vec{t}]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s}, 1^{\eta}, \phi^{\vec{t}}_{\mathcal{M}^{\mathcal{A}},\rho_s,\eta}) = 1\} \\ &-\mathbf{Pr}_{\rho_s}\{\mathcal{B}([\![\vec{u}]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s}, 1^{\eta}, \phi^{\vec{u}}_{\mathcal{M}^{\mathcal{A}},\rho_s,\eta}) = 1\}|\end{aligned}$$

is negligible in η. Here, ρ_s is drawn according to a distribution such that every finite prefix is uniformly sampled, and the probabilities also depend on the inherent probabilistic behavior of the two QTMs. The satisfaction relation is extended to full first-order logic as usual.

At a high level, the proof follows naturally from our previous design of a faithful interpretation of terms that is sound for black-box interactive attackers. We directly obtain that by quantifying over all possible $\mathcal{M}^{\mathcal{A}}$, the value of $[\![\vec{t}]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s}$ describes all possible behaviours of the protocol modelled by t interacting with any black-box interactive attacker. Thus, if there exists an attack on the protocol in the real world, it will correspond to an attack on the (computational) interpretation of the terms. Furthermore, because Definition 8 exactly quantifies universally over all $\mathcal{M}^{\mathcal{A}}$, if there is an attack on the protocol, the predicate is not valid, and if there is none, it is valid. Finally, because an interactive black-box attacker soundly models a quantum attacker, the logic is shown to be post-quantum sound.

4.1.6 Overwhelming probabilistic truth

The classical BC logic as well as Squirrel has a subset of its rule dedicated to proving the validity of statements of the form $u \sim \mathsf{true}$.

In our case the attacker $\mathcal{A}$ from the computational model and the final distinguisher $\mathcal{B}$ share their (quantum) tape, while in the classical BC definition [7], $\mathcal{A}$ and $\mathcal{B}$ do not share their working tape, but only their source of randomness Thus, compared to the classical BC definition, as soon as u executes an attacker machine, $(u \doteq u) \sim \mathsf{true}$ will not hold, as $\mathcal{B}$ simply checks if an attacker machine was executed.

Note that in the classical BC logic, a proof that $u \sim \mathsf{true}$ is in fact a statement independent of the final distinguisher: we just prove that u is equal to true with overwhelming probability. This is what we call a probabilistic statement.

For PQ-BC, we thus define a predicate dedicated to proving overwhelming probabilistic truth, for which we inherit all the truth rules of BC and Squirrel.

Definition 9. *Given a computational model $\mathcal{M}^{\mathcal{A}}$ and a ground term t, $\mathbb{T}(t)$ is satisfied by $\mathcal{M}^{\mathcal{A}}$, denoted by $\mathcal{M}^{\mathcal{A}} \models \mathbb{T}(t)$, if,*

$$\mathbf{Pr}_{\rho_s}\{[\![\vec{t}]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} = 1\}$$

is overwhelming in η. The satisfaction relation is extended to full first-order logic as usual.

5 Post-Quantum soundness

In order to prove the soundness of our logic, we must first define the classical computational indistinguishability of protocols.

5.1 Protocols

Our building blocks for protocols are actions. An action essentially models a single protocol step of an input and output that can occur under a condition. The attacker specifies to which actions he wishes to send a value by giving a dedicated channel identifier c from a fixed channel set $\mathcal{C}$.

Definition 10. *A protocol term is a term without attacker function symbols and built over an infinite set of variables $\mathcal{X}$.*

An action $A = (c^A, x^A, \phi^A, f^A)$ is given by a channel identifier $c \in \mathcal{C}$, a variable $x^A \in \mathcal{X}$, and two protocol terms ϕ^A, f^A.

Intuitively, action A captures the protocol $\mathbf{in}(c, x^A); \mathbf{if}\ \phi^A\ \mathbf{then\ out}(c, f^A)\ \mathbf{else\ out}(c, \mathbf{0})$. We can then define a protocol as a set of such actions, and by adding sequential dependencies between some actions.

Definition 11. *A protocol is given by a set of actions $\{A_i\}$ and a partial-order $<$ over actions, such that for all i, $fv(f^{A_i}) \cup fv(\phi^{A_i}) \subset \{x^B | B < A\}$.*

This definition simply means that some actions must occur before some others, and an action output and condition may only depend on the input variables of the actions that it depends on.

We now define, given a protocol, the *protocol oracle* that models the protocol behaviour.

Definition 12. *Given a functional model $\mathcal{M}_f$, a protocol $P = \{A_i\}$, a security parameter η and an infinite secret random tape ρ_s, we define the protocol oracle $\mathcal{O}_P$ as the stateful oracle that:*

1. *initializes an empty substitution σ and an empty set of actions E;*
2. *on input (c, m), it fetches the action $A \in P$ such that $c^A = c$, update σ to $\sigma \cup \{x^A \mapsto m\}$ and $E = E \cup \{A\}$ and*
   - *if $\{B | B < A\} \subset E$ and $[\![\phi^A]\!]^{\eta,\sigma}_{\mathcal{M}_f,\rho_s}$ evaluates to true, returns $[\![f^A]\!]^{\eta,\sigma}_{\mathcal{M}_f,\rho_s}$;*
   - *else returns 0.*

In this definition we assume that there is a single action $\mathcal{A} \in P$ that corresponds to the given channels. Such protocols are called *action deterministic*, and we will always assume that we are given such protocols.

5.2 Indistinguishability

With the definition of a *protocol oracle*, we can define in the expected sense the post-quantum indistinguishability of two protocols:

Definition 13. *Given a functional model $\mathcal{M}_f$ and protocols P, Q, we write $P \cong Q$ if for every QTM $\mathcal{A}$, the attacker's advantage* $\mathrm{Adv}^{P \cong Q}$ *equal to*

$$|\mathbf{Pr}_{\rho_s}\{\mathcal{A}^{\mathcal{O}_P}(1^\eta) = 1\} - \mathbf{Pr}_{\rho_s}\{\mathcal{A}^{\mathcal{O}_Q}(1^\eta) = 1\}|$$

is negligible in η.

5.3 Soundness

We first formally define the set of frames of a protocol, that is the set of sequence of terms that will capture all the protocols behaviours. We define one such sequence of terms for each possible sequence of actions. Fixing the sequence of actions essentially fixes the sequence of channel identifiers sent by the attacker.

Definition 14. *Let P be a protocol. A scheduling $\tau = A^1, \ldots, A^n$ of the protocol is an executable sequence of protocols actions.*

Given a scheduling $\tau = A^1, \ldots, A^n$, we define inductively the frame ψ^τ_P as:

- $\psi_P^A :=$ **if** $\phi^A\{x^A \mapsto \mathtt{att}_0()\}$ **then** $f^A\{x^A \mapsto \mathtt{att}_0()\}$ **else 0**;
- $\psi_P^{A^1,\ldots,A^n} :=$ $\psi_P^{A^1,\ldots,A^{n-1}}$, **if** $\phi^A\{x^A \mapsto \mathtt{att}_{n-1}(\psi_P^{A^1,\ldots,A^{n-1}})\}$ **then** $f^A\{x^A \mapsto \mathtt{att}_{n-1}(\psi_P^{A^1,\ldots,A^{n-1}})\}$ **else 0**

Lemma 3. *For any P and τ, the sequence ψ_P^τ satisfies the consistency and monotonicity properties.*

Proof. Direct by definition. □

Based on those definitions, we now prove the post-quantum computational soundness of the logic.

Lemma 4. *Given two simple protocols P, Q with the same set T of schedulings, a random tape ρ_s and a functional model $\mathcal{M}_f$, we have:*

$$\forall \tau \in T, \forall \mathcal{M} \supset \mathcal{M}_f. \quad \mathcal{M}^{\mathcal{A}} \models \psi_P^\tau \sim \psi_Q^\tau$$
$$\Rightarrow$$
$$P \cong Q$$

Proof. Let us proceed by contradiction. We are given a distinguisher $\mathcal{B}$ against $P \cong Q$. We thus have,

$$\mathrm{Adv}_{\mathcal{B}}^{P\cong Q} = |\mathbf{Pr}_{\rho_s}\{\mathcal{B}^{\mathcal{O}_P}(1^\eta) = 1\} - \mathbf{Pr}_{\rho_s}\{\mathcal{B}^{\mathcal{O}_Q}(1^\eta) = 1\}|$$

is non-negligible. We must find τ and $\mathcal{M}^{\mathcal{A}}$ such that $\mathcal{M}^{\mathcal{A}} \not\models \psi_P^\tau \sim \psi_Q^\tau$

5.4 Find a trace τ

We split this probability, by conditioning over the observable traces. We denote by E_τ the event "τ is the scheduling produced by $\mathcal{B}$".

By dichotomy, we have

$$\begin{aligned}\mathrm{Adv}_{\mathcal{B}}^{P\cong Q} &= |\mathbf{Pr}_{\rho_s}\{\mathcal{B}^{\mathcal{O}_P}(1^\eta) = 1\} - \mathbf{Pr}_{\rho_s}\{\mathcal{B}^{\mathcal{O}_Q}(1^\eta) = 1\}| \\ &= \textstyle\sum_{\tau\in T} \mathbf{Pr}_{\rho_s}\{E_\tau\}\times \\ &\quad |\mathbf{Pr}_{\rho_s}\{\mathcal{B}^{\mathcal{O}_P}(1^\eta) = 1 \mid E_\tau\} - \mathbf{Pr}_{\rho_s}\{\mathcal{B}^{\mathcal{O}_Q}(1^\eta) = 1 \mid E_\tau\}|\end{aligned}$$

We of course have that for any τ, $\mathbf{Pr}_{\rho_s}\{E_\tau\} \leq 1$. We thus obtained the following upper-bound.

$$\mathrm{Adv}_{\mathcal{B}}^{P\cong Q} \leq \textstyle\sum_{\tau\in T} |\mathbf{Pr}_{\rho_s}\{\mathcal{B}^{\mathcal{O}_P}(1^\eta) = 1 \mid E_\tau\} - \mathbf{Pr}_{\rho_s}\{\mathcal{B}^{\mathcal{O}_Q}(1^\eta) = 1 \mid E_\tau\}|$$

As the advantage is overwhelming, so is the sum. Recall that as P and Q are simple, they are finite, and in particular T is finite. Now, we classically have that a finite sum is overwhelming if and only if one of its element is overwhelming. Thus, there exists a trace τ such that $|\mathbf{Pr}_{\rho_s}\{\mathcal{B}^{\mathcal{O}_P}(1^\eta) = 1 \mid E_\tau\} - \mathbf{Pr}_{\rho_s}\{\mathcal{B}^{\mathcal{O}_Q}(1^\eta) = 1 \mid E_\tau\}|$ is overwhelming.

5.5 Build the model $\mathcal{M}^{\mathcal{A}}$

Let us consider this τ of length n given. We consider two QTM $\mathcal{A}'$ and $\mathcal{B}'$, where $\mathcal{A}'$ is $\mathcal{B}$ running until the last oracle query of $\mathcal{B}$ when $\mathcal{B}$ follows τ, and $\mathcal{B}'$ is the remainder of $\mathcal{B}$. $\mathcal{A}'$ and $\mathcal{B}'$ return a fixed default value when $\mathcal{B}$ does not follow the scheduling τ.

By definition of their construction, ψ_τ^P exactly matches the oracle answers from the execution of $\mathcal{B}^{\mathcal{O}_P}$ when it follows the trace τ.

Thus, in the model $\mathcal{M}^{\mathcal{A}}$ defined by $\mathcal{M}_f$ with attacker machine $\mathcal{A}'$, the oracle answers from the execution of $\mathcal{B}^{\mathcal{O}_P}$ follows the same distribution as the $[\![\psi_P^\tau]\!]^\eta_{\mathcal{M}^{\mathcal{A}},\rho_s}$ when conditioning over E_τ.

Thus, we have that whenever we condition over event E_τ, $\mathcal{B}$ and $\mathcal{B}'$ distinguish with the same probability, i.e.,

$$\begin{aligned}&|\mathbf{Pr}_{\rho_s}\{\mathcal{B}'([\![\psi_P^\tau]\!]^\eta_{\mathcal{M}^{\mathcal{A}},\rho_s}, 1^\eta) = 1 \mid E_\tau\} - \mathbf{Pr}_{\rho_s}\{\mathcal{B}'([\![\psi_Q^\tau]\!]^\eta_{\mathcal{M}^{\mathcal{A}},\rho_s}, 1^\eta) = 1 \mid E_\tau\}| \\ &= |\mathbf{Pr}_{\rho_s}\{\mathcal{B}^{\mathcal{O}_P}(1^\eta) = 1 \mid E_\tau\} - \mathbf{Pr}_{\rho_s}\{\mathcal{B}^{\mathcal{O}_Q}(1^\eta) = 1 \mid E_\tau\}|\end{aligned}$$

Thus, both of these terms are overwhelming, and we have that

$$|\mathbf{Pr}_{\rho_s}\{\mathcal{B}'([\![\psi_P^\tau]\!]^\eta_{\mathcal{M}^{\mathcal{A}},\rho_s}, 1^\eta) = 1\} - \mathbf{Pr}_{\rho_s}\{\mathcal{B}'([\![\psi_Q^\tau]\!]^\eta_{\mathcal{M}^{\mathcal{A}},\rho_s}, 1^\eta) = 1\}|$$

is also overwhelming. In other terms, $\mathcal{B}'$ is a witness that $\mathcal{M}^{\mathcal{A}}$ is a computational model such that $\mathcal{M}^{\mathcal{A}} \not\models \psi_P^\tau \sim \psi_Q^\tau$, which concludes the proof. □

$$\text{=-REFL}\ \frac{}{\mathbb{T}(u \doteq u)} \qquad \text{=-IND}\ \frac{}{\mathbb{T}(u \not\doteq \mathsf{n})}\ \text{when } \mathsf{n} \text{ is not a subterm of } u \qquad \text{=-SYM}\ \frac{\mathbb{T}(u \doteq v)}{\mathbb{T}(v \doteq u)} \qquad \text{=-TRANS}\ \frac{\mathbb{T}(u \doteq v) \quad \mathbb{T}(v \doteq w)}{\mathbb{T}(u \doteq w)}$$

$$\text{=-SUBST}\ \frac{\mathbb{T}(u_1 \doteq v_1) \quad \dots \quad \mathbb{T}(u_n \doteq v_n)}{\mathbb{T}(h(u_1, \dots, u_n) \doteq h(v_1, \dots, v_n))}$$

Figure 9: Truth rules of PQ-BC (identical to those of BC)

Notice that with our interpretation of terms, the soundness is actually simpler to prove than the original soundness proof of [7], which need to split the original attacker into many multiple pieces. Furthermore, the total running time of the attacker machines is the same, in the logic or in the original computational model. This will be a key point that will enable us to instrument the logic with concrete security bounds.

5.6 Logical rules

5.6.1 Probabilistic statements

We introduce some of the logical rules to reason about the $\mathbb{T}()$ predicate, which are all direct transpositions of the rules of the BC logic that are statements about formulas of the form $u \sim \mathsf{true}$. Their soundness proofs are completely similar, as the corresponding BC rules are in fact sound for any Turing Machine, even with unbounded computational power. This means that if the premices are valid, so are the conclusions. We present in Fig. 9 a subset of such rules that allows to reason about the equality between terms, with the other rules being transposed similarly.

