Quantum Relational Hoare Logic

Dominique Unruh
University of Tartu

We present a logic for reasoning about pairs of interactive quantum programs – quantum relational Hoare logic (qRHL). This logic follows the spirit of probabilistic relational Hoare logic (Barthe et al. 2009) and allows us to formulate how the outputs of two quantum programs relate given the relationship of their inputs. Probabilistic RHL was used extensively for computer-verified security proofs of classical cryptographic protocols. Since pRHL is not suitable for analyzing quantum cryptography, we present qRHL as a replacement, suitable for the security analysis of post-quantum cryptography and quantum protocols. The design of qRHL poses some challenges unique to the quantum setting, e.g., the definition of equality on quantum registers. Finally, we implemented a tool for verifying proofs in qRHL and developed several example security proofs in it.

The full paper appears as arXiv:1802.03188 [quant-ph].

Introduction. Handwritten security proofs of cryptographic protocols are inherently error prone – mistakes may and most likely will happen, and they can stay unnoticed for years. (The most striking examples are probably the OAEP construction [6] whose proof went through a number of fixes [16, 14, 15] till it was finally formally proven in [2] after years of industrial use, and the PRF/PRP switching lemma which was a standard textbook example for many years before it was shown that the standard proof is flawed [7].) This undermines the purpose of said proofs. In order to enable high trust in our cryptographic protocols, computer-aided verification is the method of choice.

In recent years, a number of logics and tools have been presented for reasoning about cryptographic protocols (without using so-called symbolic models that idealize away from the actual cryptography). For example, the CryptoVerif tool [10, 13] employs a heuristic search using special rewriting rules to simplify a protocol into a trivial one. The CertiCrypt tool [1, 11] and its easier to use EasyCrypt successor [3, 4] allow us to reason about pairs of programs (that represent executions of cryptographic protocols) using the so-called "probabilistic relational Hoare logic" (pRHL). Proofs in the EasyCrypt tool are developed manually, using backwards reasoning with a set of special tactics for reasoning about pRHL judgments.

All those tools only consider classical cryptography. In recent years, interest in quantum cryptography, both in research and in industry, has surged. Quantum cryptography is concerned with the development of protocols that are secure even against attacks using quantum computers, to ensure that we can securely use cryptography in the future (post-quantum cryptography). And quantum cryptography is concerned with the design of quantum protocols that make active use of quantum mechanics in their communication, such as quantum key distribution protocols (an area pioneered by [8]).

Can we formalize quantum cryptographic proofs using the existing tools? For quantum protocols, the answer is clearly no, because the tools do not even allow us to encode the quantum operations performed in the protocol. Yet, even when it comes to post-quantum security where the protocols themselves are purely classical and can be modeled in tools such as EasyCrypt, existing logics fail because quantum adversaries have to be considered. At least for the EasyCrypt tool and the CryptHOL framework [5], we show that they are unsound even for post-quantum cryptography. (By formalizing an example where they fail to be sound, namely the CHSH game [12].)

Submitted to:
QPL 2018

Thus the question arises: can we design tools for the verification of quantum cryptography? (Both for post-quantum security proofs, and for actual quantum protocols.) What underlying logics should we use? In this paper, we set out to answer that question.

Classical RHL. The simplest case of RHL is deterministic RHL [9]. Deterministic RHL allows us to relate two deterministic programs. In our context, a program is always a program in some imperative programming language, operating on a finite set of variables. Consider two programs $\mathbf{c}_1$ and $\mathbf{c}_2$, operating on variables $\mathbf{x}, \mathbf{y}, \dots$ An RHL judgment is an expression $\{A\}\mathbf{c}_1 \sim \mathbf{c}_2\{B\}$ where A and B are Boolean predicates involving the variables from $\mathbf{c}_1$ and $\mathbf{c}_2$ (tagged with index 1 and 2, respectively). For example A might be $\mathbf{x}_1 = \mathbf{x}_2$, and B might be $\mathbf{x}_1 > \mathbf{x}_2$. Then $\{A\}\mathbf{c}_1 \sim \mathbf{c}_2\{B\}$ has the following meaning: If the initial memories m_1, m_2 of $\mathbf{c}_1$ and $\mathbf{c}_2$ satisfy the predicate A, then the memories m_1', m_2' after execution of $\mathbf{c}_1$ and $\mathbf{c}_2$, respectively, satisfy B.

