Advanced Cryptology (GBX9SY06)
Attacking LPN$(100, 1/8)$

2020-01-09

Grading

This assignment is graded as part of the *contrôle continu*. You must send a written report (in a portable format) describing your work and the corresponding source code, *including all tests*, **with compilation and execution instructions** by 2020-01-17T18:00+0100) to:

`pierre.karpman@univ-grenoble-alpes.fr`.

Working in teams of two is allowed and encouraged (but not mandatory), in which case only one report needs to be sent, with the name of both students clearly mentioned.

Objective

The goal of this exercise is to implement Blum, Kalai & Wasserman ("BKW")'s algorithm to solve LPN problems in dimension 100 with noise probability 1/8. For a description of the algorithm, one may refer to the lecture notes and/or to [BL12].

How to proceed

You must first download the tarball `https://www-ljk.imag.fr/membres/Pierre.Karpman/bkw.tar.bz2` which includes a file `bkw_fillme.c` that implements a function `lpn_sampler_100_8` to be used to generate problem instances. This file also declares some suggested data structures to implement BKW, but you are free not to use them.

You should then:

- Determine appropriate parameters for the algorithm, that will allow you to solve the problems "efficiently" (e.g. in less than 15 minutes) with a "high" success probability (e.g. more than 0.5).
- Implement a straightforward version of BKW as a function `void bkw(const uint64_t s[2], uint64_t rs[2])`, where `s` is the "secret" to be found and used in the LPN sampler, and `rs` is to be updated with the solution.
- If time permits, you could also implement an improved version of your solver by e.g. using a fast Walsh-Hadamard transform, or a resampler for an equivalent sparse secret, or both.

Some advices:

— You can use the fact that you actually know the secret to easily test parts of your algorithm and to fine-tune the number of samples needed for a high sucess probability.

— It may be a good idea to first implement the last step of the algorithm (i.e. a straight majority-logic decoding process).

— Recursive programming is cool.

References

[BL12] Daniel J. Bernstein and Tanja Lange, *Never Trust a Bunny*, Radio Frequency Identification. Security and Privacy Issues - 8th International Workshop, RFIDSec 2012, Nijmegen, The Netherlands, July 2-3, 2012, Revised Selected Papers (Jaap-Henk Hoepman and Ingrid Verbauwhede, eds.), Lecture Notes in Computer Science, vol. 7739, Springer, 2012, Available as https://eprint.iacr.org/2012/355.pdf, pp. 137–148.