Lemma 5. *The truth rules, shown in Fig. 9, are sound in the* PQ-BC *interpretation.*

Proof. =-REFL In any model, $u \doteq u$ evaluates to true with probability 1.

=-SYM As $\doteq$ is a PPTM which interprets a symmetric predicate, the interpretation of the term is equal both in the premice and the conclusion of the rule.

=-TRANS For any model $\mathcal{M}^{\mathcal{A}}$, thanks to the premices, we have that

$$\mathbf{Pr}_{\rho_s}\{[\![u \not\doteq v]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} = 1\} \leq f(\eta)$$

$$\mathbf{Pr}_{\rho_s}\{[\![v \not\doteq w]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} = 1\} \leq g(\eta)$$

where f and g are negligible functions. So, we have that $u \doteq v$ with overwhelming probability $\mathbf{Pr}_{u \doteq v} = (1 - f(\eta))$, and $v \doteq w$ with overwhelming probability $\mathbf{Pr}_{v \doteq w} = (1 - g(\eta))$. The probability that the two events happens at the same time is $\mathbf{Pr}_{u \doteq v \wedge v \doteq w} = \mathbf{Pr}_{u \doteq v} + \mathbf{Pr}_{v \doteq w} - \mathbf{Pr}_{u \doteq v \vee v \doteq w} \geq \mathbf{Pr}_{u \doteq v} + \mathbf{Pr}_{v \doteq w} - 1 \geq 1 - f(\eta) - g(\eta)$

=-SUBST We assume that $\mathbf{Pr}_{u_i \doteq v_i} = (1 - f_i(\eta))$ is overwhelming. Then, we have that:

$$\begin{aligned}
\mathbf{Pr}_{h(u_1,\dots,u_n) \doteq h(v_1,\dots,v_n)} &\geq \mathbf{Pr}_{\bigwedge_{1 \leq i \leq n} u_i \doteq v_i} \\
&\geq \mathbf{Pr}_{u_1 \doteq v_1} + \mathbf{Pr}_{\bigwedge_{2 \leq i \leq n} u_i \doteq v_i} - \mathbf{Pr}_{u_1 \doteq v_1 \vee \bigwedge_{2 \leq i \leq n} u_i \doteq v_i} \\
&\geq \mathbf{Pr}_{u_1 \doteq v_1} + \mathbf{Pr}_{\bigwedge_{2 \leq i \leq n} u_i \doteq v_i} - 1 \\
&\geq \textstyle\sum_{1 \leq i \leq n} \mathbf{Pr}_{u_i \doteq v_i} - (n-1) \\
&\geq \textstyle\sum_{1 \leq i \leq n} (1 - f_i(\eta)) - (n-1) \\
&\geq 1 - \textstyle\sum_{1 \leq i \leq n} f_i(\eta)
\end{aligned}$$

=-INDEP Thanks to the monotonicity property of u, the distribution of u is completely defined by its set of subterms. As n does not occur in the term u, the distribution of u and n are independent,

$$\begin{aligned}
\mathbf{Pr}_{\rho_s}\{[\![u \doteq \mathsf{n}]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} = 1\} &= \textstyle\sum_c \mathbf{Pr}_{\rho_s}\{[\![u]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} = c \wedge [\![\mathsf{n}]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} = c\} \\
&= \textstyle\sum_c \mathbf{Pr}_{\rho_s}\{[\![u]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} = c\} \times \mathbf{Pr}_{\rho_s}\{[\![\mathsf{n}]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} = c\} \\
&= \textstyle\sum_c \mathbf{Pr}_{\rho_s}\{[\![u]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} = c\} \times \frac{1}{2^{\eta}} \\
&= \frac{1}{2^{\eta}}
\end{aligned}$$

□

$$\text{REFL}\ \frac{}{\vec{u} \sim \vec{u}} \qquad \text{TRANS}\ \frac{\vec{u} \sim \vec{v} \quad \vec{v} \sim \vec{w}}{\vec{u} \sim \vec{w}} \qquad \text{PERM}\ \frac{(u_1, \ldots, u_n) \sim (v_1, \ldots, v_n)}{(u_{\pi(1)}, \ldots, u_{\pi(n)}) \sim (v_{\pi(1)}, \ldots, v_{\pi(n)})} \qquad \text{SYM}\ \frac{\vec{v} \sim \vec{u}}{\vec{u} \sim \vec{v}}$$

$$\text{FRESH}\ \frac{\vec{u} \sim_g \vec{v}}{(\vec{u}, \mathsf{n}) \sim_g (\vec{v}, \mathsf{n}')}\ \text{when the new names do not occur in } \vec{u}, \vec{v} \qquad \alpha\text{-EQU}\ \frac{}{\vec{u} \sim \vec{u}\alpha}\ \text{when } \alpha \text{ is an injective renaming of the names} \qquad \text{FA}\ \frac{\vec{u} \sim \vec{v}}{h(\vec{u}) \sim h(\vec{v})}$$

$$\text{RESTR}\ \frac{(\vec{u}, s) \sim (\vec{v}, t)}{\vec{u} \sim \vec{v}}\ \text{when } \mathsf{Max}_{\mathtt{att}}(\vec{u}) = \mathsf{Max}_{\mathtt{att}}(\vec{v}) \qquad \text{DUP}\ \frac{(\vec{u}, t) \sim (\vec{v}, t)}{(\vec{u}, t, t) \sim (\vec{v}, t, t)}$$

$$\text{EQU}\ \frac{\mathbb{T}(s \doteq t)}{(\vec{u}, s) \sim_g (\vec{u}, t)}\ \text{when } \mathsf{Max}_{\mathtt{att}}(\vec{u}, s) = \mathsf{Max}_{\mathtt{att}}(\vec{u}, t) \qquad \text{IFT}\ \frac{\big(\vec{u}, C[\textbf{if } s \doteq t \textbf{ then } C_0[s] \textbf{ else } w]\big) \sim \vec{v}}{\big(\vec{u}, C[\textbf{if } s \doteq t \textbf{ then } C_0[t] \textbf{ else } w]\big) \sim \vec{v}}$$

$$\text{CS}\ \frac{(\vec{w}, b, u) \sim (\vec{w}', b', u') \quad (\vec{w}, b, v) \sim (\vec{w}', b', v')}{(\vec{w}, \textbf{if } b \textbf{ then } u \textbf{ else } v) \sim (\vec{w}', \textbf{if } b' \textbf{ then } u' \textbf{ else } v')}\ \begin{array}{l}\text{when } \mathsf{Max}_{\mathtt{att}}(\vec{w}, b, u) = \mathsf{Max}_{\mathtt{att}}(\vec{w}, b, v) \\ \quad = \mathsf{Max}_{\mathtt{att}}(\vec{w}', b', u') = \mathsf{Max}_{\mathtt{att}}(\vec{w}', b', v')\end{array}$$

Figure 10: Indistinguishability rules of PQ-BC. We mark new side conditions in blue.

5.6.2 Indistinguishability rules

As mentioned earlier, some statements of classical BC are false under the single-attacker interpretation. For example, $u \sim \mathsf{true}$ is false as soon as u contains an attacker symbol, because the final distinguisher also executes the corresponding attacker call, and could therefore differentiate the sides. Yet, such statements are provable inside BC, because it does not assume the final distinguisher sees all attacker calls. Thus, the existing BC rules are not correct in our new interpretation, and we must add a side condition to make PQ-BC sound.

Similar to the rules for the truth predicate $\mathbb{T}()$, the indistinguishability rules in Fig. 10 are also sound for any attacker, without any assumption on their computational power. Thus, the soundness issues only come from the fact that the distinguisher $\mathcal{B}$ now inherits the state of the attacker $\mathcal{A}$. In the end, the main case where an existing BC rule becomes unsound is when it yields in the conclusion a $\sim$ statement where the attacker $\mathcal{A}$ is not called the same number of times on both sides, which corresponds to the *balance* condition.

Definition 15 (Balance)**.** *Given a sequence of terms $\vec{u}$, we denote by $\mathsf{Max}_{\mathtt{att}}(\vec{u})$ the biggest index i such that the function symbol $\mathtt{att}_i$ appears in $\vec{u}$.*
We say that $\vec{u} \sim \vec{v}$ satisfies the balance conditions, or is balanced, if $\mathsf{Max}_{\mathtt{att}}(u) = \mathsf{Max}_{\mathtt{att}}(v)$.

Definition 16. *We denote by $\mathsf{st}_{\mathtt{att}}(\vec{u})$ the subterm of $\vec{u}$ with $\mathtt{att}_{\mathsf{Max}_{\mathtt{att}}(\vec{u})}$ as the top-level function symbol.*

Most BC rules can then be transformed to PQ-BC rules by additionally requiring the *balance* condition over their indistinguishability. In practice, we will thus require that the balance condition holds over all indistinguishabilities appearing inside the proof tree. In the following, however, we only add the side condition where it is needed, in order to pinpoint which rules may break the balance condition. We present in Fig. 10 the new corresponding set of rules, using the color blue to indicate the new side-conditions. Note that the consistency and monotonicity conditions are enforced globally, and hence also hold for all terms appearing in rules. The CS rule requires a stronger condition than the balance condition. Though we provide it for completeness, this rule is not used in Squirrel nor in PQ-Squirrel, and the balance condition is thus sufficient for their rules to get post-quantum soundness.

Lemma 6. *The indistinguishability rules, shown in Fig. 10, are sound in the* PQ-BC *interpretation.*

Intuitively, this means that whenever we construct a proof in the logic, then if there is an attack on the proven formula, there is an attack on the axioms. If we combine this with the soundness of the logic, we get that a proof in the logic implies the existence of a post-quantum sound reduction from an attack on the protocol to an attack on post-quantum sound cryptographic assumptions.

Proof. Most of the proofs are transpositions of proofs from [7,49], where we additionally rely on Lemma 7 to show the additional information given to the distinguisher $\mathcal{B}$ through the state $\phi^{\vec{u}}_{\mathcal{A}}$ of the interactive attacker $\mathcal{A}$ does not allow it to distinguish the terms.

- REFL The distributions are exactly the same on both sides.
- TRANS The result is direct by triangle inequality over the three advantages.
- PERM We easily obtain a simulator by mixing the inputs.
- FRESH As the names are fresh, there distribution is independent of the other terms. And taken in isolation, an attacker as a null advantage to distinguish two uniform distributions.
- SYM The definition of the interpretation of $\sim$ is symmetric.
- α-EQU Names are all pointers to independent uniform distributions, so applying an injective mapping over them does not change the distribution of the terms.
- FA Applying a deterministic function to the attacker input strictly decreases its advantage. (intuitively, the deterministic function can only reduce the quantity of information available to the attacker)
- RESTR We are given a model with attacker $\mathcal{A}$ and distinguisher $\mathcal{B}$ against $\vec{u} \sim \vec{v}$. Similarly to Lemma 7,we set $\mathcal{A}'$ as the attacker that drops all queries bigger than $\mathsf{Max}_{\mathtt{att}}(\vec{u})$ and forwards the others to $\mathcal{A}$, and set is final state as the final one of $\mathcal{A}$. As we have $\mathsf{Max}_{\mathtt{att}}(\vec{u}) \leq \mathsf{Max}_{\mathtt{att}}(\vec{u}, s)$, we have by Lemma 7 that $\phi^{\vec{u}}_{\mathcal{A}} = \phi^{\vec{u},s}_{\mathcal{A}'}$. Further, as $\mathsf{Max}_{\mathtt{att}}(\vec{u}) = \mathsf{Max}_{\mathtt{att}}(\vec{v}) \leq \mathsf{Max}_{\mathtt{att}}(\vec{v}, t)$, we have that $\phi^{\vec{v}}_{\mathcal{A}} = \phi^{\vec{v},t}_{\mathcal{A}'}$.
- EQU As it is slightly longer, we push this proof in Lemma 8.
- IFT In the positive branch of the conditional, we are replacing two terms that have exactly the same distribution with overwhelming probability. Note that this crucially rely on the locality property.
- CS We assume the premices. With $\vec{s} = \vec{w}, \mathbf{if}\ b\ \mathbf{then}\ u\ \mathbf{else}\ v$ and $\vec{t} = \vec{w'}, \mathbf{if}\ b'\ \mathbf{then}\ u'\ \mathbf{else}\ v'$, let $\mathcal{A}$ and $\mathcal{B}$ be attackers against $\vec{s} \sim \vec{t}$. We let $p_{l,t}, p_{l,f}, p_{r,t}, p_{r,f}$ be the following quantities:

$$\begin{aligned}
p_{l,t} &= \mathbf{Pr}_{\rho_s}\{(y,\vec{x}) = [\![b,\vec{s}]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} \wedge y = 1 \wedge \mathcal{B}(x, 1^{\eta}, \phi^{\vec{s}}_{\mathcal{M}^{\mathcal{A}},\rho_s,\eta}) = 1\} \\
p_{l,f} &= \mathbf{Pr}_{\rho_s}\{(y,\vec{x}) = [\![b,\vec{s}]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} \wedge y = 0 \wedge \mathcal{B}(x, 1^{\eta}, \phi^{\vec{s}}_{\mathcal{M}^{\mathcal{A}},\rho_s,\eta}) = 1\} \\
p_{r,t} &= \mathbf{Pr}_{\rho_s}\{(y,\vec{x}) = [\![b,\vec{s}]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} \wedge y = 1 \wedge \mathcal{B}(x, 1^{\eta}, \phi^{\vec{t}}_{\mathcal{M}^{\mathcal{A}},\rho_s,\eta}) = 1\} \\
p_{r,f} &= \mathbf{Pr}_{\rho_s}\{(y,\vec{x}) = [\![b,\vec{s}]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} \wedge y = 0 \wedge \mathcal{B}(x, 1^{\eta}, \phi^{\vec{t}}_{\mathcal{M}^{\mathcal{A}},\rho_s,\eta}) = 1\}
\end{aligned}$$

We define the attacker $\mathcal{B}_t$ against $\vec{w}, b, u \sim \vec{w}, b', u'$:

$$\mathcal{B}_t(1^{\eta}, \vec{x}, y, z, \phi) = \begin{cases} \mathcal{B}(1^{\eta}, \vec{x}, z, \phi) \text{ if } y = 1 \\ 1 \text{ otherwise} \end{cases}$$

Now, thanks to $\mathsf{Max}_{\mathtt{att}}(\vec{w}, b, u) = \mathsf{Max}_{\mathtt{att}}(\vec{w}, b, v)$, we actually have that $\mathsf{st}_{\mathtt{att}}(\vec{w}, \mathbf{if}\ b\ \mathbf{then}\ u\ \mathbf{else}\ v) = \mathsf{st}_{\mathtt{att}}(\vec{w}, b, u)$, from which we conclude thanks to Lemma 7 that $\phi^{\vec{w},b,u}_{\mathcal{A},\mathcal{M}^{\mathcal{A}},\rho_s,\eta} = \phi^{\vec{w},\mathbf{if}\ b\ \mathbf{then}\ u\ \mathbf{else}\ v}_{\mathcal{A},\mathcal{M}^{\mathcal{A}},\rho_s,\eta}$. With this, and based on the definition of $\mathcal{B}_t$, we can conclude that:

$$\mathbf{Pr}_{\rho_s}\{\mathcal{B}_t(1^{\eta}, [\![\vec{w}, b, u]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s}, \phi^{\vec{w},b,u}_{\mathcal{M}^{\mathcal{A}},\rho_s,\eta}) = 1\} = p_{l,f} + \mathbf{Pr}_{\rho_s}\{[\![b]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} = 1\} \tag{4}$$

$$\mathbf{Pr}_{\rho_s}\{\mathcal{B}_t(1^{\eta}, [\![\vec{w'}, b', u']\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s}, \phi^{\vec{w'},b',u'}_{\mathcal{M}^{\mathcal{A}},\rho_s,\eta}) = 1\} = p_{r,f} + \mathbf{Pr}_{\rho_s}\{[\![b']\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} = 1\} \tag{5}$$

