An efficient method to solve a system of equations with elementary symmetric polynomials using SAT solvers

Jung Youl Park, Hong-Yeop Song

Coding and Cryptography Lab.,
TMS (BK21), Yonsei University

February 24, 2011

Outline

1. Introduction
   - Motivation
   - Approaches using an SAT solver
2. Our method
   - Background
   - The first approach by Bard et al.
   - Our construction
   - Analysis

Outline

An algebraic attack

How an algebraic attack works:

- Step 1:
  A cryptographic problem $\longrightarrow$ A system of algebraic equations
- Step 2:
  Solve a system of algebraic equations

How to obtain a system of equations

- direct analysis
  - analyze the cryptosystem and compute equations directly
  - Toyocrypt, E_0(bluetooth), NTRU, ...
- indirect analysis
  - find equations describing states of the cryptosystem best

Example : NTRU cryptosystem

Algebraic equations for NTRU cryptosystem (direct analysis)

- NTRU : a public key cryptosystem on a polynomial ring $\left(\frac{\mathbb{Z}}{q\mathbb{Z}}\right)[X]/\left(X^N - 1\right)$
- Using the Witt vector, its algebraic equations are obtained:

$$\sum_{i<j\in S_k} F_iF_j + h_{k,0}\sum_{i\in S_k} F_i + \sum_{i\in T_k} F_i + h_{k+1,0} + h_{k,1} = 0$$

$$\sum_{i<j\in U} F_iF_j = d_{f,1}$$

$$\sum_{i\in U} F_i = d_{f,0}$$

for $k = 0, \ldots, N-1$, and S_k, T_k and U are sets of indices.

How to solve algebraic equations?

Linear equations

Gaussian elimination. We're done.

Equations with degrees ≥ 2

- XL and its successors
  - Linearize higher-order terms and solve them
- Gröbner basis (F4, F5)
  - Compute a Gröbner basis for equations
  - Slow and requires too much memory
- An SAT solver
  - a system of quadratic equations over $\mathbb{F}_2$ $\longrightarrow$ a logical problem called a CNF-SAT problem
  - A CNF-SAT problem can be solved using SAT solvers
  - Requires significantly less memory than Gröbner basis

Solving by an SAT solver

Bard et al. (2006)

- Each quadratic term → new extra variable
- Cut a long linear equation to reduce the number of clauses
- Similar to XL
- Not good if there exist many quadratic terms

Park et al. (2010)

- Extension of Bard et al.'s work
- Focused on symmetric quadratic terms such as

$$a_1a_2 + a_1a_3 + a_1a_4 + a_2a_3 + a_2a_4 + a_3a_4$$

- Good if there exist large blocks of symmetric quadratic terms

Outline

Conjunctive Normal Form (CNF)

Conjunctive Normal Form

- Boolean variables — $a,\ b,\ x,\ y,\ v_1,\ \ldots$
- Literal — $a,\ \overline{a},\ b,\ \overline{b},\ x,\ \overline{x},\ \ldots$
- Clause — $a \vee \overline{x},\ b \vee \overline{x},\ \overline{a} \vee \overline{b} \vee x,\ \ldots$
- Conjunctive Normal Form — $(a \vee \overline{x}) \wedge (b \vee \overline{x}) \wedge (\overline{a} \vee \overline{b} \vee x)$

Satisfiability problem

- Satisfying assignment(s)
  - a set(s) of boolean values that make CNF true
- CNF-Satisfiability problem
  - Does there exist satisfying assignments?

Example

Example 1

T has a satisfying assignment of $T = \textit{true}$, or equivalently,
$T = 1 \quad \Leftrightarrow T + 1 = 0$

Example 2

$(a \vee \overline{x}) \wedge (b \vee \overline{x}) \wedge (\overline{a} \vee \overline{b} \vee x)$ has satisfying assignments of

$$(a, b, x) = \begin{cases} (0,0,0) \\ (0,1,0) \\ (1,0,0) \\ (1,1,1) \end{cases} \quad \Leftrightarrow \quad ab + x = 0$$

Example

Example 3

Satisfying assignments of $(a \vee \overline{x}) \wedge (\overline{a} \vee x)$ are

$$(a, x) = \begin{cases} (0,0) \\ (1,1) \end{cases} \quad \Leftrightarrow \quad a + x = 0$$

Example 4

Satisfying assignments of
$(a \vee b \vee \overline{x}) \wedge (a \vee \overline{b} \vee x) \wedge (\overline{a} \vee b \vee x) \wedge (\overline{a} \vee \overline{b} \vee \overline{x})$ are

$$(a, b, x) = \begin{cases} (0,0,0) \\ (0,1,1) \\ (1,0,1) \\ (1,1,0) \end{cases} \quad \Leftrightarrow \quad a + b + x = 0$$

