Synthesis of Public-Key Encryption Schemes

Gilles Barthe[1], Juan Manuel Crespo[1], Martin Gagné[2], César Kunz[1,3], and Yassine Lakhnech[2]

[1] IMDEA [2] VERIMAG [3]U. Politecnica Madrid

The goal of program synthesis is to generate automatically code that achieves a particular purpose, often specified by some input/output specification. To date, program synthesis has been useed for many application domains, including geometry, graph algorithms, bitvectors algorithms, program inverses, and cryptographic protocols. In this abstract, we report on the first application of program synthesis to public-key encryption schemes. Our approach is based on the following steps:

- smart generation of public-key encryption schemes built from one-way functions, random oracles, and operations on bitstrings;
- efficient symbolic filters for eliminating insecure schemes, and schemes for which the decryption oracle is ill-defined;
- automated proofs of semantic security, using automated Hoare logics for cryptographic constructions [2], and strategies for game-based proofs;
- a new compiler for transforming IND-CPA schemes into IND-CCA schemes.

Our tool generates and proves some well-known schemes, for instance Bellare and Rogaway encryption schemes, and REACT. However, not all generated schemes can be proved secure using automated Hoare logics, or game-based proofs. For instance, our tool generates but cannot prove the security of ZAEP [1], a new redundancy-free public-key encryption scheme based on the Rabin function and RSA with exponent 3. This example suggests that synthesis techniques may lead to surprising discoveries.

References

1. G. Barthe, D. Pointcheval, and S. Zanella-Béguelin. Verified security of redundancy-free encryption from rabin and rsa. Cryptology ePrint Archive, Report 2012/308, 2012.
2. J. Courant, M. Daubignard, C. Ene, P. Lafourcade, and Y. Lakhnech. Towards automated proofs for asymmetric encryption schemes in the random oracle model. In *15th ACM Conference on Computer and Communications Security, CCS 2008*, pages 371–380, New York, 2008. ACM.