Grain of Salt
An Automated Way to Test Stream Ciphers through SAT Solvers
Presentation at Tools for Cryptanalysis

Mate Soos\textsuperscript{1}

UPMC LIP6, PLANETE team INRIA, SALSA team INRIA

23rd June 2010

\textsuperscript{1}The author was supported by the RFID-AP ANR Project, project number ANR-07-SESU-009
Table of Contents

1 Context

2 Translating shift register-based stream ciphers to SAT problems
1 Context
- SAT solvers
- Stream ciphers

2 Translating shift register-based stream ciphers to SAT problems
- Unwinding time
- Describing feedback and filter functions
- Help bits
Motivations and goals

Motivations

- Stream ciphers may be broken using SAT solving tools
- Analysis of stream ciphers is possible using SAT solvers

Goals

- Describe different methods to translate shift register-based stream ciphers to SAT problems
- Demonstrate a tool that can help in this process
What is a SAT solver

Solves a problem in CNF

CNF is an “and of or-s”

\[(x_1 \lor \neg x_3) \land (\neg x_2 \lor x_3) \land (x_1 \lor x_2)\]

Uses DPLL(ϕ) algorithm

1. If formula ϕ is trivial, return SAT/UNSAT
2. ret = DPLL(ϕ with v ← true)
3. if ret == SAT, return SAT
4. ret = DPLL(ϕ with v ← false)
5. if ret == SAT, return SAT
6. return UNSAT
Search tree

Example search tree

Visualised
- Guesses
- Propagations
- Generated learnt clauses
- Clause group causing the propagation

Calculated stats
- Average depth
- Most conflicted clauses
- No. of guess/branch
- Most guessed vars
- Most propagated vars

Mate Soos (UPMC LIP6, PLANETE team INRIA, SALSA team INRIA)
Stream ciphers

Shift register-based stream ciphers
- Use a set of *shift registers*
- Shift registers’ *feedback function* is either linear or non-linear
- Uses a *filter function* to generate 1 secret bit from the state
- Working: clock-filter-clock-filter... feedback-filter-feedback-filter...

Example cipher

![Diagram of a stream cipher with shift registers, feedback, and a filter generating a keystream.]
Outline

1 Context
   - SAT solvers
   - Stream ciphers

2 Translating shift register-based stream ciphers to SAT problems
   - Unwinding time
   - Describing feedback and filter functions
   - Help bits
Translating shift register-based stream ciphers to CNF

**Unwinding time**
- SAT solvers don’t understand the notion of “time”
- We must present the problem as if no timing was involved
- To make unwinding simple, regular clocking is needed

**Describe feedback and filter functions**
- Translate ANF (Algebraic Normal Form) to CNF
  - Either through XORs and Monomials
  - Or through direct translation using Karnaugh maps

**Give help bits to aid solving**
- Select and fix shift register states: guess-and-determine
- Select shift register states randomly, fix randomly, and solve for random problems
Unwinding time

Why unwind?

- CNF cannot handle the notion of “time”
- Must present all shift register states in the CNF at once
- No irregular clockings

Example

- Crypto1 state: 48 bits
- We have observed 56 bits of output
- We need to clock 56 times

Shift register

0 48 103
Unwinding the time

Base-shifting

- If the feedback functions are reversible, we can start in the middle
- Usually best to shift to the middle:

  Shift register

  0   32   79   103

- The total depth of the search is then halved

Grain of Salt

- `./grainofsalt --outputs 56 --crypto crypto1
  --base-shift 32`
- `./grainofsalt --outputs 160 --crypto grain
  --base-shift 80,80`
Describing feedback and filter functions

**ANF→CNF**

- ANF (Algebraic Normal Form): \( a = bc \oplus b \oplus c \)
- Must be converted to CNF: \( x \lor y = \text{true} \)

1. Assign each higher degree monomial an internal variable
2. Describe the XOR as CNF

**Extended monomials**

- CNF does not limit terms to be non-negated as ANF does
- \( bc \oplus b \oplus c = b(c + 1) \oplus c = (b + 1)(c + 1) \oplus 1 = \neg b \neg c \oplus 1 \)
- Default in grainofsalt, can be disabled with --noextmonomials

**Karnaugh maps**

- Uses Karnaugh maps to map functions to CNF
- Does not need internal variables
- May generate substantially smaller function descriptions
Describing feedback and filter functions

Function descriptions in *grainofsalt*

File `grain/functions/sr0/feedback.txt`

sr1-62
sr1-51
sr1-38
sr1-23
sr1-13
sr1-0

Defines feedback function: \[ s_{i+80} = s_{i+62} + s_{i+51} + s_{i+38} + s_{i+23} + s_{i+13} + s_{i} \]

Arbitrary functions

- Arbitrary functions can be defined and used in other functions
- Dependency graph is built from output bits, only used functions are generated in CNF
- Allows to efficiently map ciphers built from functional blocks
Describing feedback and filter functions

Figure: Crypto-1 cipher filter function diagram
Help bits

Why?
- Usually the generated problem is too slow to solve
- Except when breaking Crypto-1 (London travel card) — 40sec. approx
1. Give bits at fixed positions, multiply
2. Give bits at random positions, extrapolate

Given information use
- Recursively propagated at the ANF level
- Monomials are replaced with their constant equivalents
- Monomials are replaced with their monomial equivalents
- To disable: --nopropagate
Help bits

Fixed help bits

- Difficult to know which combination of variables will be fastest
- Automated, randomised (Monte Carlo) greedy algorithm
- Sets all bits, tests difficulty based on generated CNF size
- Average smallest size wins
- To generate: `--genDeterBits N` To use: `--deterBits N`

Probabilistic help bits

- Fix \( n \) randomly picked vars randomly, measure time
- Do above step (everything random) many times and average
- Do above steps for \( n - 1, n - 2 \ldots \) until possible
- Plot graph — if done well, it is perfectly exponential
- But exponential factor \( \ll 2 \)
- To use: `--probBits N`
Overview

Flow diagram

Example usage

./grainofsalt --crypto grain --outputs 100
--genDeterBits 60

./grainofsalt --crypto grain --outputs 100 --stats
--cnfDir generatedCNFs --num 100 --deterBits 55
Outline

1. Context
   - SAT solvers
   - Stream ciphers

2. Translating shift register-based stream ciphers to SAT problems
   - Unwinding time
   - Describing feedback and filter functions
   - Help bits
Concluding remarks

- **grainofsalt** gives an integrated platform to conduct experiments with SAT solvers on shift register-based stream ciphers.
- It is GPL, with open GIT repository, well-documented, and extensible.
- [http://gitorious.com/grainofsalt/](http://gitorious.com/grainofsalt/)

Future work

- Add more ciphers — they are very simple to describe.
- Add more functionality — non-shift register based ciphers.
- Could be integrated into sage partially/fully.
Thank you for your time

Any questions?