Using parallel SAT solving algorithms to study the inversion of MD4 hash function

Irina Gribanova, Oleg Zaikin, Ilya Otpuschennikov, and Alexander Semenov

Matrosov Institute for System Dynamics and Control Theory
Irkutsk, Russia
Hash functions

*Hash function* - total computable discrete function of the kind

\[ \chi : \{0,1\}^* \rightarrow \{0,1\}^C, \]

where \( C \) is some constant representing a length of hash value.
Cryptographic hash functions

*Cryptographic hash functions* – hash functions which are infeasible to invert.

Input data – *message*.

Output data - *message digest, digest, hash*.

*Collision* – two different messages with the same digest.
Cryptographic hash functions

Cryptographic hash functions are used, for example, for password verification.

Storing a user password as cleartext is unsafety. Solution – to store the hash digest of a password.

The password presented by a user is hashed and compared with the stored hash.
The considered problem


The process of hash calculation is divided into 3 rounds, each of them consists of 16 steps.

We consider the following problem.

- Searching for preimages for the truncated 39-steps version of MD4 (for a given hash $h$ to find such $m$ that $MD4 - 39(m) = h$).
SAT approach

We reduced the aforementioned problem to SAT.

*Boolean satisfiability problem (SAT)* - for an arbitrary Boolean formula to determine if there exists such assignment of Boolean variables from this formula that makes it true.

Usually a Boolean formula in considered in the form of CNF (Conjunctive Normal Form).
SAT approach

Example of CNF with 3 clauses over 5 variables:

\[ C = (x_1 \lor \overline{x_2}) \cdot (x_2 \lor x_3 \lor \overline{x_4}) \cdot (\overline{x_3} \lor x_4 \lor \overline{x_5}) \]

This CNF is satisfiable, for example on (11001).

Real hard SAT problem can consist of thousands variables and millions clauses.

Real SAT problem can be unsatisfiable.
Searching for MD4 collisions

Authors of the paper * made SAT encoding of this problem. To find one MD4 collision it took them about 10 minutes in average.

We made more compact SAT encoding. To find 10 000 MD4 collisions it took less than 2 hours (i.e. less than a second in average for one collision).

Finding preimages for the weakened MD4

Despite the fact that the problem of finding collisions for MD4 can be effectively solved, the problem of finding preimages for the full version of this function remains computationally hard.

Authors of ** used SAT approach to invert 39-steps MD4 (totally it has 48 steps). In average it took about 8 hours on one CPU core.

Using our SAT solver PDSAT and our SAT encodings we managed solve it faster (3 h. on one CPU core).

Transalg system

Transalg – our system for encoding algorithms into SAT.

An algorithm should be written in C-like TA language.

All transformations are performed automatically.

Output – CNF.
Finding preimages for the weakened MD4

In our experiments we used the parallel SAT solver PDSAT.

- PDSAT is based on the partitioning approach.
- PDSAT works in master/slave scheme.
- In the estimation mode it searches for a decomposition set with good time estimations. (Monte Carlo approach and various optimization metaheuristics are used).
- In the solving mode PDSAT solves all SAT instances from a family obtained from a given decomposition set.
Finding preimages for the weakened MD4

We involved so called Dobbertin’s conditions, which make the inversion problem simpler by reducing the number of possible solutions.

11 out of 12 conditions were used in SAT encodings, the variables of the other 1 condition were varied (16 Boolean variables).

The problem was solved by PDSAT in 5 seconds on 5 computing nodes (160 CPU cores in total). Thus, if we recalculate this time on the case of a sequential launch, we obtain the time of about 13 minutes.
Thank you for your attention!