Uniform Reduction of Cryptographic Problems to SAT

Milan Šešum, Predrag Janičić
URL: www.matf.bg.ac.yu/~sesum, www.matf.bg.ac.yu/~janicic
Faculty of Mathematics, University of Belgrade, Serbia

Workshop on Formal and Automated Theorem Proving and Applications
Belgrade, Serbia, January 31, 2009.

Agenda

- Logical Cryptoanalysis
- Uniform Reduction to SAT
- Logical Cryptoanalysis of DES
- Further Work and Conclusions

What is logical cryptoanalysis

- Rather than using classical, problem-specific cryptoanalysis techniques, logical cryptoanalysis reduces cryptoanalysis tasks to SAT
- There are several (all recent) methods for logical cryptoanalysis
- Some approaches combine classical and logical cryptoanalysis

Reducing cryptoanalysis problems to SAT

- Many hard problems, including cryptoanalysis problems, can be reduced to SAT
- However, constructing of corresponding propositional formula, based on a problem specification is often tedious and error-prone
- It is welcome to have a methodology for automated reduction of cryptoanalysis problems to SAT

Uniform reduction to SAT

- One approach for this problem was proposed recently (Jovanović and Janičić, 2005)
- The approach generates SAT formulae that correspond to cryptoanalysis tasks by:
  - using an implementation of the cryptographic algorithm in C/C++
  - using polymorphism in C/C++

Uniform reduction to SAT - part II

- The approach was first used for cryptoanalysis of hash functions, both for the inversion problem and for finding collisions
- The approach was unable to solve the full problems, but solved weakened versions
- In addition, the approach produces hard satisfiable SAT formula (from the inversion problem) and hard unsatisfiable SAT formula (which is one of the biggest SAT challenges)

Polymorphism and Uniform Reduction to SAT

- Polymorphism is a programming language feature that allows values of different data types to be handled using a uniform interface
- Polymorphic functions are functions that can be applied to values of different types
- Operator overloading is a specific case of polymorphism, in which operators commonly used in programming are treated as polymorphic functions

Implementation

- We implemented two basic classes:
  - `Formula` — basic class of the hyerarchy of all types of formulas
  - `Word` — class which instances have to behave like integer data type, but instead of bits, these instances have to be arrays of formulas
- For class `Word` we had to overload all operators which arised in implementations of cryptographic functions which we used
- Overloading of logical operators is straightforward
- Overloading of arithmetic operators is bit more complicated because the value of a bit in the result depends on all previous bits of the operands

Implementation

- Main idea: use operator overloading to replace the standard computation by a generation of relevant formulas
- Example: overloading operator ~

```
Word Word::operator ~ () {
   Word notWord; // NOT of the word

   /* Compute NOT */
   for (int i = 0; i < bitArray.size(); i++) {
      Formula *f = new FormulaNot(bitArray[i]);
      notWord.setFormulaAt(i, f);
   }

   return notWord; // Return the calculated not
}
```

Logical Cryptoanalysis of MD5

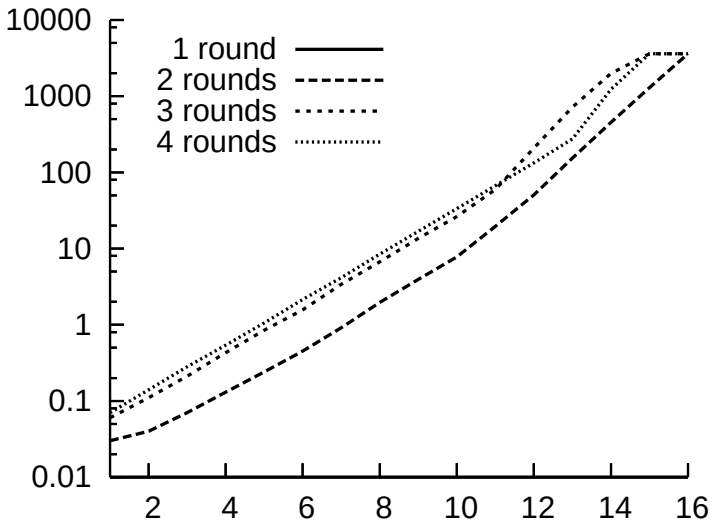
- We applied the above approach to the cryptoanalysis of MD5
- MD5 is widely used hash function
- It has an input (message) of arbitrary length and output (hash value) of length 128
- Hash function (*hash*) is required to have the following feature:
  - *preimage resistant* If hash value h is given it is computationally infeasible to find x such that $hash(x) = h$
  - *collisoin resistant* It is computationally infeasible to find two distinct messages x and y such that $hash(x) = hash(y)$

