Formal Proof of Provable Security by Game-Playing in a Proof Assistant

Reynald Affeldt[1], Miki Tanaka[2], and Nicolas Marti[3]

[1] Research Center for Information Security,
National Institute of Advanced Industrial Science and Technology
reynald.affeldt@aist.go.jp
[2] Information Security Research Center,
National Institute of Information and Communications Technology
[3] Department of Computer Science, University of Tokyo

Abstract. Game-playing is an approach to write security proofs that are easy to verify. In this approach, security definitions and intractable problems are written as programs called games and reductionist security proofs are sequences of game transformations. This bias towards programming languages suggests the implementation of a tool based on compiler techniques (syntactic program transformations) to build security proofs, but it also raises the question of the soundness of such a tool. In this paper, we advocate the formalization of game-playing in a proof assistant as a tool to build security proofs. In a proof assistant, starting from just the formal definition of a probabilistic programming language, all the properties required in game-based security proofs can be proved internally as lemmas whose soundness is ensured by proof theory. Concretely, we show how to formalize the game-playing framework of Bellare and Rogaway in the Coq proof assistant, how to prove formally reusable lemmas such as the fundamental lemma of game-playing, and how to use them to formally prove the PRP/PRF Switching Lemma.

1 Introduction

Game-playing is an approach to write security proofs that are easy to verify. In this approach, security definitions and intractable problems are written as programs called games and reductionist security proofs are sequences of game transformations [6,7,8].

The bias of game-playing towards programming languages suggests the implementation of a tool based on compiler techniques to build security proofs [9], but it also raises the question of the soundness of such a tool. To make our point clearer, let us consider CryptoVerif [13], a pioneer implementation of game-playing that has been applied to several standard cryptographic schemes taken from the literature [4,5]. To perform game transformations, CryptoVerif implements techniques of compiler optimization (constant propagation, dead-code elimination, etc.). The latter program transformations sometimes rely on high-level program equivalences that are only proved on paper and introduced in

W. Susilo, J.K. Liu, and Y. Mu. (Eds.): ProvSec 2007, LNCS 4784, pp. 151–168, 2007.

CryptoVerif as axioms (see Appendix B of [13]). This can be seen as an important limitation of CryptoVerif because it endangers its soundness.

In this paper, we advocate the formalization of game-playing in a proof assistant as a tool to build security proofs. In a proof assistant, starting from just the formal definition of a probabilistic programming language, all the properties required in game-based security proofs can be proved internally as lemmas whose soundness is ensured by proof theory: no game transformation needs to be proved out of the box. Concretely, we show how to formalize the game-playing framework of Bellare and Rogaway [7] in the Coq proof assistant [1], how to prove formally reusable lemmas such as the fundamental lemma of game-playing, and how to use these lemmas to formally prove the PRP/PRF Switching Lemma. To our knowledge, this is the first formalization of game-playing with a random oracle and a working fundamental lemma used in a complete use-case.

About the Coq Proof Assistant. The Coq proof assistant [1] is an implementation of proof theory developed at INRIA in France since 1984. It provides a higher-order logic (i.e., even predicates can be quantified) to state mathematical properties and a functional programming language to build proofs. This setting stems from the Curry-Howard isomorphism [2], through which logical formulas are considered as types of functional programs that are themselves considered as proofs. This makes up for a very small and well-understood proof-checking mechanism that justifies the reliability of proof assistants. Proof assistants are now reasonably mature tools and, in particular, the Coq proof assistant recently made it possible for several important achievements such as the formalization of the four color theorem or the certification of a C compiler.

Notations in this Paper. All the definitions and lemmas in this paper are written in the Coq syntax. This syntax uses only ASCII characters; the mathematical notations are just to ease reading (for example, we write $\forall$ instead of the Coq `forall` construct, $\wedge$ instead of `/\`, etc.). We display Coq code as it appears in our formalization. To improve understanding, we sometimes put comments (between `(*` and `*)`) or hide non-relevant parts (using "..."). In our experience, using the Coq syntax in this way is the best way to present a formalization because it avoids ambiguities while being accessible to readers with little familiarity with formal methods or functional programming languages. There are some Coq-specific constructs, but we introduce them gently in the first sections. We concentrate on the main points of the formalization (basic definitions and statements of lemmas) and do not enter the details of formal proofs; for technical inquiries, the complete Coq development is available online [16].

The rest of this paper is organized as follows. In Sect. 2, we explain how we formalize the notions of distribution and probability in Coq. In Sect. 3, we explain how we formalize random oracles and a probabilistic programming language to write games. In Sect. 4, we formalize a version of the fundamental lemma of game-playing, the most important tool for game-playing. In Sect. 5, we apply our formalization of the game-playing framework to the proof of the PRP/PRF Switching Lemma. We review related work in Sect. 6 and conclude in Sect. 7.