techvista
CHENNAI 2008

Program Obfuscation and One-Time Programs

Shafi Goldwasser

Program obfuscation is the process of taking a program as an input and modifying it so that the resulting program has the same I/O behavior as the input program but otherwise looks `completely garbled' to the entity that runs it, even if this entity is adversarial and has full access to the program. Impossibility results, origination with the work of Barak etal in 2001, have been proved that assert that several strong (albeit natural) formulations of obfuscation are impossible to achieve for general programs. That is, there is no generic mechanism that can successfully obfuscate large classes of programs.

Yet, even more recent theoretical results have pointed out a way in which, in spite of these generic impossibility results, the basic concept of program obfuscation is obtainable in certain settings. One setting on which we will elaborate is of one-time programs: programs that can be executed only a restricted and pre-specified number of times. Naturally, these programs cannot be achieved using software alone. We show how to build them using `simple' and `universal' secure memory components.

One-time programs serve many of the same purposes of program obfuscation, the obvious one being software protection. However, the applications of one-time programs go well beyond those of obfuscation, since one-time programs can only be executed once (or more generally, a limited number of times) while obfuscated programs have no such bounds. For example, one-time programs lead naturally to temporary delegation of cryptographic ability, electronic token schemes such as subway tokens, and to ``one-time proofs'': proofs that can only be verified once and then become useless and unconvincing. We show how to use a classical witness and simple secure memory to efficiently construct such ``one-time proofs'' for any NP statement.

Joint work with Yael Kalai and Guy Rothblum.