Now, as we assumed $\vec{w}, b, u \sim \vec{w'}, b', u'$, and as b and b' are booleans, we actually must have that $\mathbb{T}(()b \doteq b')$ is valid, and in other terms, $|\mathbf{Pr}_{\rho_s}\{[\![b]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} = 1\} - \mathbf{Pr}_{\rho_s}\{[\![b]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} = 1\}|$ is negligible. From this, and Eqs. (4) and (5), we conclude that the advantage of $\mathcal{B}_t$ in $\mathcal{M}^{\mathcal{A}}$ against $\vec{w}, b, u \sim \vec{w'}, b', u'$ is equal to $|p_{l,f} - p_{r,f}|$ up to a negligible quantity. Let us define $\mathcal{B}_e$ against $\vec{w}, b, v \sim \vec{w}, b', v'$:

$$\mathcal{B}_e(1^{\eta}, \vec{x}, y, z, \phi) = \begin{cases} \mathcal{B}(1^{\eta}, \vec{x}, z, \phi) \text{ if } y = 0 \\ 1 \text{ otherwise} \end{cases}$$

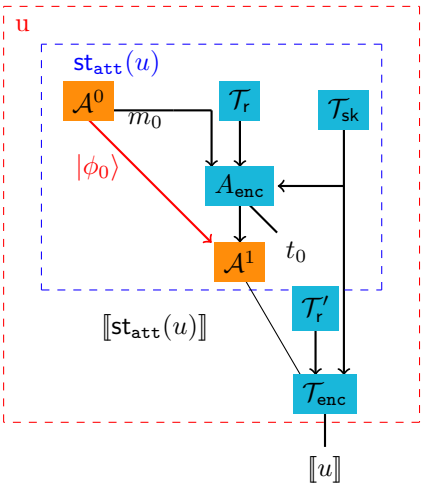

Figure 11:

By symmetry to the previous case, we can obtain that the advantage of $\mathcal{B}_e$ in $\mathcal{M}^{\mathcal{A}}$ against $\vec{w}, b, v \sim \vec{w'}, b', v'$ is $|p_{l,t} - p_{r,t}|$ up to a negligible quantity.

To conclude, we observe that the advantage of $\mathcal{B}$ in $\mathcal{M}^{\mathcal{A}}$ against $\vec{s} \sim \vec{t}$ is $|p_{l,t} + p_{l,f} - p_{r,t} - p_{r,f}|$, which is upper-bounded by triangle inequality by $|p_{l,t} - p_{r,t}| + |p_{l,f} - p_{r,f}|$. AS the advantage of $\mathcal{B}$ is non-negligible, it follows that at either the advantage of $\mathcal{B}_t$ or $\mathcal{B}_e$ must also be, and thus one of the premice is false.

□

To show that the balance condition is not enough to restore the post-quantum soundness of the rule CS, we consider the following example.

Example 8.

$$\textbf{if } \mathit{true} \textbf{ then } \mathsf{n} \textbf{ else } \mathtt{att}_1(\mathsf{n}) \sim \textbf{if } \mathit{true} \textbf{ then } \mathsf{n} \textbf{ else } \mathtt{att}_1(\mathsf{n}')$$

This equivalence does not hold in PQ-BC*, as* $\mathtt{att}_1$ *is always executed independently of the condition, and thus the final distinguisher always gets the value of* n*. However, if* CS *was sound just with the balance condition, we could prove the previous statement by proving:*

$$\mathit{true}, \mathsf{n} \sim \mathit{true}, \mathsf{n} \wedge \mathit{true}, \mathtt{att}_1(\mathsf{n}) \sim \mathit{true}, \mathtt{att}_1(\mathsf{n}')$$

And both of those statements could trivially be proved true in the post-quantum setting.

We recall that we can still only consider the balance condition for PQ-Squirrel, as Squirrel does not rely on the CS rule, it only relies on its equivalent regarding probabilistic statements. Note that if one day the CS was needed in Squirrel, we could make it sound with the balance condition by adapting the PQ-BC interpretation, so that only the attacker symbols appearing in the executed branch of the conditional are executed. This would however make the interpretation depend explicitly on a fixed function symbol, and it would not be generic anymore.

Lemma 7. *Let* $\vec{u}$ *be a sequence of term. For any attacker* $\mathcal{A}$*, we denote by* $\phi^{\vec{v}}_{\mathcal{A}}$ *the (quantum) configuration reached by* $\mathcal{A}$ *during the computation of* $[\![\vec{v}]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s}$ *(which we do not denote by* $\phi^{\vec{v}}_{\mathcal{A},\mathcal{M}^{\mathcal{A}},\rho_s,\eta}$ *for concision). This corresponds to the quantum state forwarded to the final distinguisher* $\mathcal{B}$*.*

Let $\mathcal{A}$ *be an attacker. We first have that*

$$\phi^{\vec{u}}_{\mathcal{A}} = \phi^{\mathsf{st}_{att}(\vec{u})}_{\mathcal{A}} \quad (1)$$

We have for any $\vec{v}$ *such that* $\vec{u}, \vec{v}$ *is a well-formed sequence that*

$$\mathsf{Max}_{att}(\vec{v}) = \mathsf{Max}_{att}(\vec{u}) \Rightarrow \phi^{\vec{u}}_{\mathcal{A}} = \phi^{\vec{v}}_{\mathcal{A}} \quad (2)$$

And if we set $\mathcal{A}'$ *as the attacker that drops all queries bigger than* $\mathsf{Max}_{att}(\vec{u})$ *and forwards the others to* $\mathcal{A}$*, and set is final state as the final one of* $\mathcal{A}$*, we have:*

$$\mathsf{Max}_{att}(\vec{u}) \leq \mathsf{Max}_{att}(\vec{v}) \Rightarrow \phi^{\vec{u}}_{\mathcal{A}} = \phi^{\vec{v}}_{\mathcal{A}'} \quad (3)$$

Proof. 1. In the example of Fig. 11, we can see that as soon as we finished interpreting $\mathsf{st}_{\mathtt{att}}(\vec{u})$, the rest of the computation becomes completely independent of the attacker. Thus, the set of the attacker after $[\![\vec{u}]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s}$ is exactly the same as $[\![\mathsf{st}_{\mathtt{att}}(\vec{u})]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s}$. More formally, in the interpretation from Definition 4, the attacker is only called during steps 1 and 2, and its final state only depends on those steps. And the last iteration of step 2 precisely consist of the evaluation of $[\![\mathsf{st}_{\mathtt{att}}(\vec{u})]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s}$, as it is the biggest attacker function symbol occurrence. Hence, $\phi^{\vec{u}}_{\mathcal{A}}$ is equal to the state of the attacker reached in $[\![\mathsf{st}_{\mathtt{att}}(\vec{u})]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s}$, which is $\phi^{\mathsf{st}_{\mathtt{att}}(\vec{u})}_{\mathcal{A}}$.

2. Suppose that $\mathsf{Max}_{\mathtt{att}}(\vec{v}) = \mathsf{Max}_{\mathtt{att}}(\vec{u})$. It implies that $\mathsf{st}_{\mathtt{att}}(\vec{u}) = \mathsf{st}_{\mathtt{att}}(\vec{v})$, else, $\vec{u}, \vec{v}$ would not satisfy the consistency property. We conclude with (1).

3. By construction of $\mathcal{A}'$.

□

This Lemma directly allows us to link oblivious and non-oblivious indistinguishability.

Lemma 8. *For any terms s, t and any sequence of terms $\vec{u}$ such that $\mathsf{Max}_{att}(\vec{u}, s) = \mathsf{Max}_{att}(\vec{u}, t)$, we have that:*

$$\mathbb{T}(s \doteq t) \Rightarrow \vec{u}, s \sim \vec{u}, t$$

Proof. The left-hand side of the implication directly implies that $\mathbf{Pr}_{\rho_s,\mathcal{A}}\{[\![s \doteq t]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} = 0\}$ is negligible. We thus have that $\sum_c |\mathbf{Pr}_{\rho_s,\mathcal{A}}\{[\![s]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} = c\} - \mathbf{Pr}_{\rho_s,\mathcal{A}}\{[\![t]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} = c\}|$ is negligible. Now, by Lemma 2, we have that $[\![s, \mathbf{0}]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} = [\![s, \mathbf{0}(\vec{u})]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s}$, and we also have that $[\![t, \mathbf{0}]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} = [\![t, \mathbf{0}(\vec{u})]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s}$. This notably implies that $\mathbf{Pr}_{\rho_s,\mathcal{A}}\{[\![s]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} = c\} = \mathbf{Pr}_{\rho_s,\mathcal{A}}\{[\![\vec{u}, s]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} = _c\}$ and similarly for $\vec{u}, t$.

We thus have that $\sum_c |\mathbf{Pr}_{\rho_s,\mathcal{A}}\{[\![\vec{u}, s]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} = c\} - \mathbf{Pr}_{\rho_s,\mathcal{A}}\{[\![\vec{u}, t]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s} = c\}|$ is negligible.

As the statistical distance between $[\![\vec{u}, s]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s}$ and $[\![\vec{u}, t]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s}$ is negligible, we can apply the same function to both and still have a negligible distance, we thus now that the following is negligible for any $\mathcal{B}$.

$$\begin{aligned}|&\mathbf{Pr}_{\rho_s}\{\mathcal{B}([\![\vec{u}, s]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s}, 1^{\eta}, \phi^{\vec{u},t}_{\mathcal{A}}) = 1\} \\ &\quad -\mathbf{Pr}_{\rho_s}\{\mathcal{B}([\![\vec{u}, t]\!]^{,\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s}, 1^{\eta}, \phi^{\vec{u},t}_{\mathcal{A}}) = 1\}|\end{aligned}$$

We now look at the final state of the attacker $\mathcal{A}$. As $s \doteq t$, $\vec{u}, s$ and $\vec{u}, t$ are all well-formed, we also have that $\vec{u}, s, \vec{u}, t$ is well-formed. Thus, as $\mathsf{Max}_{\mathtt{att}}(\vec{u}, s) = \mathsf{Max}_{\mathtt{att}}(\vec{u}, t)$, we have by Lemma 7 that $\phi^{\vec{u},t}_{\mathcal{A}} = \phi^{\vec{u},s}_{\mathcal{A}}$. We can thus conclude that the following is negligible

$$\begin{aligned}|&\mathbf{Pr}_{\rho_s}\{\mathcal{B}([\![\vec{u}, s]\!]^{\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s}, 1^{\eta}, \phi^{\vec{u},s}_{\mathcal{A}}) = 1\} \\ &\quad -\mathbf{Pr}_{\rho_s}\{\mathcal{B}([\![\vec{u}, t]\!]^{,\eta}_{\mathcal{M}^{\mathcal{A}},\rho_s}, 1^{\eta}, \phi^{\vec{u},t}_{\mathcal{A}}) = 1\}|\end{aligned}$$

And thus, we do have that $\vec{u}, s \sim \vec{u}, t$.

□

5.7 Cryptographic assumptions in PQ-BC

In Section 2.4 we discussed how BC encodes cryptographic assumptions, such as PRF, IND-CCA, EUF-CMA, ENC-KP, INT-CTXT, OTP, and DDH. The original proofs of soundness of these encodings in [7, 49] were aimed at a classical attacker. We revisited all these proofs, and due to their direct black-box nature, it turns out these proofs also directly apply against a post-quantum attacker: if there exist an instantiating that satisfies the assumption against a quantum attacker, then the corresponding BC axiom is post-quantum sound.

However, knowing that a BC rule is post-quantum sound w.r.t. the cryptographic assumption does not mean that we know an instantiation (i.e., a concrete scheme) that is secure with respect to a quantum attacker. While we know instantiations for most of the above assumptions with respect to a classical attacker, at this moment we do not know of a post-quantum secure instantiation of the DDH assumption. In the future, a candidate for post-quantum DDH could be the CSI-DDH [45] assumption, based on the CSIDH assumption [24]. Their concrete security is however the subject of discussions [13, 16, 20, 54]. We therefore omit this for now from list of allowed cryptographic assumptions for PQ-BC.

This yields the following list of currently usable PQ-BC axioms for post-quantum proofs: PRF, IND-CCA, EUF-CMA, ENC-KP, INT-CTXT, and OTP. Concretely, this means that a proof in

PQ-BC yields guarantees for post-quantum attackers under the assumption that the previous axioms are instantiated in a post-quantum secure way, e.g., by a protocol that uses CRYSTALS-Dilithium [35] to instantiate EUF-CMA, and see e.g. [67] for a post-quantum sound instantiation of a PRF. Out of these assumptions, the most debatable w.r.t. instantiability is likely ENC-KP, as discussed recently in [40].

6 Mechanization in PQ-Squirrel and Case Studies

In this section, we describe PQ-Squirrel, our extension of Squirrel that produces post-quantum sound proofs in the PQ-BC logic. We give an overview of the post-quantum protocol analysis results that we obtained using PQ-Squirrel in Table 1. As we will show later, it turns out that despite the new term interpretation and corresponding new side conditions, several existing Squirrel proofs could be re-interpreted by PQ-Squirrel as post-quantum sound proofs in PQ-BC.

6.1 PQ-Squirrel

6.1.1 Ensuring post-quantum soundness

To make Squirrel post-quantum sound, we must specify which cryptographic axioms are post-quantum sound, and enforce the three syntactic side conditions from Definitions 5, 6 and 15. Furthermore, if there are cryptographic assumptions for which we know instantiations that are secure against classical attackers, but do not know any post-quantum secure instantiations, we no longer rely on them.

Recall that a substantial amount of work in adapting the BC logic to PQ-BC was due to the flexibility in specifying multiple attackers. In contrast, Squirrel specifications do not include attacker terms: Squirrel automatically produces the attacker terms from the input and output commands in the process specification, under the assumption that there is only one attacker. Concretely, the attacker terms are produced by the interpretation of $\mathsf{input}^P@\tau$, which is equal to $\mathtt{att}_\tau(\mathsf{frame}^P@\mathsf{pre}(\tau))$. While this means that Squirrel only supports a subset of the BC logic, it strongly simplifies protocol specification for the user, and prevents users from accidentally modeling a weaker threat model with multiple disjoint attackers. For our post-quantum purposes, this historical choice is very convenient: all terms produced by the meta-logic of Squirrel already satisfy the consistency and monotonicity properties.

We still need to ensure that PQ-Squirrel verifies the new side conditions w.r.t. the $\mathsf{Max}_{\mathtt{att}}()$ on both sides of the indistinguishability formulas from Fig. 10. Given a meta-logic formula $\vec{u} \sim \vec{v}$, we need to check that for the maximal timestamp element τ of the trace, $\mathsf{input}@\tau$ appears on both sides. If this is the case, we say that a formula satisfies the *synchronization* property.

We cannot check the synchronization property directly since Squirrel internally omits the inputs from the frames when they are redundant, and would lead to falsely discarding proof steps. We instead check a generalized property, which is true if either $\mathsf{input}@\tau$ or $\mathsf{frame}@\tau$ or $\mathsf{frame}@\mathsf{pre}(\tau)$ occurs in the frame.