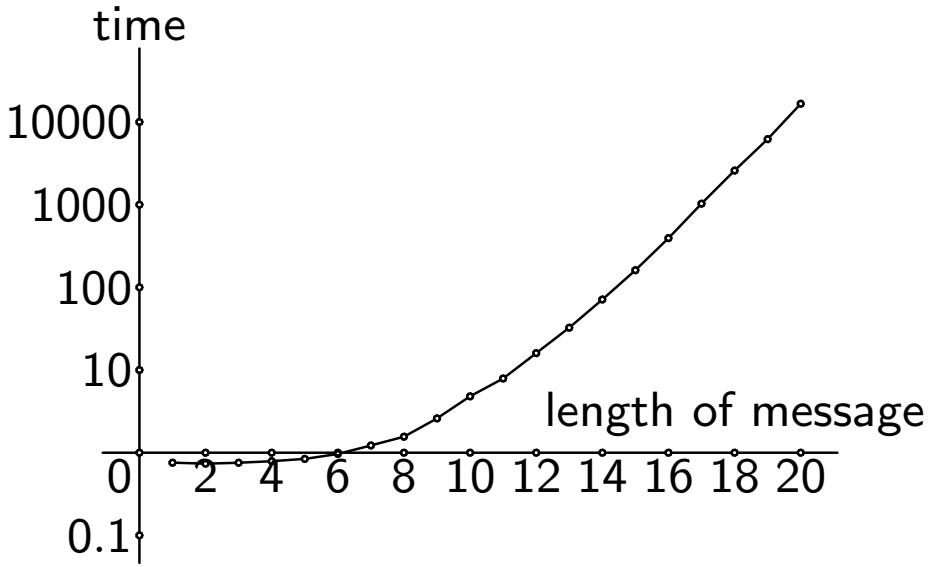
Our results on MD5

- For MD5 we assume that we know hash value and we tried to find the message
- Length of the message was varied in interval $[1, 20]$ bits
- If we assumed that length of the message is 20, our approach found the message in more then 4h

Comparison of results

- Comparison of existing results (Jovanović, Janičić) and results of our reimplementation applied on weakened MD5 (weaken by decreasing message's length)





Logical Cryptoanalysis of DES

- We applied the above approach to the cryptoanalysis of DES
- DES is one of the most popular symmetric cryptographic algorithms
- It has 16 rounds and uses 56 bits long keys
- Given one message and its cipher, the goal is to discover a secret key

