SAT Solvers in the Context of Cryptography

v2.0

Presentation at Montpellier

Mate Soos

UPMC LIP6, PLANETE team INRIA, SALSA Team INRIA

10th of June 2010

Table of Contents

Outline

Motivations and goals

Motivations

- Cryptographic primitives may be broken using SAT solving tools
- Analysis of cryptographic primitives is possible using SAT solvers

Goals

- Show why SAT solvers work well to analyse and/or break cryptographic primitives
- Draw attention to the drawbacks a and how to overcome them

What is a SAT solver

Solves a problem in CNF

CNF is an "and of or-s"

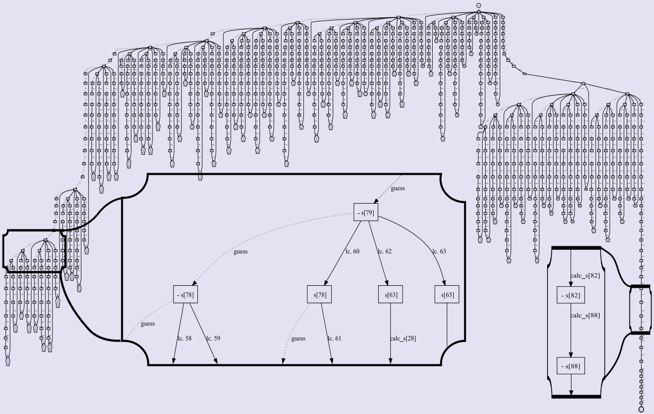
$$(x_1 \vee \neg x_3) \quad \wedge \quad (\neg x_2 \vee x_3) \quad \wedge \quad (x_1 \vee x_2)$$

Uses DPLL(φ) algorithm

1. If formula φ is trivial, return SAT/UNSAT
2. ret = DPLL(φ with $v \leftarrow$ `true`)
3. if ret == SAT, return SAT
4. ret = DPLL(φ with $v \leftarrow$ `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

SAT solver internals

Conflict clauses

- Generated when current assignment doesn't satisfy a clause
- Collection of information leading to conflict
- Used to avoid similar wrong parts of the tree next time

Most important parts

- Lazy data structures
- Learning (and forgetting)
- How to pick a variable
- When to restart

Cryptographic problems

Stream ciphers: Grain, HiTag2

- Generates pseudorandom keystream given public IV and secret key
- Step-by-step iteration is easy to describe in ANF
- ANF is relatively easy to convert to CNF

Block ciphers: DES, AES

- Encodes a plaintext to a chipertext given a secret key
- Can have relatively difficult internal parts e.g. S-box
- May be difficult to model in CNF

Hash functions: SHA-1

- Generates one-way, (second)preimage-resistant fingerprint of text
- Usually has relatively difficult internal parts e.g. circular left-shift
- Difficult to model in CNF

Outline

Advantages of SAT solvers in the context of cryptography

Brute force

- Try every setting of the unknowns
- Apply every setting to every equation
- If doesn't work, start from the beginning

SAT solvers

- Learnt clauses
  - Act as memory
  - Trim the search tree
- Lazy data structures
  - Fast partial back-tracking
  - Keep partially computed values in memory
- Variable activity heuristics
  - Find good points of entry
  - E.g. key bits, shift register states, etc.

Learnt clauses

Memory model

- When guesses were wrong $\Rightarrow$ *conflict*
- At conflict record how we came here $\Rightarrow$ *learnt clause*
- Learnt clauses act as memory

Learnt clause erasure strategy

- Learnt clause is active in new conflict $\Rightarrow$ *activity* $\uparrow$
- Other clauses' activity $\downarrow$
- Low-activity clauses are periodically *erased*

In the context of cryptography — demonstration

- `./cryptominisat --stats hitag2-shifted-31.cnf`
- `./cryptominisat --stats grain-shift-60.cnf`

Lazy data structures

Watchlists

- Only act upon a clause when we have to
- When all literals are assigned `false` except for one → assign free literal to `true`:

$$v1 \vee v2 \vee v3$$

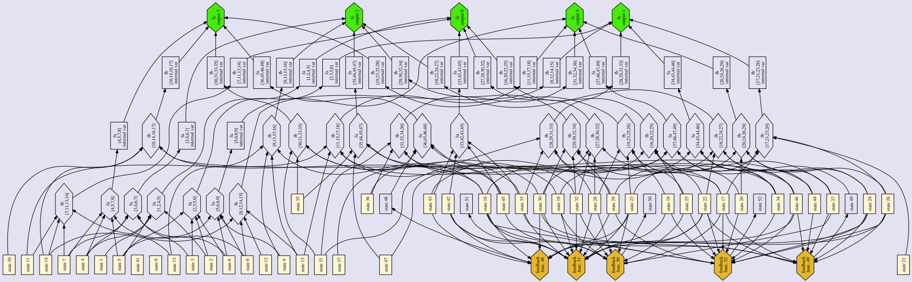
$$v1 = \texttt{false}, v2 = \texttt{false} \quad \longrightarrow \quad v3 = \texttt{true}$$

Internal variables

- It is possible to describe complex functions without internal variables using Karnaugh maps
- But it *slows down* the solving
- Many think they are a necessary evil. They in fact *help*
- They let the solver go back to a point in the search tree without the need to re-compute values