Lemma 9. *Let $\Gamma \Rightarrow \vec{u} \sim \vec{v}$ be a meta-logic formula of the classical* BC *and* Squirrel*, where Γ is a set of ordering constraints over the timestamp variables appearing in $\vec{u}$ and $\vec{v}$. Let $\mathtt{Max}_\Gamma$ be the set of maximal elements implied by the partial ordering. If for all τ in $\mathtt{Max}_\Gamma$, we have that there exists an element e in $\{\mathsf{input}@\tau\, \mathsf{frame}@\tau, \mathsf{frame}@\mathsf{pre}(\tau)\}$ such that $e \in \vec{u}$ and $e \in \vec{v}$, then, we have that:*

$$\Gamma \Rightarrow (\vec{w}, \vec{u} \sim \vec{w}, \vec{v} \Leftrightarrow \vec{u} \sim \vec{v})$$

where $\vec{w} := \mathsf{input}@\tau_1, \ldots, \mathsf{input}@\tau_k$ when $\mathtt{Max}_\Gamma := \{\tau_1, \ldots, \tau_k\}$.

Further, every meta-logic rule that applies to $\vec{u} \sim \vec{v}$ can also be applied to $\vec{w}, \vec{u} \sim \vec{w}, \vec{v}$.

Proof. In this proof, we refer to the rules of the Squirrel meta-logic from [5]. By Restr, we can easily show that $(\vec{w}, \vec{u} \sim \vec{w}, \vec{v} \Rightarrow \vec{u} \sim \vec{v})$. For the converse, we reason by case analysis over each τ_i inside $\mathtt{Max}_\Gamma$.

- if $\mathsf{input}@\tau_i$ appears in $\vec{u}$ and $\vec{v}$, we have that $\vec{u} \sim \vec{v} \Rightarrow \vec{u}, \mathsf{input}@\tau_i \sim \vec{v}, \mathsf{input}@\tau_i$ using Dup.
- if $\mathsf{frame}@\mathsf{pre}(\tau)$ does, we have that $\vec{u} \sim \vec{v} \Rightarrow \vec{u}, \mathsf{input}@\tau_i \sim \vec{v}, \mathsf{input}@\tau_i$ using Fa-Dup.
- if $\mathsf{frame}@\tau$ does, we first duplicate it with Dup, then expand the macro to add $\mathsf{frame}@\mathsf{pre}(\tau)$ to frame (using Restr to drop the extra element of the frame, and then we are in the previous case.

Now, given a proof step applied to $\vec{u} \sim \vec{v}$, the same proof step can be applied over $\vec{w}, \vec{u} \sim \vec{w}, \vec{v}$. Indeed, all the meta-logic rules allow for an extra context. And the side condition verified over this context are

Protocol	LoC	Primitives and Assumptions				Security properties
		h	sign	enc	⊕	
New case studies of key exchange protocols						
IkeV1$_{\text{PSK}}$ [23]	850	PRF				Strong Secrecy & Authentication
IkeV2$^{\text{SIGN}}_{\text{PSK}}$ [37, 44]	300	PRF	EUF-CMA			Strong Secrecy & Authentication
KE$_{\text{BCGNP}}$ [21]	355	PRF		IND-CCA	OTP	Strong Secrecy
KE$_{\text{FSXY}}$ [38]	620	PRF		IND-CCA	OTP	Strong Secrecy
SC-AKE [41]	745	PRF	SUF-CMA	IND-CCA	OTP	Strong Secrecy & Authentication
Proving post-quantum soundness of Squirrel *case studies [5]*						
Basic Hash [22]	100	PRF				Authentication & Unlinkability
Hash Lock [43]	130	PRF				Authentication & Unlinkability
LAK (with pairs) [42]	250	PRF				Authentication & Unlinkability
MW [52]	300	PRF			OTP	Authentication & Unlinkability
Feldhofer [36]	270			ENC-KP INT-CTXT		Authentication & Unlinkability
Private Authentication [7]	100			ENC-KP IND-CCA		Anonymity

Table 1: PQ-Squirrel case studies: we constructed new models of key exchange protocols with static key compromise, and revisited previous Squirrel protocol models. PQ-Squirrel proves that these protocols are computationally post-quantum secure when they are implemented with post-quantum secure primitives for each of their assumptions.

performed by reasoning over the set of subterms appearing inside the frame. But $\vec{w}$ does not contain any new subterm.

□

This Lemma instantly gives us that any Squirrel proof where all indistinguishability predicate satisfy the extended synchronization condition can be mapped to a proof where the $\mathsf{Max}_{\mathtt{att}}()$ side condition is verified everywhere.

6.1.2 Implementation

PQ-Squirrel can operate in classic Squirrel mode. Additionally, it offers a post-quantum-mode switch that can be enabled inside proof files. When enabled, PQ-Squirrel operates in post-quantum mode: it only allows tactics and axioms that have been proven post-quantum sound, and checks synchronization for every indistinguishability appearing at any step of a proof.

The source-code of PQ-Squirrel is available at [1]. Thanks to our identification of a minimal set of simple syntactic conditions, the PQ-Squirrel extension could be concisely implemented, and only comprises a few hundred line of codes in addition to Squirrel's code base.

6.2 Case studies

We summarize the case studies we performed using PQ-Squirrel in Table 1. They fall into two categories: new case studies for the Internet Key Exchange (IKE) standards and of key exchange protocols based on Key-Encapsulation Mechanisms (KEMs); and previous Squirrel case studies that we could prove post-quantum sound in PQ-Squirrel.

All model files and the prover are at [1].

Required effort: In total, modeling the protocols took in the order of hours, and constructing their proofs in interaction with PQ-Squirrel took in the order of weeks. PQ-Squirrel verifies each resulting proof file in under 10 seconds on a laptop with a quad-core CPU at 1,8GHz.

As a side contribution, we also developed new generic tactics for the prover that enabled the case studies. We first present case studies in Sections 6.3 and 6.4, and then introduce the new tactics in Section 6.5.

6.3 Key exchange case studies: IKE and KEM-based

6.3.1 Threat model and security properties

We modeled five key exchange protocols, for which we proved, e.g.:

- Authentication - if a party accepts, another accepts (with the same parameters).
- Strong Secrecy - the keys derived by the parties are indistinguishable from fresh random values.

In these initial case studies, we consider the same threat model for all key exchange protocols: an arbitrary number of initiators and responders willing to answer to anybody, including to dishonest/compromised identities with attacker-controlled keys. We did not yet prove properties with respect to dynamic corruptions nor more complex security properties like perfect forward secrecy, and we consider them out of scope for this work. We stress however that in a similar fashion to EasyCrypt and CryptoVerif, PQ-Squirrel does not have any hard-coded threat model, and these case studies can be extended in future work.

6.3.2 IKE case studies

The IKE standards, version 1 [44] and 2 [44], specify suites of key exchanges. They are Diffie-Hellman key exchanges that support multiple authentication modes. RFC 8784 [37] addresses the issue of quantum computers breaking the DDH assumption, and its authors claim that the authentication mode based on a pre-shared key in version 1 ($\text{IKEV1}_{\text{PSK}}$) is post-quantum sound. For the same purpose, they also define a way to extend version 2 so that the final key computation depends on a pre-shared key.

For $\text{IKEV1}_{\text{PSK}}$, we use PQ-Squirrel to prove that a pre-shared key between two entities allows to derive an authenticated secret key indistinguishable from a random.

We also analyze the version 2 protocol with signatures for the authentication and extended with the pre-shared key, which we call $\text{IKEV2}^{\text{SIGN}}_{\text{PSK}}$. The proof of $\text{IKEV2}^{\text{SIGN}}_{\text{PSK}}$ is simpler than $\text{IKEV1}_{\text{PSK}}$, because the signatures simplify the derivation of the authentication property.

6.3.3 KEM based key exchanges

KEMs are currently considered as a possible replacement for DH-like key exchanges. KEMs abstract mechanisms that generate and send fresh key material encrypted to another party, from which both parties derive a fresh shared key. Some generic constructions of KEM based key exchanges have been proposed in [21, 38], and have for instance been expanded into a full alternative to TLS in [57] or as post-quantum sound variants of the Signal X3DH handshake [41]. These key exchanges were specifically designed to not rely on any DH-like operations, and their security instead relies on assumptions on the corresponding KEM constructions, i.e., IND-CCA.

In PQ-Squirrel, we generally model KEM-based key exchanges by modeling some common internals of KEMs: generating fresh key material, sending this encrypted to the other party, and then deriving a key from this material at both parties using a key derivation function.

The *basic KEM-based key exchange pattern* is to perform the KEM operation at both parties with respect to their peer's long-term public keys, and then to xor the two resulting fresh keys (one for each direction). This generic pattern was illustrated in our example from Fig. 4. As the knowledge of both fresh keys is needed to derive the final key, the attacker cannot obtain it unless it knows both long-term private keys. Note that such schemes provide implicit authentication, but not (explicit) authentication: only a trusted party can derive the final key, but there is no guarantee that such a party exists.

In our specification, we use `enc` to talk about an abstract KEM construction, while in practice, it is referred to as the encapsulation mechanism, and the decryption is the decapsulation. Presenting it using an encryption symbol directly allows to model it inside PQ-Squirrel, but this does not affect the validity of the proofs.

For all our KEM based case studies, our models include an unbounded number of initiators with distinct secret decapsulation keys sk_I, each willing to initiate an unbounded number of sessions with any honest responder with encapsulation key pk_R, as well as an unbounded number of honest responders willing to engage with an unbounded number of sessions with any arbitrary public key that may be attacker-controlled.

Recall that a generic KEM based construction does not provide explicit authentication properties; thus, for KEM based key exchange we only prove the strong secrecy of the derived keys. In our basic example of Fig. 4, this would be achieved in three steps, first by using IND-CCA to hide the secret ephemeral keys k_I and k_R from the attacker; second by using the PRF assumption to derive valid keys, i.e., showing that $\texttt{kdf}(e_x)$ is indistinguishable from a fresh random n_x; third by using the OTP assumptions that enforces the one-time pad property, and thus that the final key is indistinguishable from random, as it is always equal to $n_x \oplus t$ for some t and fresh random bitstring n_x. For illustration purposes, the actual PQ-Squirrel proof corresponding to this example can be found along with the other case studies at [1]. Interestingly, the proofs carried out in PQ-Squirrel follow this high-level structure for the multiple KEM based case studies, and were thus straightforward to establish.

6.3.4 Concrete case studies

The $\mathrm{KE}_{\mathrm{BCGNP}}$ protocol closely follows the generic pattern. $\mathrm{KE}_{\mathrm{FSXY}}$ uses an additional ephemeral key used for each session, as well as an additional round of PRF application to the key materials before xor-ing them. We prove the strong secrecy of the derived keys for both protocols.

The SC-AKE protocol, intended as a possible post-quantum replacement of Signal's X3DH, can be seen as a variant of $\mathrm{KE}_{\mathrm{FSXY}}$ extended with a third message send from the Responder to the Initiator, containing a signature to provide a form of deniable authentication. Instead of deriving a single key $k := \mathtt{kdf}(sid, e_i) \oplus \mathtt{kdf}(sid, e_r)$, it derives two keys k_s and k_f using `kdf1` and `kdf2`. The first one is used to xor, and thus hide, the signature of the *sid* sent for authentication, and the second one is the derived key. Because of these constructions and their properties, SC-AKE is our most complex case study. The proof first requires proving the authentication of the responder to the initiator, by relying on the EUF-CMA assumption on the signature. After having shown that the material used by the initiator to derive the secret key is from an honest source, we can show that the secret key is strongly secret by following the previous pattern. Such proofs illustrate a strength of the PQ-Squirrel prover: it allows interactions between a part of the logic dedicated to proving reachability properties (e.g. authentication), and then use those properties inside indistinguishability proofs (e.g. secrecy).

6.4 Proving post-quantum soundness of Squirrel case studies

We used PQ-Squirrel to verify the proofs of the nine previous Squirrel case studies. Out of those, PQ-Squirrel was able to prove that six were post-quantum sound, and three were not, as they relied on the DDH assumption.

Thus, it seems that most existing proofs in Squirrel are already post-quantum sound, even though we know it is possible to prove statements in Squirrel that are not post-quantum sound. This appears to be because the proofs of realistic protocols rely on an induction on the length of the trace. We then reason on frames of protocols and prove that each of their possible last messages does not break the indistinguishability. This pattern seems to avoid violating the balance condition.

6.5 Additional tactics

6.5.1 A Non-Malleability tactic

Squirrel already had a tactic for the IND-CCA axiom, which is the one usually used for KEM. However, the IND-CCA axiom is not only used to provide secrecy in the context of KEM, but also a form of weak authentication. If a party receives the ciphertext of something that corresponds to an honest ephemeral share, then there exists a session of an honest initiator that sent it. To prove such an authentication property, the IND-CCA axiom is ill-suited, because we rely on the non-malleability property of the scheme, which is implied by IND-CCA [12, 53]. We developed a new tactic that allows to say that encrypted honest secret share cannot have been tampered with by the attacker, and must have been sent by some honest party. We provide details in Appendix C Using this tactic, we could directly prove the weak authentication of the schemes.

6.5.2 Global tactics

Inside Squirrel proofs, we often consider statements of the form

$$\mathsf{frame@pre}(\tau), t \ \mathsf{frame@pre}(\tau), t'$$

We then show that t and t' are indistinguishable while leaving the frame abstract. Most Squirrel tactics only allow manipulating the terms appearing inside t and t'. However, we sometimes need to perform actions globally: not only on t and t' but also on the terms that may appear inside $\mathsf{frame@pre}(\tau)$, i.e., all the terms inside the protocol. We implemented four new tactics that enable such global manipulations:

- a tactic to globally substitute a name by a fresh name;
- a tactic that allows to prove the indistinguishability of two protocols by proving their functional equivalence, i.e., that they in fact produce exactly the same distributions with overwhelming probability;
- an IND-CCA2 tactic to replace all occurrences of some cipher by a version with perfectly hidden plaintext; and

- a PRF tactic to replace all occurrences of the hash of a given message by the same random.

In comparison, the original Squirrel tactic for PRF only allows to replace one instance of a hash inside t by a random as long as one can prove this hash was never computed before by the protocol.

Details are provided in Appendix D

7 Current limitations and future work

7.1 Minimality of the syntactic conditions

Each of the three syntactic conditions is needed to forbid unsound operations over quantum attackers, as illustrated by the three examples in Section 3. However, they may not be the minimal possible conditions, and our conditions do reduce expressivity. Notably, our current restrictions rule out zero knowledge proofs, whose security analysis often requires rewinding. However, our current assessment is that any weakening of the conditions would inherently be very domain specific, and thus only useful for a small set of protocols. For instance, while we could have loosened the consistency condition to allow for some particular form of post-quantum sound rewinding [62, 64], all such techniques are dedicated to particular cases. We currently believe it would be very difficult to derive a general post-quantum rewinding technique (see e.g. [3]).

7.2 Scope of the tool

In terms of security properties, we have already used PQ-Squirrel to verify a range of properties like unlinkability, anonymity, strong secrecy, and authentication. In general, it allows expressing properties using arbitrary first-order logic formula, which can mix indistinguishability properties and reachability properties, and can thus be used to express all classical security properties.