Massacci's work

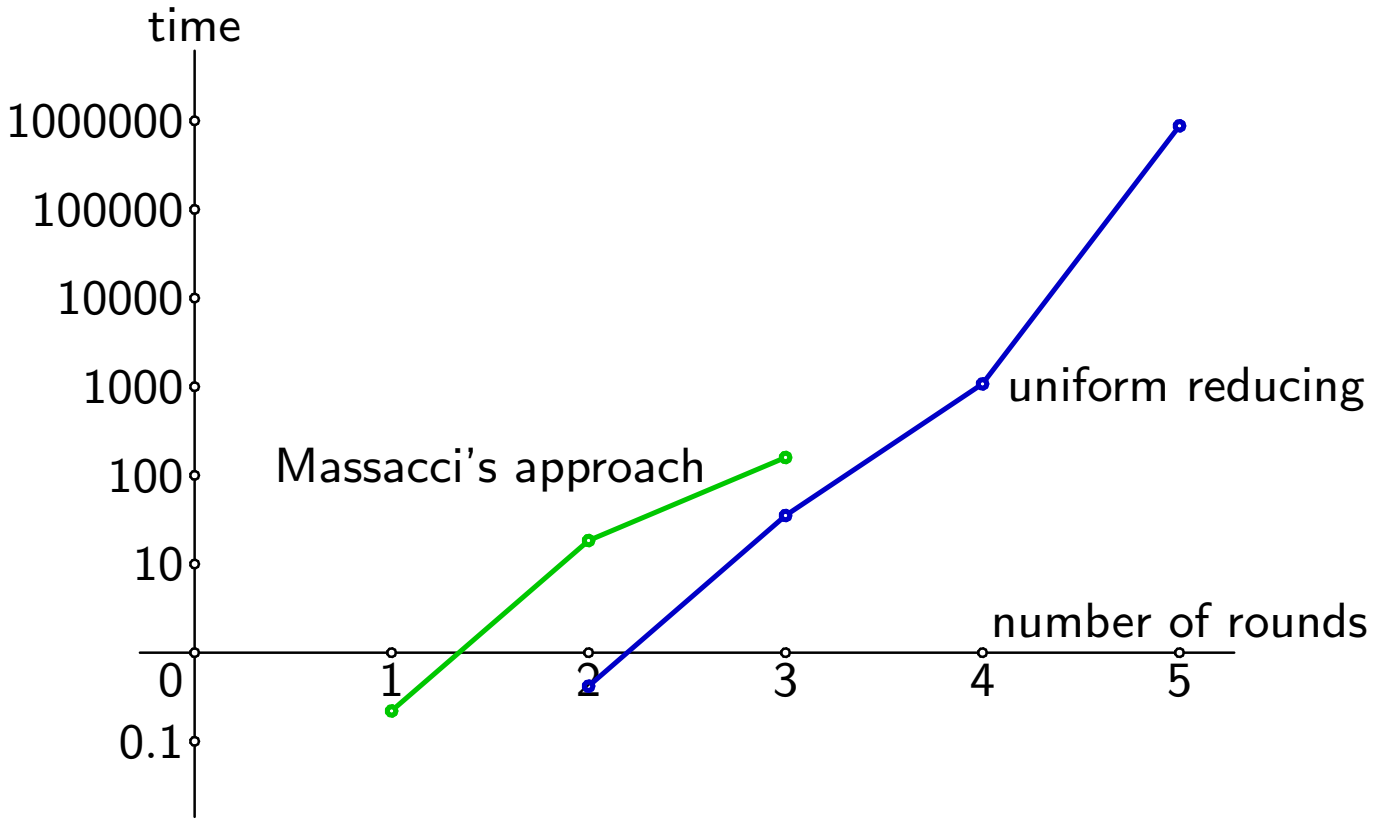
- The first logical cryptoanalysis of DES was done by Massacci et.al in 1999.
- They analyzed the specification of DES and hand-crafted a program that generate relevant SAT formulae
- Their method was able to break three rounds of DES (i.e., to discover a secret key if only 3 out of 16 rounds were used)

Our results

- For DES we assume that we know plain and corresponding cipher text and we tried to find the key
- As DES have 16 rounds, we used weak version of DES which consist of less number of rounds
- In number of rounds, our approach can break 5 rounds of DES

Our results

- Comparison of the uniform coding and Massacci's approach on weaken DES



Hard satisfiable and unsatisfiable instances of SAT problem

- Using this approach we can easily generate propositional formula corresponding to given cryptographic algorithm
- Those formulas obviously have a solution but it is not easy to find it
- This methodology gives us the mechanism for easy generation of hard and:
  - satisfiable SAT formulas (when formula correspond to problem of finding unknown key or message)
  - unsatisfiable SAT formulas (when formula correspond to problem of collision-resistant)

Further work

- The described approach can be applied to other cryptographic algorithms, but also to other domains
- It can be used for inverting different complex functions, given their implementation in C/C++
- It can be also used for solving many NP-complete problems, given the implementation of the test that some object is solution of the problem
- Inverting such test-function, actually solves the given problem

Conclusions

- We described one generic approach to logical cryptoanalysis
- We reimplemented the approach and applied to DES
- Our results are better than those obtained by domain-specific analysis
- The approach can be applied to many domains, which is a subject of our current work