Observation

A CNF for a equation

Given polynomial p, a CNF tautologically equivalent to an equation $p = 0$ is denoted by $CNF(p)$

CNFs for some simple equations

- $CNF(ab + x) = (a \vee \overline{x}) \wedge (b \vee \overline{x}) \wedge (\overline{a} \vee \overline{b} \vee x)$
- $CNF(a + x) = (a \vee \overline{x}) \wedge (\overline{a} \vee x)$
- $CNF(a + b + x) =$ $(a \vee b \vee \overline{x}) \wedge (a \vee \overline{b} \vee x) \wedge (\overline{a} \vee b \vee x) \wedge (\overline{a} \vee \overline{b} \vee \overline{x})$
- $CNF\left(a_1 + a_2 + \cdots + a_n\right)$ consists of 2^{n-1} clauses, where each clause is an arrangement of n variables, with odd number of negations less than n

Bard et al.'s work

Basic idea

- Linearization; replace quadratic terms with new extra variable
- Computing CNFs for both linear equations and quadratic replacements and combine them all
- Solve a CNF-Satisfiability problem by an SAT solver

Improvement : a cutting number

- $a_1 + a_2 + \cdots + a_9 = 0 \rightarrow 2^8 = 256$ clauses!
- $\begin{cases} a_1 + a_2 + a_3 + a_4 + u_1 = 0 \\ u_1 + a_5 + a_6 + a_7 + u_2 = 0 \\ u_2 + a_8 + a_9 = 0 \end{cases} \rightarrow 2^4 + 2^4 + 2^2 = 36$ clauses
- A best cutting number is reported to be 6

Example

$$v_1v_2 + v_1v_3 + v_1v_4 + v_2v_3 + v_2v_4 + v_3v_4 + v_2 + v_3 + v_4 + 1 = 0$$

$$\Rightarrow \begin{cases} v_1v_2 + v_{1,2} = 0 \\ v_1v_3 + v_{1,3} = 0 \\ v_1v_4 + v_{1,4} = 0 \\ v_2v_3 + v_{2,3} = 0 \\ v_2v_4 + v_{2,4} = 0 \\ v_3v_4 + v_{3,4} = 0 \\ v_{1,2} + v_{1,3} + v_{1,4} + v_{2,3} + v_{2,4} + u_1 = 0 \\ u_1 + v_{3,4} + v_2 + v_3 + v_4 + T = 0 \\ T = 1 \end{cases}$$

Observation

Recall the algebraic equations of NTRU cryptosystem

$$\sum_{i<j\in S_k} F_iF_j + h_{k,0}\sum_{i\in S_k} F_i + \sum_{i\in T_k} F_i + h_{k+1,0} + h_{k,1} = 0$$

$$\sum_{i<j\in U} F_iF_j = d_{f,1}$$

$$\sum_{i\in U} F_i = d_{f,0}$$

for $k = 0, \ldots, N-1$, and S_k, T_k and U are sets of indices.

Observation

- There exist many blocks of *symmetric* quadratic terms.
- How can we handle them whole and efficiently?

Notation

A set of indices

By S we denote a set of indices of boolean variables F_i's.

$$S = \{v_1, v_2, \ldots, v_n\}$$

An elementary symmetric polynomial of degree k for S

$$S^k = \sum_{1 \le i_1 < i_2 < \cdots < i_k \le n} F_{v_{i_1}} F_{v_{i_2}} \cdots F_{v_{i_k}}$$

A CNF for a polynomial p

We use $CNF(p + v_p)$ by introducing an extra variable v_p associated to p since we do not know whether $p = 0$ or not. ($p + v_p = 0$; v_p is a representative of p)

Main idea

Combining rules for sets

If $S = S_1 \dot{\cup} S_2 \dot{\cup} \cdots \dot{\cup} S_l$ (a disjoint union), then

$$S^1 = \sum_{i=1}^{l} S_i^1$$

$$S^2 = \sum_{i=1}^{l} S_i^2 + \sum_{1 \le i_1 < i_2 \le l} S_{i_1}^1 \cdot S_{i_2}^1$$

Main idea (cont'd)

Combining rules for CNFs for S^1

If $S = S_1 \dot\cup S_2 \dot\cup \cdots \dot\cup S_l$, then

$$S^1 = \sum_{i=1}^{l} S_i^1$$

implies

$$CNF\left(S^1 + v_{S^1}\right) =$$

$$CNF\left(v_{S^1} + \sum_{i=1}^{l} v_{S_i^1}\right) \wedge \bigwedge_{i=1}^{l} CNF\left(S_i^1 + v_{S_i^1}\right)$$