For protocols, PQ-Squirrel cannot currently carry out proofs that require rewinding or that are in the ROM, but such restriction do not hinder proving a wide range of protocols from the literature. As discussed previously, it is unclear which kind of rewinding should be integrated currently in Squirrel. Integrating the ROM (and QROM) into PQ-BC and PQ-Squirrel could be interesting future work. However, while (Q)ROM is often needed for the analysis of primitives, it is less often needed for protocols. A class of protocol we cannot typically verify are e-voting protocols, that often rely on the ROM. However, there is a long-standing debate on the ROM model, because it is not realizable in practice. Typically, it is often preferable to use the PRF assumption when a secret seed is derived in the protocol, as we did in our case studies. In other cases, integrating results such as [55] into PQ-BC might offer a solution in the future.

Besides protocols whose proofs require rewinding or the ROM, we are not aware of any inherent limitations of the logic or the prover that would hinder the proofs of other post-quantum protocols.

7.3 Refining the case studies

The goal of our case studies is to show the usability and scalability of the tool, not to provide an exhaustive analysis of each of them. For example, our current key exchange analysis only consider a model with static compromise. However, this is not an inherent limitation of the logic, and the threat model is only part of the protocol modeling. As such, a natural future work is to refine our case studies and try to prove more advanced properties such as PFS or PCS.

7.4 Improving automation

Based on our first set of case studies, PQ-Squirrel shows the potential to tackle complex case studies such as the proposal of a recent post-quantum TLS [58]. Performing such complex case studies would benefit from improving the automation in PQ-Squirrel. We see two main possible routes to achieve this. First, some low level reasoning about message equalities and inequalities is already automated in many cases, but could be improved by leveraging SMT solvers. Second, the unique abstraction level of our logic enables us to reason both at the high-level of the executions traces as well as at the low level of the indistinguishability of two messages. This abstraction opens the door for the application of more advanced constraint solving techniques, similar to the one used in symbolic tools, which can further improve automation.

8 Conclusion

We defined the PQ-BC logic for proving protocol security against quantum attackers, and a corresponding PQ-Squirrel prover to mechanize the reasoning. In the process of extending the BC logic for this purpose, we identified three simple syntactic side conditions that are both necessary and sufficient; these conditions, and the new tactics we developed for PQ-Squirrel, can be useful for the classical setting as well. Our initial case studies show that PQ-Squirrel can be effectively used to prove post-quantum protocol security.

Acknowledgments. We thank Hubert Comon for many interesting discussions.

References

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

[2] José Bacelar Almeida, Manuel Barbosa, Gilles Barthe, Matthew Campagna, Ernie Cohen, Benjamin Gregoire, Vitor Pereira, Bernardo Portela, Pierre-Yves Strub, and Serdar Tasiran. A Machine-Checked Proof of Security for AWS Key Management Service. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 63–78, London United Kingdom, November 2019. ACM.

[3] Andris Ambainis, Ansis Rosmanis, and Dominique Unruh. Quantum attacks on classical proof systems: the hardness of quantum rewinding. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 474–483. IEEE, 2014.

[4] Michael Backes, Jörn Müller-Quade, and Dominique Unruh. On the Necessity of Rewinding in Secure Multiparty Computation. In *Theory of Cryptography*, Lecture Notes in Computer Science, pages 157–173, Berlin, Heidelberg, 2007. Springer.

[5] David Baelde, Stéphanie Delaune, Charlie Jacomme, Adrien Koutsos, and Solène Moreau. An interactive prover for protocol verification in the computational model. In *42nd IEEE Symposium on Security and Privacy (S&P 2021)*. IEEE, 2021.

[6] Gergei Bana, Rohit Chadha, and Ajay Kumar Eeralla. Formal Analysis of Vote Privacy Using Computationally Complete Symbolic Attacker. In *Computer Security - 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part II*, pages 350–372, 2018.

[7] Gergei Bana and Hubert Comon-Lundh. A Computationally Complete Symbolic Attacker for Equivalence Properties. In *Proceedings of the 21st ACM Conference on Computer and Communications Security (CCS'14)*, pages 609–620, Scottsdale, Arizona, USA, November 2014. ACM Press.

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

[9] Gilles Barthe, François Dupressoir, Benjamin Grégoire, César Kunz, Benedikt Schmidt, and Pierre-Yves Strub. EasyCrypt: A Tutorial. In *Foundations of Security Analysis and Design VII: FOSAD 2012/2013 Tutorial Lectures*, Lecture Notes in Computer Science, pages 146–166. Springer International Publishing, Cham, 2014.

[10] David Basin, Ralf Sasse, and Jorge Toro-Pozo. Card brand mixup attack: Bypassing the PIN in non-Visa cards by using them for Visa transactions. In *30th USENIX Security Symposium USENIX Security 21)*, 2021.

[11] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 259–274. Springer, 2000.

[12] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In *Annual International Cryptology Conference*, pages 26–45. Springer, 1998.

[13] Daniel J Bernstein, Tanja Lange, Chloe Martindale, and Lorenz Panny. Quantum circuits for the CSIDH: optimizing quantum evaluation of isogenies. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 409–441. Springer, 2019.

[14] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM Journal on computing*, 26(5):1411–1473, 1997.

[15] Karthikeyan Bhargavan, Bruno Blanchet, and Nadim Kobeissi. Verified models and reference implementations for the TLS 1.3 standard candidate. In *IEEE Symposium on Security and Privacy (SP 2017)*, pages 483–502. IEEE, 2017.

[16] Jean-François Biasse, Annamaria Iezzi, and Michael J. Jacobson. A note on the security of CSIDH. In *Progress in Cryptology – INDOCRYPT 2018*, pages 153–168, Cham, 2018. Springer International Publishing.

[17] B. Blanchet. A Computationally Sound Mechanized Prover for Security Protocols. *IEEE Transactions on Dependable and Secure Computing*, 5(4):193–207, October 2008. Conference Name: IEEE Transactions on Dependable and Secure Computing.

[18] Bruno Blanchet. Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif. *Foundations and Trends® in Privacy and Security*, 1(1-2):1–135, October 2016. Publisher: Now Publishers, Inc.

[19] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 41–69. Springer, 2011.

[20] Xavier Bonnetain and André Schrottenloher. Quantum security analysis of CSIDH. *Advances in Cryptology–EUROCRYPT 2020*, 12106:493, 2020.

[21] Colin Boyd, Yvonne Cliff, Juan M. Gonzalez Nieto, and Kenneth G. Paterson. One-round key exchange in the standard model. *International Journal of Applied Cryptography*, 1(3):181, 2009.

[22] Mayla Brusò, Konstantinos Chatzikokolakis, and Jerry den Hartog. Formal verification of privacy for RFID systems. In *CSF*, pages 75–88. IEEE Computer Society, 2010.

[23] David Carrel and Dan Harkins. The Internet Key Exchange (IKE). RFC 2409, November 1998.

[24] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An efficient post-quantum commutative group action. In *Advances in Cryptology – ASIACRYPT 2018*, pages 395–427, Cham, 2018. Springer International Publishing.

[25] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. A formal security analysis of the Signal messaging protocol. *Journal of Cryptology*, 33(4):1914–1983, 2020.

[26] Hubert Comon, Charlie Jacomme, and Guillaume Scerri. Oracle simulation: a technique for protocol composition with long term shared secrets. In *Proceedings of the 27st ACM Conference on Computer and Communications Security (CCS'20)*, Orlando, USA, November 2020. ACM Press.

[27] Hubert Comon and Adrien Koutsos. Formal Computational Unlinkability Proofs of RFID Protocols. In *Proceedings of the 30th IEEE Computer Security Foundations Symposium (CSF'17)*, pages 100–114, Santa Barbara, California, USA, 2017. IEEE Computer Society Press.

[28] V. Cortier, C. C. Drăgan, F. Dupressoir, B. Schmidt, P. Strub, and B. Warinschi. Machine-Checked Proofs of Privacy for Electronic Voting Protocols. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 993–1008, May 2017. ISSN: 2375-1207.

[29] Cas Cremers, Marko Horvat, Jonathan Hoyland, Sam Scott, and Thyla van der Merwe. A comprehensive symbolic analysis of TLS 1.3. In *ACM SIGSAC Conference on Computer and Communications Security (CCS 2017)*, pages 1773–1788. ACM, 2017.

[30] Cas Cremers, Benjamin Kiesl, and Niklas Medinger. A formal analysis of IEEE 802.11's WPA2: Countering the kracks caused by cracking the counters. In *29th USENIX Security Symposium (USENIX Security 2020)*, pages 1–17. USENIX Association, 2020.

[31] Maria Luisa Dalla Chiara, Roberto Giuntini, and Richard Greechie. *Reasoning in quantum theory: sharp and unsharp quantum logics*, volume 22. Springer Science & Business Media, 2013.

[32] A. Delignat-Lavaud, C. Fournet, M. Kohlweiss, J. Protzenko, A. Rastogi, N. Swamy, S. Zanella-Beguelin, K. Bhargavan, J. Pan, and J. K. Zinzindohoue. Implementing and Proving the TLS 1.3 Record Layer. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 463–482, May 2017. ISSN: 2375-1207.

[33] Antoine Delignat-Lavaud, Cédric Fournet, Bryan Parno, Jonathan Protzenko, Tahina Ramananandro, Jay Bosamiya, Joseph Lallemand, Itsaka Rakotonirina, and Yi Zhou. A Security Model and Fully Verified Implementation for the IETF QUIC Record Layer. page 17.

[34] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[35] Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 238–268, 2018.

[36] Martin Feldhofer, Sandra Dominikus, and Johannes Wolkerstorfer. Strong authentication for RFID systems using the AES algorithm. In *CHES*, volume 3156 of *Lecture Notes in Computer Science*, pages 357–370. Springer, 2004.

[37] Scott Fluhrer, Panos Kampanakis, David McGrew, and Valery Smyslov. Mixing Preshared Keys in the Internet Key Exchange Protocol Version 2 (IKEv2) for Post-quantum Security. RFC 8784, June 2020.

[38] Atsushi Fujioka, Koutarou Suzuki, Keita Xagawa, and Kazuki Yoneyama. Strongly Secure Authenticated Key Exchange from Factoring, Codes, and Lattices. Technical Report 211, 2012.

[39] Tommaso Gagliardoni. Quantum Security of Cryptographic Primitives. *arXiv:1705.02417 [quant-ph]*, May 2017. arXiv: 1705.02417.

[40] Paul Grubbs, Varun Maram, and Kenneth G. Paterson. Anonymous, Robust Post-Quantum Public Key Encryption. 2021. `https://ia.cr/2021/708`.

[41] Keitaro Hashimoto, Shuichi Katsumata, Kris Kwiatkowski, and Thomas Prest. An efficient and generic construction for Signal's handshake (X3DH): Post-quantum, state leakage secure, and deniable. In *Public-Key Cryptography – PKC 2021*, pages 410–440, Cham, 2021. Springer International Publishing.

[42] Lucca Hirschi, David Baelde, and Stéphanie Delaune. A method for unbounded verification of privacy-type properties. *J. Comput. Secur.*, 27(3):277–342, 2019.

[43] Ari Juels and Stephen A. Weis. Defining strong privacy for RFID. *ACM Trans. Inf. Syst. Secur.*, 13(1):7:1–7:23, 2009.

[44] Charlie Kaufman, Paul E. Hoffman, Yoav Nir, Pasi Eronen, and Tero Kivinen. Internet Key Exchange Protocol Version 2 (IKEv2). RFC 7296, October 2014.

[45] Tomoki Kawashima, Katsuyuki Takashima, Yusuke Aikawa, and Tsuyoshi Takagi. An efficient authenticated key exchange from random self-reducibility on CSIDH. In *International Conference on Information Security and Cryptology*, pages 58–84. Springer, 2020.

[46] Nadim Kobeissi, Karthikeyan Bhargavan, and Bruno Blanchet. Automated verification for secure messaging protocols and their implementations: A symbolic and computational approach. In *IEEE European symposium on security and privacy (EuroS&P 2017)*, pages 435–450. IEEE, 2017.

[47] Nadim Kobeissi, Georgio Nicolas, and Karthikeyan Bhargavan. Noise Explorer: Fully automated modeling and verification for arbitrary Noise protocols. In *IEEE European Symposium on Security and Privacy (EuroS&P 2019)*, pages 356–370. IEEE, 2019.

[48] A. Koutsos. The 5G-AKA Authentication Protocol Privacy. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 464–479, June 2019.

[49] Adrien Koutsos. *Preuves symboliques de propriétés d'indistinguabilité calculatoire*. Theses, Université Paris-Saclay, September 2019.

[50] B. Lipp, B. Blanchet, and K. Bhargavan. A Mechanised Cryptographic Proof of the WireGuard Virtual Private Network Protocol. In *2019 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 231–246, June 2019.

[51] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In *Computer Aided Verification*, Lecture Notes in Computer Science, pages 696–701, Berlin, Heidelberg, 2013. Springer.

[52] David Molnar and David A. Wagner. Privacy and security in library RFID: issues, practices, and architectures. In *CCS*, pages 210–219. ACM, 2004.

[53] Waka Nagao, Yoshifumi Manabe, and Tatsuaki Okamoto. On the equivalence of several security notions of key encapsulation mechanism. *IACR Cryptol. ePrint Arch.*, 2006:268, 2006.

[54] Chris Peikert. He gives C-sieves on the CSIDH. *Advances in Cryptology–EUROCRYPT 2020*, 12106:463, 2020.

[55] Phillip Rogaway. Formalizing human ignorance. In *International Conference on Cryptology in Vietnam*, pages 211–228. Springer, 2006.

[56] Guillaume Scerri and Stanley-Oakes Ryan. Analysis of Key Wrapping APIs: Generic Policies, Computational Security. pages 281–295. IEEE Computer Society, June 2016.

[57] Peter Schwabe, Douglas Stebila, and Thom Wiggers. Post-Quantum TLS Without Handshake Signatures. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, pages 1461–1480, New York, NY, USA, October 2020. Association for Computing Machinery.

[58] Peter Schwabe, Douglas Stebila, and Thom Wiggers. Post-quantum TLS without handshake signatures. In *ACM SIGSAC Conference on Computer and Communications Security (CCS 2020)*, pages 1461–1480. ACM, 2020.

[59] Fang Song. A note on quantum security for post-quantum cryptography. In *International Workshop on Post-Quantum Cryptography*, pages 246–265. Springer, 2014.

[60] Nikhil Swamy, Cătălin Hriţcu, Chantal Keller, Aseem Rastogi, Antoine Delignat-Lavaud, Simon Forest, Karthikeyan Bhargavan, Cédric Fournet, Pierre-Yves Strub, Markulf Kohlweiss, Jean-Karim Zinzindohoue, and Santiago Zanella-Béguelin. Dependent types and multi-monadic effects in F*. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '16, pages 256–270, New York, NY, USA, January 2016. Association for Computing Machinery.

[61] Dominique Unruh. Universally Composable Quantum Multi-party Computation. In *Advances in Cryptology – EUROCRYPT 2010*, Lecture Notes in Computer Science, pages 486–505, Berlin, Heidelberg, 2010. Springer.

[62] Dominique Unruh. Quantum proofs of knowledge. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 135–152. Springer, 2012.

[63] Dominique Unruh. Quantum Relational Hoare Logic. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–31, 2019.

[64] John Watrous. Zero-knowledge against quantum attacks. *SIAM Journal on Computing*, 39(1):25–58, 2009.