Deterministic RHL can reason only about deterministic programs. At the first glance, it may seem that it is easy to generalize RHL to the probabilistic case (pRHL). However, there are some subtleties in the definition of pRHL. Consider the program $\mathbf{c}$ that assigns a uniformly random bit to $\mathbf{x}$. In this case, is the following pRHL judgment true? $\{\mathtt{true}\}\mathbf{c} \sim \mathbf{c}\{\mathbf{x}_1 = \mathbf{x}_2\}$. That is, if we have two programs that pick a random bit, should pRHL say that after the execution, the bit is the same? At the first glace, one might expect that the answer should be "no" because $\mathbf{x}_1$ and $\mathbf{x}_2$ are chosen independently and thus equal only with probability $\frac{1}{2}$. However, if we define pRHL that way, we cannot express the fact that the two programs $\mathbf{c}$ do the same. We might argue that $\mathbf{x}_1 = \mathbf{x}_2$ should just mean that $\mathbf{x}$ is chosen according to the same distribution in both programs. The formalization of pRHL from [1] takes exactly this approach. They define the pRHL judgment $\{A\}\mathbf{c}_1 \sim \mathbf{c}_2\{B\}$ as follows:

> If the initial memories m_1, m_2 of $\mathbf{c}_1, \mathbf{c}_2$ satisfy the predicate A, then there *exists* a joint distribution of memories m_1', m_2' such that: the pair (m_1', m_2') satisfies B and the marginal distributions m_1' and m_2' are the distributions of the final memories of $\mathbf{c}_1, \mathbf{c}_2$, respectively.

[1] derives a number of useful rules for pRHL. Using these rules, one can derive complex relationships between probabilistic programs. For example, the successful EasyCrypt tool [3, 4] uses these to prove the security of various cryptographic schemes.

pRHL is well-suited for formalizing cryptographic security proofs because those proofs usually proceed by transforming the initial protocol (formulated as a program, called a "game") in a sequence of small steps to a trivial protocol, and the relationship between any two of those games can be analyzed using pRHL.

Quantum RHL. We define a quantum variant of RHL, qRHL. The language of programs we consider is a simple imperative language with classical and quantum variables that can initialize, apply unitaries to, and measure quantum variables. The denotational semantics $[\![\mathbf{c}]\!]$ of a program $\mathbf{c}$ are given as a function mapping quantum states (represented as density operators) to quantum states.

Our central contribution is the notion of qRHL judgments: We lift the definition of pRHL judgments to the quantum case. As it turns out, there are several different possibilities how to do that, in the full paper we describe three variants. As we show, however, two of them lack one or another important reasoning rule, so we are left with the following definition:

Definition 1 (Quantum RHL, informal) $\{A\}\mathbf{c}_1 \sim \mathbf{c}_2\{B\}$ *holds iff: For any separable quantum state ρ that describes a bipartite system (consisting of initial memories for m_1, m_2), and that satisfies the predicate A, there exists a separable quantum state ρ' that describes a bipartite system (consisting of* *final memories) such that: ρ' satisfies the predicate B, and the left and right subsystems of ρ' are the result of applying $\mathbf{c}_1, \mathbf{c}_2$ to the left or right system of ρ, respectively. (Note: that does not mean that ρ' is the result of applying $\mathbf{c}_1 \otimes \mathbf{c}_2$ to ρ; we do not require that entanglement between the two systems is maintained.)*

Of course, we also need to define what a predicate (used for pre- and postconditions) is. In this work, we model a predicate as a subspace of states. Using suitable syntactic sugar, this makes is easy to model both classical and quantum conditions.

When reasoning with qRHL judgments, it turns out that we need one more concept that does not occur in non-relational Hoare logics: The concept of equality between two quantum registers. For example, we often need to model the fact that $\mathbf{q}_1 = \mathbf{q}_2$ where $\mathbf{q}_1$ is a quantum variable containing the state of an adversary in the left program, and $\mathbf{q}_2$ the state of the adversary in the right program. We introduce a definition of quantum equality that is a predicate in our sense (i.e., a subspace of states), and that interacts well with qRHL (as evidenced by the reasoning rules developed for this equality notion).

