Automated Game-Based Cryptographic Proofs

Gilles Barthe\textsuperscript{1}, Benjamin Grégoire\textsuperscript{2}, Sylvain Heraud\textsuperscript{2}, and Santiago Zanella Béguelin\textsuperscript{1}

\textsuperscript{1} IMDEA Software Institute, Madrid, Spain, 
{\{}Gilles.Barthe,Santiago.Zanella\}\{\textcolor{red}{@imdea.org} \}
\textsuperscript{2} INRIA Sophia Antipolis-Méditerranée, France, 
{\{}Sylvain.Heraud,Benjamin.Gregoire\}\{\textcolor{red}{@inria.fr} \}

The security of cryptographic schemes is generally proved by reduction, showing that the existence of an efficient adversary attacking the security of a scheme would contradict a security assumption. A typical reduction is structured as a sequence of probabilistic experiments, often called games, where the first experiment in the sequence encodes the security of the scheme against an efficient adversary $\mathcal{A}$, and the last experiment gives an explicit construction that uses $\mathcal{A}$ to efficiently solve a problem assumed to be computationally hard, e.g. computing discrete logarithms. Since relating the probability of success of the initial adversary to the success of the final construction can be involved, intermediate games are introduced to decompose the proof in steps of more manageable complexity. As noticed by Bellare and Rogaway [5] and Halevi [6], game-based cryptographic proofs can be rigorously formalized by taking a code-based approach, representing games as probabilistic programs and justifying proof steps using programming-language techniques.

CertiCrypt [3] is a general framework built on top of the Coq proof assistant to certify the security of game-based cryptographic schemes using a code-based approach. The adoption of programming idioms allows giving precise definitions of games, and paves the way for applying programming language methods to justify proof steps rigorously. Specifically, many proof steps involve establishing observational equivalence between two programs, or proving that they satisfy a relational invariant. These statements are established formally using an equational theory for observational equivalence or a full-fledged relational Hoare logic, and certified program transformations. To date, CertiCrypt has been successfully applied to verify prominent cryptographic constructions, including OAEP [2], FDH [7], and Zero-Knowledge protocols [4].

EasyCrypt [1] is an automated tool that builds machine-checked proofs from proof sketches. Proof sketches offer a machine-processable representation of the essence of a security proof, including the sequence of games, relations between the probability of events in those games, and Hoare
logic judgments that justify them. In a nutshell, EasyCrypt implements a verification condition generator that computes for any probabilistic relational Hoare logic judgment a set of verification conditions, expressed in the language of first-order logic, and amenable to automated verification by state-of-the-art tools such as SMT solvers and theorem provers. Moreover, the verification condition generator is proof-producing, and generates Coq files that can be machine-checked using the CertiCrypt framework. Additionally, EasyCrypt implements an automated mechanism for proving claims about probabilities. Overall, EasyCrypt demonstrates that provable security can dramatically benefit from automation using state-of-the-art verification technology, and that verifiable game-based proofs can be constructed with only a moderate effort.

References