[65] W. K. Wootters and W. H. Zurek. A single quantum cannot be cloned. *Nature*, 299(5886):802–803, October 1982.

[66] A Chi-Chih Yao. Quantum circuit complexity. In *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*, pages 352–361. IEEE, 1993.

[67] Mark Zhandry. How to construct quantum random functions. *Journal of the ACM (JACM)*, 68(5):1–43, 2021.

A Preliminaries on Quantum computing

We first present generalities about quantum computation with circuits, manipulating computation model for security similar to [39, 59]. As in the security context, attackers are more often described as Turing Machines, we then provide a high level definition of Quantum Turing Machines with access to a classical oracle.

Though we provide those preliminaries for completeness, remark we will not manipulate quantum states in the following. In the context of our logic, we will only perform black-box straight line reductions, where given an quantum attacker $\mathcal{A}$, we construct a classical attacker $\mathcal{B}$ with oracle access to $\mathcal{A}$, and $\mathcal{A}$ is only executed in a straight line fashion. Those reductions directly yield post-quantum security, if the assumptions are post-quantum secure. When we will need to bound explicitly the probability of success of polynomial time quantum machine, we will bound the probability of success of a polynomial space classical Turing Machine. Those two facts ensure that we do provide valid reduction to prove post-quantum security, but allow us to ignore the inner workings of quantum machines.

A.1 Quantum Computing with circuits

In quantum systems, a (pure) state is described by a vector $|\phi\rangle$ in some Hilbert space $\mathcal{H}$. We only wish to consider quantum circuits, and thus systems operating over Hilbert spaces of the form $\mathcal{H} = \mathcal{C}^N$ with $N = \{0,1\}^*$ to model sequences of qbits. We consider a fixed orthonormal basis $\{|x\rangle \mid x \in N\}$, which is the computational basis. Each element of the computational basis is a classical state, and every pure quantum states can be seen as a superposition of elements of the computational basis. Every pure state can thus be written as:

$$|\phi\rangle = \sum_x a_x \times |x\rangle$$

where $\sum_x |a_x|^2 = 1$.

A *measurement* allows obtaining probabilistically classical state from a quantum state, where the measurement of $\sum_x a_x \times |x\rangle$ will yield state $|x\rangle$ with probability $|a_x|^2$.

Given separate systems $\mathcal{H}_1, \mathcal{H}_2$, we describe the joint systems with the tensor product $\mathcal{H}_1 \otimes \mathcal{H}_2$

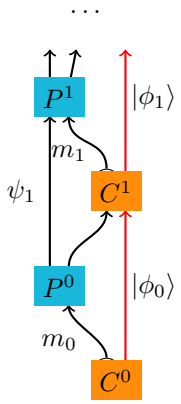

Figure 12:

We now define the machine model, corresponding to a simplified version of the model of [61]. A quantum circuit operates on a state in $\mathcal{H}^{quant} \otimes \mathcal{H}^{class}$ and returns a state in $\mathcal{H}^{quant} \otimes \mathcal{H}^{class}$, where $\mathcal{H}^{quant}$ is a quantum state and $\mathcal{H}^{class}$ is a classical state[1]. Quantum circuit cannot perform all possible operations over quantum states, but details are not relevant to this work.

An interactive machine C is described by a sequence of quantum circuits $\{C_k^i\}$ operating over $\mathcal{H}^{quant} \otimes \mathcal{H}^{class}$, where intuitively $\mathcal{H}^{quant}$ is the state of the machine, $\mathcal{H}^{class}$ is used for incoming and outgoing messages, and C_k^i is the circuit ran at steps i of the execution, when the security parameter is equal to k. An interactive machine is quantum-polynomial-time if there exists a deterministic polynomial time classical Turing Machine that on input 1^k outputs the description of the sequence of circuits C_k^i.

We now define the execution of an interactive quantum machine C with a protocol P. A protocol is simply an interactive quantum machine, and thus a sequence P_k^i of circuits. Fixing the k, we denote by $|\phi_i\rangle \otimes |m_i\rangle$ the output of C^i, and $\psi_i \times l_i$ the output of P^i. $|\phi\rangle$ corresponds to the state passed between the C^i, $|m_i\rangle$ is measured and given as input to P^i and ψ_i is the state passed between the P^i. We describe informally in Fig. 12 the beginning of such an execution, where red wires are quantum, and black wires are classical. The execution of $\{C_k^i\}$ interacting with $\{P_k^i\}$ is given by, at step i, computing the result $|\phi_i\rangle \otimes |m_i\rangle$ of C_k^i over $|\phi_{i-1}\rangle \otimes |l_{i-1}\rangle$, where this is empty when $i = 0$, or $\psi_{i-1} \times l_{i-1}$ is the result of P^{i-1} over $\psi_{i-1} \otimes m_{i-1}$, m_{i-1} being the result of the measurement of $|m_{i-1}\rangle$.

[1] we only separate the two for clarity, but it could be encoded into a single quantum state

A.2 Quantum Computing with Turing Machines

In the following, we consider Post-Quantum Turing Machines with access to oracles, that have an input tape, a working tape, an output tape extra, and oracle query and answer tapes. We refer the reader to [14, Section 3] for the formal definitions of a Quantum Turing Machine.

Intuitively, a QTM can maintain a superposition of configurations, where one of the configuration can itself transition to a superposition of configurations, yielding the linear combination of the superposition. The final output of a QTM is obtained by measuring the quantum state, which corresponds to collapsing the superposition to a single one, where the probability to obtain a given configuration is specified by the superposition.

An oracle QTM can query an oracle by going inside a dedicated state, and writing the oracle input on a dedicated tape. It then gets in a single step the answer of the oracle. Note that in our case, we only have classical oracles, which means that oracle inputs are measured, and outputs are classical. It also means that to perform a query, all configurations in the superposition must be in a query state to perform a measurement and query the oracle.

Formally, we assume as given the Definitions of [14, Section 3] for multi-track Quantum Turing Machines. Such a machine accepts if it goes inside a superposition of configuration such that all configurations

Definition 17. *Given a classical Oracle $\mathcal{O} : \{0,1\}^* \mapsto \{0,1\}^*$, an oracle Quantum Turing Machine is defined as a four tape Turing Machine, an input, a working tape, an output tape and an oracle query tape, with two dedicated pre-query and post-query states. When the Turing-machine is executed, it produces a linear superposition of configuration by following its transition table. Additionally, when all configurations of the superposition are in the pre-query state, in a single step of computation, the oracle query tape is measured, and its content is used as input for $\mathcal{O}$, and then replaced by the oracle answer, and all configurations in the superposition go to the post-query state.*

This model can also be seen as a sequence of QTM without oracle, with quantum input, quantum working tape and classical output. The first classical output is measured when the machine halts, it is given to the oracle, and then we run the second machine with quantum input the concatenation of the classical oracle answer and the content of the working tape of the previous machine.

Note that QTMs are equivalent to circuit based definitions [66] and thus our approach is similar to [39, 59]. See [19] for a discussion on differences between classical and quantum oracles.

Given an oracle $\mathcal{O}$, We denote $A^{\mathcal{O}}$ a QTM with classical access to $\mathcal{O}$. An execution of such a machine can be intuitively seen as a sequence of quantum computations interleaved with some partial measurements of the quantum state for oracle calls.

Polynomial time quantum computing is known to be more powerful than probabilistic attackers, but not more than polynomial space attackers ($\mathsf{BPP} \subset \mathsf{BQP} \subset \mathsf{PSPACE}$ [14, Theorem 8.3&8.4]). It allows us to bound the probability of success of a quantum attacker by bounding the probability of success of a polynomial space attacker.

B A rewinding example in BC

We provide in this section a small example of a proof of a protocol in the BC logic that relies on a rewinding, and describe how it breaks with our additional conditions. We use to this end the small MPC rewinding example of [4].

Definition 18. *The functionality f_{mult} takes an input $a \in \{0,1\}$ from Alice and an input $b \in \{1,2\}$ from Bob and returns $a \times b$.*

If we have a term t_A (resp. t_B) corresponding to the output of Alice (resp. Bob), we easily write in a term the result of the function as $t_A \times t_B$.

We consider the following protocol:

Alice := $\mathbf{in}(c, a)$;
$\mathbf{out}(bob, a)$;
$\mathbf{in}(alice, r)$
if $a = 0$ **then** $\mathbf{out}(c, 0)$
else if $a = 1 \& r \notin \{1, 2\}$ **then** $\mathbf{out}(c, 1)$
else $\mathbf{out}(c, r)$

Bob := $\mathbf{in}(c, b)$;
$\mathbf{in}(alice, a)$;
if $a \notin \{0, 1\}$ **then** $\mathbf{out}(alice, 0); \mathbf{out}(c, 0)$
else $\mathbf{out}(alice, a \times b); \mathbf{out}(c, a \times b)$

Lemma 10. *We can prove in the classical* BC *logic that the previous protocol securely implements f_{mult} with perfect security in the stand-alone model.*

Proof. We consider an attacker replacing Bob, whose first (output for Alice) and final outputs are denoted by $\mathtt{att}_1(b, _)$ and $\mathtt{att}_2(b_)$, where the hole corresponds to the input received by Bob. With those notations, we can write the final value of Alice as $t_A = \textbf{if } a = 0 \textbf{ then } 0 \textbf{ else if } \mathtt{att}_1(b, a) \notin \{1, 2\} \textbf{ then } 1 \textbf{ else } \mathtt{att}_1(b, a)$.

Security when Bob is corrupt is when there exists simulators s^1, s^2 allowed to use $\mathtt{att}_1$ and $\mathtt{att}_2$ such that

$$f(a, s^2) \doteq t_A(a), s^1 \doteq \mathtt{att}_2(b, a) \sim \mathsf{true}, \mathsf{true}$$

We set:

$$s^2 := \textbf{if } \mathtt{att}_1(b, 1) \notin \{1, 2\} \textbf{ then } 1 \textbf{ else } \mathtt{att}_1(b, 1)$$
$$s^1 := \textbf{if } f(a, s^2) = 0 \textbf{ then } \mathtt{att}_2(b, 0) \textbf{ else } \mathtt{att}_2(b, 1)$$

If we evaluate the terms by case disjunction:

a	t_A	$f(a, s^2) = a \times s^2$	s^1
0	0	0	$\mathtt{att}_2(b, 0)$
1	s^2	$s^2 (\in \{1, 2\})$	$\mathtt{att}_2(b, 1)$

This concludes the proof, modulo a tedious casting of the case disjunction inside the BC logic. □

The good news is that the previous security proof breaks in the PQ-BC logic, because s^2 does not satisfy the consistency condition!

C The non malleability version of the IND-CCA$_2$ axiom

We define a new BC axiom that corresponds to the non malleability of a ciphertext. We rely on the axiom for Indistinguishability against Chosen Ciphertexts Attacks from [12] and its multi-users variants [11].

First, the challenger computes a public/private key pair $(\mathtt{pk}(\mathsf{n}), \mathtt{sk}(\mathsf{n}))$ (using a nonce n of length η uniformly sampled), and sends $\mathtt{pk}(\mathsf{n})$ to the attacker. The attacker has access to two oracles:

- The left-right oracle $\mathcal{O}^b_{LR}(\mathsf{n})$, that takes two messages m_0, m_1, and returns $\{m_b\}^{\mathsf{r}}_{\mathtt{pk}(\mathsf{n})}$, where b is a Boolean.
- The decryption oracle $\mathcal{O}_{\mathtt{dec}}(\mathsf{n})$, that given m, returns $\mathtt{dec}(m, \mathtt{sk}(\mathsf{n}))$, if m does not correspond to an output of the left-right oracle, and a sequence of zeros of length m otherwise.

There already exists formalisations of the IND-CCA$_2$ axiom, see e.g. [49]. Yet, they are difficult to use in order to prove authentication, as they force us to render encryption after encryption the attacker view independent of the secret challenge used to establish some authentication, and finally, prove that only the honest execution can pass all the test of the protocol. To prove the strong secrecy of a name, an easily usable rule however, is CCA1.

To take a slightly different approach, we take ideas from the non-malleability axiom, and provide a reachability axiom, that says that the attacker cannot produce a message that has a significant link with a secret that it never saw non encrypted.

We propose the following base logic axiom REACH-CCA$_2$, for all t, C, s such that

- sk only occurs in key position t, C, s;
- n only occurs under encryptions performed by keys that only occur in key position in t, C, s;
- n' is fresh;
- all decryptions occurs under encryptions;
- all encryptions use valid fresh randomness.

$$t(\mathsf{n}) \doteq t(\mathsf{n}') \sim \mathsf{false} \Rightarrow \begin{array}{l} \textbf{if } \dot{\bigwedge}_{\{x\}^{\mathsf{r}}_{\mathtt{pk}(\mathsf{sk})} \in s} s \neq \{x\}^{\mathsf{r}}_{\mathtt{pk}(\mathsf{sk})} \textbf{ then} \\ \quad C[\mathtt{dec}(s, \mathsf{sk})] \doteq t(\mathsf{n}) \\ \textbf{else} \\ \quad \mathsf{false} \end{array} \sim \mathsf{false}$$

Intuitively, the attacker cannot produce the ciphertext of a message that holds a significant relation with n, as n only occurs under honest encryptions. Thus, unless it forwards an honest encryption, it cannot produce the term $t(n)$, which holds a significant relation with n. The fact that $t(\mathsf{n})$ holds a significant relation to n is witness by the fact that $t(\mathsf{n}) \doteq t(\mathsf{n}') \sim \mathsf{false}$.

Proposition 1. *The axiom* ***REACH-CCA$_2$*** *is sound in all models where the encryption is multi-user* ***IND-CCA$_2$****.*

Proof. Let us assume that we have a model that contradicts an instance t, C, s of the axiom, and thus a model where $\big(\dot{\bigwedge}_{\{x\}^{\mathsf{r}}_{\mathsf{pk}(\mathsf{sk})} \in s} \; s \neq \{x\}^{\mathsf{r}}_{\mathsf{pk}(\mathsf{sk})} \dot{\Rightarrow} C[\mathsf{dec}(s, \mathsf{sk})] \doteq t(\mathsf{n}) \big)$ evaluate to true with non-negligible probability $\mathrm{Adv}^{\texttt{REACH-CCA}_2}$, and $t(\mathsf{n}) \doteq t(\mathsf{n}')$ evaluates to true with negligible probability $\mathbf{Pr}_t$. We show how to construct an attacker that wins the $\texttt{IND-CCA}_2$ game with a non-negligible advantage.

The attacker $\mathcal{B}$ first receives $\mathsf{pk}(\mathsf{sk})$. Then it samples himself all the names occurring in the term s, t, C, and samples a duplicate n' of n. The attacker is then going to simulate the construction of two messages in parallel, one intuitively corresponding to the message $C[\mathsf{dec}(s, \mathsf{sk})]$, and the other one to the same message, where all occurrences of n have been replaced by occurrences of n'. To do so, consider the pair $S(t)$ defined recursively by:

- $S(n) = (\mathsf{n}, \mathsf{n}')$
- $S(f(\vec{u})) = (f(\vec{v_l}), f(\vec{v_r}))$ when $(\vec{v_l}, \vec{v_r}) = S(\vec{u})$ (we write as a shortcut $f(m)$, to denotes the fact that the attacker simulates the PTTM corresponding to the function symbol f, either using a machine from the functional model, or an attacker machine from the model negating $\texttt{REACH-CCA}_2$)
- $S(\{u\}^{\mathsf{r}}_{\mathsf{pk}(\mathsf{sk})}) = (m, m)$, where m is the answer of $\mathcal{O}^b_{LR}(\mathsf{sk})$ given as input the pair $S(u)$.
- $S(\mathsf{dec}(u, \mathsf{sk})) = (m_l, m_r)$ where, with $(u_l, u_r) = S(u)$, if $u_l = u_r$ is the result of a query (m_l, m_r) to the left-right oracle, and else m_l m_l (resp. m_r) corresponds to the value returned by the decryption oracle over query u_l (resp. u_r).