Our contributions.

- We define qRHL (programming language, semantics, qRHL judgments) and discuss alternatives. We propose a set of useful operations for constructing predicates, in particular a notion of quantum equality.
- We derive a number of derivation rules for qRHL, including analogues of the existing rules in pRHL for classical program constructions, and new rules for quantum operations.
- We develop a tool for computer-aided reasoning in qRHL (currently implementing a subset of our derivation rules) that can be used to experiment with qRHL and to develop verified proofs.
- We present a number of small example analyses in the tool, including the post-quantum security of a simple encryption scheme, and the correctness of quantum teleportation.
- We develop a small example analysis (of the CHSH game) in EasyCrypt and CryptHOL to demonstrate that those tools are not sound for post-quantum cryptography.

References

[1] G. Barthe, B. Grégoire, and S. Zanella Béguelin. “Formal Certification of Code-Based Cryptographic Proofs”. In: *POPL 2009*. ACM, 2009, pp. 90–101.

[2] G. Barthe, B. Grégoire, Y. Lakhnech, and S. Zanella Béguelin. “Beyond Provable Security Verifiable IND-CCA Security of OAEP”. In: *CT-RSA 2011*. Vol. 6558. LNCS. Springer, pp. 180–196.

[3] G. Barthe, B. Grégoire, S. Heraud, and S. Zanella Béguelin. “Computer-Aided Security Proofs for the Working Cryptographer”. In: *Crypto 2011*. Vol. 6841. LNCS. Springer, 2011, pp. 71–90.

[4] G. Barthe, F. Dupressoir, B. Grégoire, C. Kunz, B. Schmidt, and P.-Y. Strub. “EasyCrypt: A Tutorial”. In: *FOSAD 2012/2013 Tutorial Lectures*. Springer, 2014, pp. 146–166.

[5] D. A. Basin, A. Lochbihler, and S. R. Sefidgar. *CryptHOL: Game-based Proofs in Higher-order Logic*. IACR ePrint 2017/753. 2017.

[6] M. Bellare and P. Rogaway. “Optimal Asymmetric Encryption”. In: *Eurocrypt ’94*. Vol. 950. LNCS. Springer, 1994, pp. 92–111.

[7] M. Bellare and P. Rogaway. “The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs”. In: *Eurocrypt 2006*. Vol. 4004. LNCS. Springer, pp. 409–426.

[8] C. H. Bennett and G. Brassard. “Quantum cryptography: Public-key distribution and coin tossing”. In: *Computers, Systems and Signal Processing*. IEEE, 1984, pp. 175–179.

[9] N. Benton. “Simple Relational Correctness Proofs for Static Analyses and Program Transformations”. In: *POPL ’04*. ACM, 2004, pp. 14–25.

[10] B. Blanchet. “A Computationally Sound Mechanized Prover for Security Protocols”. In: *Security and Privacy*. IEEE, 2006, pp. 140–154.

[11] *CertiCrypt: Computer-Aided Cryptographic Proofs in Coq*. `http://certicrypt.gforge.inria.fr/`. Accessed 2017-11-05.

[12] J. F. Clauser, M. A. Horne, A. Shimony, and R. A. Holt. “Proposed Experiment to Test Local Hidden-Variable Theories”. In: *Phys. Rev. Lett.* 23 (15 1969), pp. 880–884.

[13] *CryptoVerif: Cryptographic protocol verifier in the computational model*. `http://prosecco.gforge.inria.fr/personal/bblanche/cryptoverif/`.

[14] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. “RSA-OAEP Is Secure under the RSA Assumption”. In: *Crypto ’01*. Vol. 2139. LNCS. Springer, pp. 260–274.

[15] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. “RSA-OAEP Is Secure under the RSA Assumption”. In: *J Cryptology* 17.2 (2004), pp. 81–104.

[16] V. Shoup. “OAEP Reconsidered”. In: *J Cryptology* 15.4 (2002), pp. 223–249.