R(SA)^{2}T

Constantinos Patsakis

June 11, 2013

1 R(SA)^{2}T

R(SA)^{2}T is a project for demonstrating attacks on RSA through SAT solvers. The project is under GNU license and is written in Python. Minor modifications have been made to ToughSAT (`http://toughsat.appspot.com/`) code in order to provide better modeling.

The scope of the project is to illustrate that SAT solvers can be quite efficient when the attacker has partial knowledge of the private key, e.g. cold boot attack scenario.

2 Installation

For Debian based systems you need to install the following packages:

- python-argparse
- python-gmpy

3 Arguments

- -b The number of bits that each prime has. Default is 10.
- -ps The percent of bits of p from which I start to know. Default is 0.
- -pe The percent of bits of p to which I stop to know. Default is 0.
- -qs The percent of bits of q from which I start to know. Default is 0.
- -qe The percent of bits of q to which I stop to know. Default is 0.
- -ds The percent of bits of d from which I start to know. Default is 0.

- -de The percent of bits of d to which I stop to know. Default is 0.
- -pq The percent of random bits of p and q that are known.
- -dpq The percent of random bits of d, p and q that are known.
- -nop Do not use any bit of p.
- -noq Do not use any bit of q.
- -nod Do not use any bit of d.
- -cnf3 Use 3 CNF for the main part.
- -del Delete cnf file
- -t Number of test to execute, good for probabilistic mode. Default is 1.
- -s SAT solver to use. Possible SAT solvers:

0: minisat2 (default)	5: lr_gl_shr	10: cryptominisat
1: picosat	6: precosat	11: manysat2
2: glucose	7: ebminisat	12: splitter
3: glueminisat	8: Spear-32_121	
4: mxc-sat09	9: clasp2	

Obviously the SAT solvers are **NOT** part of the project, they are included **ONLY** for the ease of the user to perform the tests.

Example usage: ./rsat -b 512 -pq 0.53 -s 2

The above will create a composite number of 1024 bits, disclosing 53% of the bits of p and q and solve the system using glucose SAT solver.