Now, one can easily check that the couple $S(s) = (s_l, s_r)$ satisfies the property that, when $b = 0$, s_l is the interpretation of s, and when $b = 1$, s_r is the interpretation of $s[\mathsf{n} \mapsto \mathsf{n}']$. Note that s_r when $b = 0$ corresponds to the interpretation of some term that mixes n and n', and we cannot say a lot about it. Furthermore, for every term in which n only occurs under encryption, $s_l = s_r$, as the two components are made equal whenever we encounter an encryption symbol (the result of the left-right oracle), and are only made dis-equal when encountering n.

Equipped with the definition of S, the attacker $\mathcal{B}$ then performs the following steps:

1. compute $(s_l, s_r) = S(C(\mathsf{dec}(s, \mathsf{sk})))$;
2. returns $t(\mathsf{n}) = s_l$.

We know that n only occurs under encryption symbols. Thus, in the execution of $\mathcal{B}$, we have that $s_l = s_r$. Further, we have that, when $b = 0$, s_l has exactly the same distribution as $C[\mathsf{dec}(s, \mathsf{sk})]$, and thus $\mathcal{B}$ outputs true with non-negligible probability $\mathrm{Adv}^{\texttt{REACH-CCA}_2}$. When $b = 1$, the distribution of s_r is completely independent of n (the left-right oracle always returned the part of the pair depending only on n'). We can conclude if $\mathbf{Pr}(s_l = t(\mathsf{n}))$ is negligible.

By contradiction, let us assume that $\mathbf{Pr}_{\mathsf{n},\rho}(s_l = t(\mathsf{n}))$ is non-negligible (where we denote ρ_r all randomness on which depends s_l), then there exists a constant c such that $\mathbf{Pr}_{\mathsf{n}}(c = t(\mathsf{n}))$ is non-negligible. Indeed, $\mathsf{sup}_c(\mathbf{Pr}_{\mathsf{n}}(c = t(\mathsf{n})))$ is non-negligible, as:

$$\begin{array}{ll} \mathbf{Pr}_{\mathsf{n},\rho}(s_l = t(\mathsf{n})) & = \sum_c \mathbf{Pr}_{\mathsf{n},\rho}(c = t(\mathsf{n}) \wedge c = s_l) \\ & = \sum_c \mathbf{Pr}_{\mathsf{n}}(c = t(\mathsf{n})) \times \mathbf{Pr}_{\mathsf{n},\rho}(c = s_l) \\ & < \sum_c \mathsf{sup}_c(\mathbf{Pr}_{\mathsf{n}}(c = t(\mathsf{n}))) \times \mathbf{Pr}_{\rho}(c = s_l) \\ & \leq \mathsf{sup}_c(\mathbf{Pr}_{\mathsf{n}}(c = t(\mathsf{n}))) \times \sum_c \mathbf{Pr}_{\rho}(c = s_l) \\ & \leq \mathsf{sup}_c(\mathbf{Pr}_{\mathsf{n}}(c = t(\mathsf{n}))) \times 1 \end{array}$$

Finally, $\mathbf{Pr}_{\mathsf{n},\mathsf{n}'}(t(\mathsf{n}') = t(\mathsf{n})) = \mathbf{Pr}_{\mathsf{n},\rho}(s_l = t(\mathsf{n}))^2$ is non-negligible, which contradicts the fact that $t(\mathsf{n}) = t(\mathsf{n}')$ with negligible probability $\mathbf{Pr}_t$. Thus, when $b = 1$, $\mathcal{B}$ outputs true with at most probability $\mathbf{Pr}_t$, and thus outputs true with negligible probability.

We conclude that we have a distinguisher against the $\texttt{IND-CCA}_2$ game, which outputs true with non-negligible probability when $b = 0$, and with negligible probability when $b = 1$. □

$$\text{NM2}\quad \frac{\emptyset \vdash t(\mathsf{n}) \not\doteq t(\mathsf{n}') \quad \Gamma, \big(\dot{\bigvee}_{\{x\}^{\mathsf{r}}_{\mathsf{pk}(\mathsf{sk})} \in s} s \doteq \{x\}^{\mathsf{r}}_{\mathsf{pk}(\mathsf{sk})}\big), C(\mathsf{dec}(s,\mathsf{sk})) \doteq t(\mathsf{n}) \vdash \phi}{\Gamma, C(\mathsf{dec}(s,\mathsf{sk})) \doteq t(\mathsf{n}) \vdash \phi}$$

$$\mathsf{sk} \sqsubseteq^{\mathcal{P}_1,\mathcal{P}_2}_{\mathsf{pk}(\bullet),\mathsf{enc}(_[\mathsf{dec}(_,\bullet)],_,\mathsf{pk}(\bullet))} C,t,s \qquad \mathsf{n} \sqsubseteq^{\mathcal{P}_1,\mathcal{P}_2}_{\mathsf{enc}(\bullet,\bullet,\mathsf{pk}(\bullet))} C,t,s \qquad \mathsf{n}' \not\sqsubseteq^{\mathcal{P}_1,\mathcal{P}_2} C,t,s \qquad \mathtt{senc\text{-}rand}^{\mathcal{P}_1,\mathcal{P}_2}_{\mathsf{sk}}$$

Figure 13: Rule NM2 (meta logic)

Transposing to KEMs The $\mathtt{IND\text{-}CCA}_2$ axioms for KEM is in fact slightly weaker than the one for public-key encryption schemes. Instead of asking that the ciphertexts of any two messages are indistinguishable, it only asks that the ciphertext of a key is indistinguishable from the ciphertext of a uniformly sampled key.

Note that BC axioms, both for $\mathtt{IND\text{-}CCA}_2$ and *CCAreachtwo*, can easily be restricted to model this behaviour, by asking that we only consider ciphertexts over names.

We refer the reader to [53] for multiple notions of non-malleability over KEMs.

C.1 Rules inside Squirrel

We provide in Fig. 13 the reachability rule NM2 for Squirrel.

Proposition 2. NM2 *is sound in any model where the encryption is multi-user* $\mathtt{IND\text{-}CCA}_2$.

Proof. Let us denote by $H = \big(\dot{\bigwedge}_{\{x\}^{\mathsf{r}}_{\mathsf{pk}(\mathsf{sk})} \in s} s \not\doteq \{x\}^{\mathsf{r}}_{\mathsf{pk}(\mathsf{sk})}\big)$. First, we trivially derive from the base logic the soundness of the following rule:

$$\text{NM-aux}\quad \frac{\emptyset \vdash t(\mathsf{n}) \not\doteq t(\mathsf{n}')}{\Gamma, H, C[\mathsf{dec}(s,\mathsf{sk})] \doteq t(\mathsf{n})) \vdash \mathsf{false}}$$

Indeed, assume $t(\mathsf{n}) \doteq t(\mathsf{n}') \sim \mathsf{false}$ (which is equivalent to $t(\mathsf{n}) \not\doteq t(\mathsf{n}') \sim \mathsf{true}$). Then, the base-logic axiom gives us

$$\begin{array}{l}\textbf{if } H \textbf{ then}\\ \quad C[\mathsf{dec}(s,\mathsf{sk})] \doteq t(\mathsf{n})\\ \textbf{else}\\ \quad \mathsf{false}\end{array} \sim \mathsf{false}$$

Now, negating this, we get

$$\begin{array}{l}\textbf{if } H \textbf{ then}\\ \quad C[\mathsf{dec}(s,\mathsf{sk})] \not\doteq t(\mathsf{n})\\ \textbf{else}\\ \quad \mathsf{true}\end{array} \sim \mathsf{true}$$

which translates in therm of implication to
$\big(H \Rightarrow C[\mathsf{dec}(s,\mathsf{sk})] \not\doteq t(\mathsf{n})\big) \sim \mathsf{true}$, and we have:

$$\begin{aligned}\big(H \Rightarrow C[\mathsf{dec}(s,\mathsf{sk})] \not\doteq t(\mathsf{n})\big) \sim \mathsf{true} \quad &\Leftrightarrow \neg H \vee C[\mathsf{dec}(s,\mathsf{sk})] \not\doteq t(\mathsf{n})\big) \sim \mathsf{true}\\ &\Leftrightarrow \neg(\neg H \vee C[\mathsf{dec}(s,\mathsf{sk})] \not\doteq t(\mathsf{n})) \Rightarrow \mathsf{false} \sim \mathsf{true}\\ &\Leftrightarrow H \wedge C[\mathsf{dec}(s,\mathsf{sk})] \doteq t(\mathsf{n}) \Rightarrow \mathsf{false} \sim \mathsf{true}\end{aligned}$$

We then obtain the soundness of the final rule with a small derivation:

$$\text{CUT}\ \frac{\text{NM-aux}\ \dfrac{\emptyset \vdash t(\mathsf{n}) \not\doteq t(\mathsf{n}')}{\neg\text{-R}\ \dfrac{\Gamma, H, C(\mathsf{dec}(s,\mathsf{sk})) = t(\mathsf{n}) \vdash \mathsf{false}}{\Gamma, C(\mathsf{dec}(s,\mathsf{sk})) = t(\mathsf{n}) \vdash \neg H}} \qquad \Gamma, \neg H, C(\mathsf{dec}(s,\mathsf{sk})) = t(\mathsf{n}) \vdash \phi}{\Gamma, C(\mathsf{dec}(s,\mathsf{sk})) = t(\mathsf{n}) \vdash \phi}$$

□

Base logic Rule	Meta-logic Rule
$\dfrac{\Delta \vdash \vec{u}, C\Big[\begin{array}{r}\textbf{if } \mathsf{HFresh}^{\mathsf{k}}(t;\vec{u},C,t) \textbf{ then } \mathsf{n} \\ \textbf{else } \mathsf{H}(t,\mathsf{k})\end{array}\Big] \sim \vec{v}}{\Delta \vdash \vec{u}, C[\mathsf{H}(t,\mathsf{k})] \sim \vec{v}}$	$\dfrac{\Delta \vdash \vec{u}, C\Big[\begin{array}{r}\textbf{if } \mathsf{HFresh}^{\mathsf{k}[\vec{i}]}_{\mathcal{P}_1}(t;\vec{u},C,t) \textbf{ then } \mathsf{n} \\ \textbf{else } \mathsf{H}(t,\mathsf{k}[\vec{i}])\end{array}\Big] \sim \vec{v}}{\Delta \vdash \vec{u}, C[\mathsf{H}(t,\mathsf{k}[\vec{i}])] \sim \vec{v}}$
when $\mathsf{n} \notin \mathsf{st}(\vec{u},C,t),\ \mathsf{k} \sqsubseteq_{\mathsf{H}(_,\bullet)} (\vec{u},C,t)$	when $\mathsf{n} \not\sqsubseteq^{\mathcal{P}_1}_{\bullet}\ \mathsf{st}(\vec{u},C,t),\ \mathsf{k} \sqsubseteq^{\mathcal{P}_1}_{\mathsf{H}(_,\bullet)} (\vec{u},C,t)$
and $\mathsf{HFresh}^{\mathsf{k}}(t;\vec{u}) \overset{def}{=} \bigwedge_{\mathsf{H}(m,\mathsf{k})\in\mathsf{st}(\vec{u})} m \neq t$	and $\mathsf{HFresh}^{\mathsf{k}[\vec{i}]}_{\mathcal{P}}(t;\vec{u}) \overset{def}{=} \bigwedge_{(\mathsf{H}(m,\mathsf{k}[\vec{i_0}]),\vec{j},c)\in\bar{\mathbf{st}}_{\mathcal{P}}(\vec{u})} \forall\vec{j}.\big((\vec{i}=\vec{i_0}\wedge c)\Rightarrow m\neq t\big)$

Figure 14: Rule PRF (base and meta logic)

D Global tactics

Most of the meta-logic rules are local, in the sense that they apply to a term in one position, and not to the full protocol at once.

This can be cumbersome, even on some simple examples. Let us consider for instance the following protocol:

$$!_i(\mathbf{out}(\mathtt{hash}(\mathsf{n}(i),\mathsf{k})|(\mathbf{out}(\mathtt{hash}(\mathsf{n}(i),\mathsf{k})$$

Intuitively, when the hash function satisfy the PRF assumption, this protocol should be indistinguishable from:

$$!_i(\mathbf{out}(\mathsf{r}(i))|(\mathbf{out}(\mathsf{r}(i))$$

To prove it, we need to replace in two different output the same hash by the same random value. However, the existing tactic for PRF in Squirrel intuitively allow to prove the following:

$$\mathsf{frame@pre}(\tau), \mathtt{hash}(\mathsf{n}(i),\mathsf{k})\ \mathsf{frame@pre}(\tau), \mathbf{if}\ \mathit{Cond}\ \mathbf{then}\ \mathsf{r}(i)\ \mathbf{else}\ \mathtt{hash}(\mathsf{n}(i),\mathsf{k})$$

Where *Cond* is a formula dedicated to this instance, that asks that the hashed message was not hashed before in the frame. It essentially allows to replace a hash by a random name only the first time this given hash is computed.

We provide three new tactics, which instead of reasoning only over the explicit terms of the current frame we are investigating, also allow performing protocol wide transformations.

D.1 Extended PRF

We first recall the existing PRF rules for the base and the meta-logic in Fig. 14.

We provide a new base logic rule, that can be derived from the previous one. We let G-PRF be the axiom such that, for all sequences $\vec{u}$ and term t, we have $\vec{u}\sigma \sim \vec{u}$ when

- $\mathsf{n} \notin \mathsf{st}(\vec{u},t)$;
- $\mathsf{k} \sqsubseteq_{\mathsf{H}(_,\bullet)} (\vec{u},t)$
- σ is the substitution such that for all $\mathsf{H}(m,\mathsf{k}) \in \mathsf{st}(\vec{u})$,

$$\mathsf{H}(m,\mathsf{k}) \mapsto \mathbf{if}\ m=t\ \mathbf{then}\ n\ \mathbf{else}\ \mathsf{H}(m,\mathsf{k})$$

Lemma 11. *The axiom* G-PRF *is sound in all models where* H *is a* PRF.