Lazy data structures

Example stream cipher

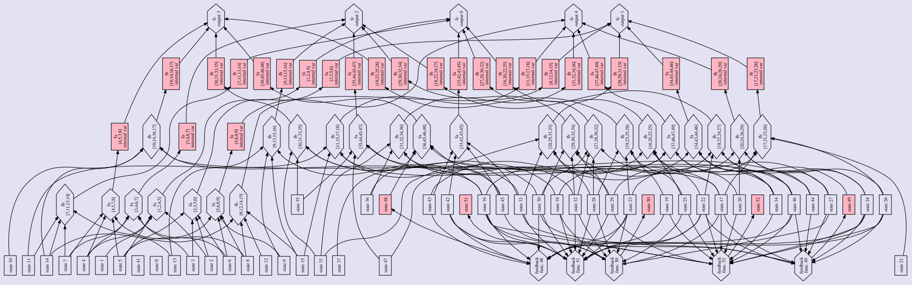


Explanation

- Hexagons: Filter and feedback functions
- Boxes: Variables (state and internal)
- Green: Final filter functions
- Yellow: Initial state
- Red: Feedback functions

Lazy data structures

Example stream cipher

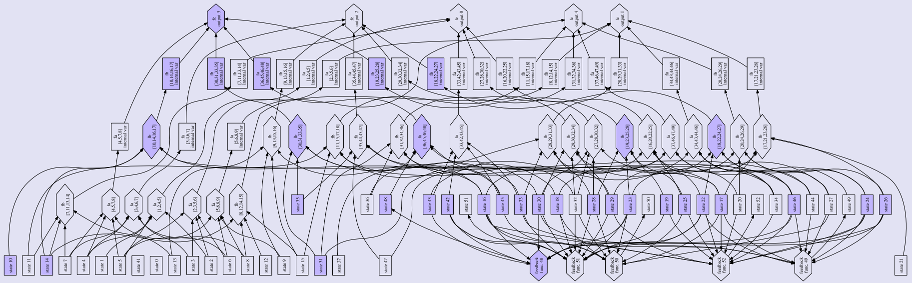


Explanation

- Hexagons: Filter and feedback functions
- Boxes: Variables (state and internal)
- Red: Internal variables

Lazy data structures

Example stream cipher



Explanation

- Hexagons: Filter and feedback functions
- Boxes: Variables (state and internal)
- Blue: Dependency of 4th output bit

Variable activity heuristics

Goal

- Make search tree balanced
- Divide the problem *equally*: $v =$ `true`, `false`
- Otherwise: unbalanced tree, search depth becomes huge
- Example:
  - Unbalanced tree: search depth = 1'000
  - Balanced tree: search depth = 100

Searching for a good branch

- Variable appears often in conflict $\Rightarrow$ activity $\uparrow$
- All other variables' activity $\downarrow$
- Most active variables branched first

Variable activity heuristics in the context of cryptography

Variable activity in crypto-problems

- Stream cipher with initialisation: branches on key bits
- Stream cipher without initialisation: branches on shift register state
- Algebraic side-channel attack: branches on internal variables of the side-channel information round

Demonstration

- HiTag2 — with initialisation
- Grain, HiTag2 — without initialisation, shifted 31,60 resp.:

- ASCA for PRESENT — not available

Outline

Disadvantages of SAT solvers in the context of cryptography

Problem structure lost

- CNFs is information-short
- Functions: Filter function? Bit-shift?
- Data: Side-channel information? Observed ciphertext?

Probabilities difficult to handle

- All clauses must be true
- How could we model $P(\text{information is correct}) = 0.4$?

Problem structure is lost

ANF vs. CNF

- Cannot find out that, e.g.

$$v1 \oplus v2 \oplus v3 = \texttt{true}$$
$$v1 \oplus v2 \oplus v4 = \texttt{true}$$
$$\therefore v3 = v4$$

- Sub-problems may be hard: e.g. Gauss elim.
- Result: simple problems become difficult

Internal variables — what do they represent?

- monomials, long xors $\Rightarrow$ too many vars
- Could disorientate variable activities
- Trivial problems may take hours to solve

Probabilities difficult to handle

Adding statistical information

- Clauses cannot have probabilities
- E.g: $P(v10 \lor v11 \lor v12 = \texttt{true}) = 0.4$
- Add multiple informations of low probability:

$$\mathbf{v1} \lor \mathbf{v2} \lor \mathbf{v3}$$

$$\begin{aligned} \text{where } v1 &\longleftrightarrow v10 \lor v11 \lor v12 \\ \text{and } v2 &\longleftrightarrow v20 \lor v21 \lor v22 \\ &\ldots \end{aligned}$$

Probabilistic information may lead to problems

- With (small) probability 0.6^3 : $v1 \lor v2 \lor v3 = \texttt{false}$
- Leads to UNSAT — need to re-start search
- Statistical information difficult to model

Outline

Conclusions

Concluding remarks

- SAT is effective at analysing cryptographic problems
- SAT can break simple cryptographic routines
- Complex ciphers: careful translation is needed

Future work

- Recover information from CNF
- Add information to the CNF
  e.g. variable categories: key, ciphertext, side-channel info
- Handle probabilistic information

Thank you for your time

Any questions?