Proof. We assume that $\mathsf{n} \notin \mathsf{st}(\vec{u},t)$ and $\mathsf{k} \sqsubseteq_{\mathsf{H}(_,\bullet)} (\vec{u},t)$, First, note that the original rule directly implies $\vec{u}\sigma_{\mathrm{PRF}} \sim \vec{u}$ where σ_1 maps $\mathsf{H}(t,\mathsf{k})$ to $\mathbf{if}\mathsf{HFresh}^{\mathsf{k}}(t;\vec{u},t)\mathbf{then}\mathsf{n}\mathbf{else}\mathsf{H}(t,\mathsf{k})$ with $\mathsf{HFresh}^{\mathsf{k}}(t;\vec{u}) \overset{def}{=} \bigwedge_{\mathsf{H}(m,\mathsf{k})\in\mathsf{st}(\vec{u})\setminus\{\mathsf{H}(t,\mathsf{k})\}} m \neq t$

Using this variant of the rule, we proceed in two steps. First, we consider σ_1 the substitution such that for all $\mathsf{H}(m,\mathsf{k}) \in \mathsf{st}(\vec{u})$,

$$\mathsf{H}(m,\mathsf{k}) \mapsto \mathbf{if}\ m=t\ \mathbf{then}\ \mathsf{H}(t,\mathsf{k})\ \mathbf{else}\ \mathsf{H}(m,\mathsf{k})$$

$$\text{G-PRF}\quad \frac{\Delta \vdash_{\mathcal{P}_1\sigma,\mathcal{P}_2} \vec{u}\sigma \sim \vec{v}}{\Delta \vdash_{\mathcal{P}_1,\mathcal{P}_2} \vec{u} \sim \vec{v}}$$

$$\begin{array}{l}
\mathsf{n} \not\sqsubseteq^{\mathcal{P}_1}_{\bullet} \mathsf{st}(\vec{u},t),\\
\mathsf{k} \sqsubseteq^{\mathcal{P}_1}_{\mathsf{H}(_,\bullet)} (\vec{u},t)\\
\quad \textbf{tryfind } \vec{i} \textbf{ suchthat } t[\vec{i}] = m \wedge \mathsf{k}[\vec{i}] = \mathsf{k}[\vec{j}] \textbf{ in}\\
\sigma := \{ \quad \mathsf{n}[\vec{i}] \qquad\qquad\qquad\qquad\qquad\qquad |\mathsf{H}(m,\mathsf{k}[\vec{j}]) \in \mathsf{st}(\vec{u})\}\\
\quad \textbf{else } \mathsf{H}(m,\mathsf{k})
\end{array}$$

Figure 15: Rule G-PRF (meta logic)

We are only introducing a trivial if then else, so we do have that $\vec{u}\sigma_1 \sim \vec{u}$. We can then go once step further, and highlight the fact that in the else branch, $m <> t$, with σ_2 the substitution such that for all $\mathsf{H}(m,\mathsf{k}) \in \mathsf{st}(\vec{u})$,

$$\mathsf{H}(m,\mathsf{k}) \mapsto \textbf{if } m = t \textbf{ then } \mathsf{H}(t,\mathsf{k}) \textbf{ else } \mathsf{H}(\textbf{if } m <> t \textbf{ then } m \textbf{ else } 0, \mathsf{k})$$

Once again, the interpretation of the conditional is such that $\vec{u}\sigma_2 \sim \vec{u}\sigma_1$. Let us now apply the PRF axiom over $\vec{u}\sigma_2$ for $\mathsf{H}(t,\mathsf{k})$. It yields that $\vec{u}\sigma_2 \sim \vec{u}\sigma_3$ with σ_3 the substitution such that for all $\mathsf{H}(m,\mathsf{k}) \in \mathsf{st}(\vec{u})$,

$$\mathsf{H}(m,\mathsf{k}) \mapsto \textbf{if } m = t \textbf{ then } \left(\textbf{if } \mathsf{HFresh}^{\mathsf{k}}(t;\vec{u}\sigma_2,t) \textbf{ then } \mathsf{n} \textbf{ else } \mathsf{H}(t,\mathsf{k})\right) \textbf{ else } \mathsf{H}((\textbf{if } m \neq t \textbf{ then } m \textbf{ else } 0), \mathsf{k})$$

Now, we actually have for any $\mathsf{H}(m,\mathsf{k}) \in \mathsf{st}(\vec{u}\sigma_2) \setminus \{\mathsf{H}(t,\mathsf{k})\}$ that $m <> t$, as such m are all of the form $\textbf{if } m \neq t \textbf{ then } m \textbf{ else } 0$.

So in fact, we get that $\mathsf{HFresh}^{\mathsf{k}}(t;\vec{u}\sigma_2,t)$ is equivalent to true. And thus, that $\vec{u}\sigma_3 \sim \vec{u}\sigma_4$ with σ_4 the substitution such that for all $\mathsf{H}(m,\mathsf{k}) \in \mathsf{st}(\vec{u})$,

$$\mathsf{H}(m,\mathsf{k}) \mapsto \textbf{if } m = t \textbf{ then } \mathsf{n} \textbf{ else } \mathsf{H}((\textbf{if } m \neq t \textbf{ then } m \textbf{ else } 0), \mathsf{k})$$

We directly conclude by reverting $\textbf{if } m \neq t \textbf{ then } m \textbf{ else } 0$ to m and by transitivity.

□

Note that we made a proof based on the previous rule, but it could have also been made directly as a reduction from the cryptographic axiom.

D.1.1 Meta logic rule

We now introduce the meta logic rule. We recall that $\Delta \vdash_{\mathcal{P}_1,\mathcal{P}_2} u \sim v$ is used to denote that the macros are expanded on the left-hand side according to the protocol $\mathcal{P}_1$ and on the right according to $\mathcal{P}_2$.

Lemma 12. G-PRF *is sound in all model where* H *satisfies the* PRF *assumption.*

Proof. Recall that to prove the soundness of a meta logic rules, we only have to show that for all possible expansions of the terms, the rule correspond to some valid implication in the base logic.

Our rule is in fact directly a multiple application of the G-PRF rule, one for each copy of t and of k. This is because the **tryfind** construct is expanded as follows, in a simple case where l is an integer and the key and message only depend on a single index:

$$\begin{array}{l}\textbf{tryfind } i \textbf{ suchthat } t[i] = m \wedge \mathsf{k}[i] = \mathsf{k}[j] \textbf{ in}\\ \quad \mathsf{n}[i]\\ \textbf{else } \mathsf{H}(m,\mathsf{k})\end{array} \quad := \quad \begin{array}{l}\textbf{if } t[1] = m \wedge \mathsf{k}[1] = \mathsf{k}[l] \textbf{ then}\\ \quad \mathsf{n}[1]\\ \textbf{else } t[2] = m \wedge \mathsf{k}[2] = \mathsf{k}[l] \textbf{ then}\\ \quad \mathsf{n}[2]\\ \ldots\\ \textbf{else } \mathsf{H}(m,\mathsf{k})\end{array}$$

With this expansion, it becomes clear that the rule is obtained by apply the base logic rule G-PRF first to $\mathsf{H}(t[1],\mathsf{k}[1])$, then $\mathsf{H}(t[2],\mathsf{k}[2])$, ...

□

This rule when used with messages that depend on the attacker may appear counter-intuitive. Let us for instance consider the protocol:

$$!_i\mathbf{in}(x_i); \mathbf{out}(\mathsf{H}(x_i, \mathsf{k}))$$

By applying G-PRF to $\mathsf{H}(x_l, \mathsf{k})$, we get

$$!_i\mathbf{in}(x_i); \mathbf{out}(\mathsf{H}(x_i, \mathsf{k})) \sim !_i\mathbf{in}(x_i); \mathbf{out}(\begin{array}{l}\mathbf{tryfind}\ l\ \mathbf{suchthat}\ x_i = x_l\ \mathbf{in}\\ \quad \mathsf{n}[l]\\ \mathbf{else}\ \mathsf{H}(x_i, \mathsf{k})\end{array})$$

Here, the else branch would never be taken, and if for instance the attacker performs the inputs $x_0 = 0,\ x_1 = 1,\ x_3 = 0$, the frame would be $\mathsf{n}[0], \mathsf{n}[1], \mathsf{n}[0]$.

D.2 Global rewritings

We also introduce two very basic tactics, to help reason globally over some systems. They are both direct consequences of rules of the base logic, that are simply lifted to protocols.

D.2.1 Renaming

For any renaming of a name n into a fresh name n', we have that $\Delta \vdash_{\mathcal{P}_1, \mathcal{P}_1\sigma} \vec{u} \sim \vec{u}\sigma$.

D.2.2 Equal

We may sometimes do the proof of an indistinguishability by instead proving that all the terms inside the protocols are in fact equal with overwhelming probability.

Recall that in the meta-logic, we define a set of action names $\mathcal{A}$, and a concrete action for $a \in \mathcal{A}$ is given by $a(\phi, o)$ where ϕ and o are terms. A protocol $\mathcal{P}_1$ is then a set of concrete actions, and we denote by $\mathcal{A}^{\mathcal{P}_1}$ the set of actions of a protocol.

$$\frac{\text{G-PRF}\quad \emptyset \vdash \bigwedge_{a(\phi_1,o_1)\in\mathcal{P}_1, a(\phi_2,o_2)\in\mathcal{P}_2 \text{ for } a\in\mathcal{A}^{\mathcal{P}_1}} (\mathbf{if}\ \phi_1\ \mathbf{then}\ o_1\ \mathbf{else}\ 0) = (\mathbf{if}\ \phi_2\ \mathbf{then}\ o_2\ \mathbf{else}\ 0)}{\Delta \vdash_{\mathcal{P}_1, \mathcal{P}_2} \vec{u} \sim \vec{u}}$$

Recall that $\Delta \vdash_{\mathcal{P}_1, \mathcal{P}_2} \vec{u} \sim \vec{u}$ is not a trivial goal, as we can for instance have $\vec{u} := \mathsf{frame}@\tau$.

D.3 Extended IND-CCA$_2$

We also provide a generalized version of the IND-CCA$_2$ predicate. We let G-CCA be the axiom such that, for all sequences $\vec{u}$, term t and name r, we have $\vec{u}\sigma \sim \vec{u}$ when

- $\mathsf{k} \sqsubseteq_{\mathtt{pk}(\bullet), \mathtt{dec}(_, \bullet)} (\vec{u}, t)$
- $\mathsf{r} \sqsubseteq_{\mathtt{enc}(t, \mathsf{r}, \mathsf{sk})} (\vec{u}, t)$
- σ is the substitution such that:
  - $\mathtt{enc}(t, \mathsf{r}, \mathsf{sk}) \mapsto \mathtt{enc}(\mathbf{0}(t), \mathsf{r}, \mathsf{sk})$
  - for all $\mathtt{dec}(m, \mathsf{k}) \in \mathsf{st}(\vec{u})$,

$$\mathtt{dec}(m, \mathsf{k}) \mapsto \mathbf{if}\ m = \mathtt{enc}(\mathbf{0}(t), \mathsf{r}, \mathsf{sk})\ \mathbf{then}\ t\ \mathbf{else}\ \mathtt{dec}(m, \mathsf{k})$$

We define the G-CCA experiment in the classical computational game-based setting as follows:

Experiment $G - CCA_{\mathtt{enc}, \mathcal{A}}(\eta)$

$\mathsf{sk} \xleftarrow{\$} \{0,1\}^\eta$
$b \xleftarrow{\$} \{0,1\}$
$m_0, m_1 \xleftarrow{\$} \mathcal{A}^{Dec^{\mathsf{sk}}_{\mathsf{sk}}(\eta, _)}(1^\eta, \mathtt{pk}(\mathsf{sk}))$
$\mathsf{r} \xleftarrow{\$} \{0,1\}^\eta$
$c := \mathtt{enc}(m_b, \mathsf{r}, \mathsf{sk})$
$b' \xleftarrow{\$} \mathcal{A}^{Dec^{c}_{\mathsf{sk}}(\eta, _)}(1^\eta, c)$
return $(b = b')$

Oracle $Dec^c_{\mathsf{sk}}(\eta, m)$

if $c = m$ **then**
  return m_0
else
  return $\mathtt{dec}(m, \mathsf{sk})$

Recall the $\texttt{IND-CCA}_2$ description of Appendix C. We actually prove in the computational model that the security of $\texttt{IND-CCA}_2$ implies the one of G-CCA. The soundness of the G-CCA BC axiom naturally follows as it is a completely direct translation.

Lemma 13. *The advantage of the attacker against the experiment* $G - CCA_{\mathtt{enc},\mathcal{A}}(\eta)$ *, i.e.*

$$|\mathbf{Pr}(G - CCA_{\mathtt{enc},\mathcal{A}}(\eta) = 1)|$$

is negligible in η *if the one against* $\texttt{IND-CCA}_2$ *is also negligible:*

$$|\mathbf{Pr}_{\mathsf{sk}}(\mathcal{A}^{\mathcal{O}^1_{LR}(\mathsf{sk}),\mathcal{O}_{dec}(\mathsf{sk})} - \mathbf{Pr}_{\mathsf{sk}}(\mathcal{A}^{\mathcal{O}^0_{LR}(\mathsf{sk}),\mathcal{O}_{dec}(\mathsf{sk})}|$$

Proof. Let $\mathcal{A}$ be an attacker that wins $G - CCA_{\mathsf{enc},\mathcal{A}}(\eta)$ with overwhelming probability. We build a second attacker $\mathcal{B}^{\mathcal{O}^b_{LR}(\mathsf{sk}),\mathcal{O}_{\mathsf{dec}}(\mathsf{sk})}$ that:

1. Simulate $\mathcal{A}^{\mathcal{O}_{\mathsf{dec}}(\mathsf{sk})}$;
2. submits (m_0, m_1) to $\mathcal{O}^b_{LR}(\mathsf{sk})$ and store the result in c;
3. if in step 1, sk was submitted to $\mathcal{O}_{\mathsf{dec}}(\mathsf{sk})$ (which can be verified by $\mathcal{B}$), use sk to return as final answer $\mathtt{dec}(c, sfsk) = m_0$;
4. otherwise simulates $\mathcal{A}^{Dec^c_{\mathsf{sk}}(\eta,_)}(1^\eta, c)$, such that whenever $\mathcal{A}$ calls $Dec^c_{\mathsf{sk}}(\eta, _)$ on m;
   - check if $m = c$ in which case returns m_0 to $\mathcal{A}$;
   - otherwise, calls $\mathcal{O}_{\mathsf{dec}}(\mathsf{sk})$ on m and returns its answers;
5. return the final answer of $\mathcal{A}$.

In step 1), $\mathcal{O}_{\mathsf{dec}}(\mathsf{sk})$ and $Dec^{\mathsf{sk}}_{\mathsf{sk}}(\eta, _)$ behaves exactly the same, unless sk is queried to the oracle. If such a thing happen, we break $\texttt{IND-CCA}_2$ with probability 1 in step 3, we can thus assume it does not happen and that $\mathcal{A}$ behaves exactly the same in this first step in the experiment or the simulation. Then, the answer of $\mathcal{O}^b_{LR}(\mathsf{sk})(m_0, m_1)$ in step 3 has exactly the same distribution as the value $c := \mathtt{enc}(m_b, \mathsf{r}, \mathsf{sk})$ computed in the original experiment. Finally, in step 4), $\mathcal{B}$ perfectly simulates the behaviour of $Dec^c_{\mathsf{sk}}(\eta, _)$ thanks to $\mathcal{O}_{\mathsf{dec}}(\mathsf{sk})$. As such, $\mathcal{A}$ will decide if c is an encryption of m_0 or m_1 with overwhelming probability.

And finally, we do have that the following is non-negligible:

$$|\mathbf{Pr}_{\mathsf{sk}}(\mathcal{B}^{\mathcal{O}^1_{LR}(\mathsf{sk}),\mathcal{O}_{\mathsf{dec}}(\mathsf{sk})} - \mathbf{Pr}_{\mathsf{sk}}(\mathcal{B}^{\mathcal{O}^0_{LR}(\mathsf{sk}),\mathcal{O}_{\mathsf{dec}}(\mathsf{sk})}|$$

□