Main idea (cont'd)

Combining rules for CNFs for S^2

If $S = S_1 \dot\cup S_2 \dot\cup \cdots \dot\cup S_l$, then

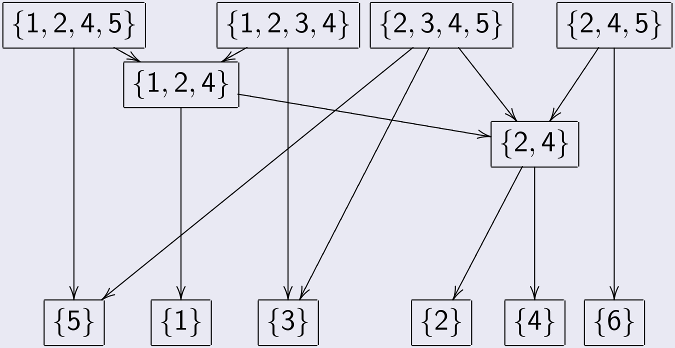
$$S^2 = \sum_{i=1}^{l} S_i^2 + \sum_{1 \le i_1 < i_2 \le l} S_{i_1}^1 \cdot S_{i_2}^1$$

implies

$$CNF\left(S^2 + v_{S^2}\right) = CNF\left(v_{S^2} + \sum_{i=1}^{l} v_{S_i^2} + \sum_{1 \le i < j \le l} v_{S_i^1} v_{S_j^1}\right) \wedge \bigwedge_{i=1}^{l} CNF\left(S_i^1 + v_{S_i^1}\right) \wedge \bigwedge_{i=1}^{l} CNF\left(S_i^2 + v_{S_i^2}\right)$$

A split graph

Figure: An example of a full split graph without simple nodes



Construction

Step 1

Represent a given system of equations as the following form:

$$\begin{aligned} L_1^1 + Q_{1,1}^2 + Q_{1,2}^2 + \cdots + Q_{1,m_1}^2 + C_1 &= 0 \\ L_2^1 + Q_{2,1}^2 + Q_{2,2}^2 + \cdots + Q_{2,m_2}^2 + C_2 &= 0 \\ &\vdots \\ L_n^1 + Q_{n,1}^2 + Q_{n,2}^2 + \cdots + Q_{n,m_n}^2 + C_n &= 0 \end{aligned}$$

where L_i's and $Q_{i,j}$'s are set of indices and C_i's are constants.

Step 2

Construct a full split graph from L_i's and $Q_{i,j}$'s.

Construction (cont'd)

Step 3

Construct CNFs for the followings:

$$L_i^1 + v_{L_i^1} = 0$$
$$Q_{i,j}^2 + v_{Q_{i,j}^2} = 0$$

for all i's and j's from the bottom of the graph to the top, using combining rules

Construction (cont'd)

Step 4

Construct CNFs for the followings:

$$v_{L_1^1} + v_{Q_{1,1}^2} + v_{Q_{1,2}^2} + \cdots + v_{Q_{1,m_1}^2} + C_1 = 0$$

$$\vdots$$

$$v_{L_n^1} + v_{Q_{n,1}^2} + v_{Q_{n,2}^2} + \cdots + v_{Q_{n,m_n}^2} + C_n = 0$$

Analysis on the size of CNFs

Theorem (Worst case of Bard et al.'s construction)

A CNF for a system has

$$O\left(\sum_{i=1}^{n}\left(|L_i| + \sum_{j=1}^{m_i}|Q_{i,j}|^2\right)\right)$$

clauses, literals and extra variables at most

Theorem (Worst case of Park et al.'s construction)

A CNF for a system has

$$O\left(\sum_{i=1}^{n}\left(|L_i| + \sum_{j=1}^{m_i}|Q_{i,j}|\right)\right)$$

clauses, literals and extra variables at most

Application to the case of NTRU cryptosystem

Bard et al.

A constructed CNF has

$$O\left(N^3\right)$$

clauses, literals and extra variables at most

Park et al.

A constructed CNF has

$$O\left(N^2\right)$$

clauses, literals and extra variables at most

Running time by an SAT solver

Table: Time taken in solving 100 instances for $N = 26$, 28 and 30.

N	Method		#var.	#cl.	#lit.	time (sec)
26	**Bard**	avg	1067	24309	141976	215.4
		max	1619	41772	246640	3678.9
	Park	avg	446	2119	6762	12.8
		max	518	2591	8459	86.4
28	**Bard**	avg	1218	27521	160626	2062.6
		max	2047	53843	318459	42237.7
	Park	avg	509	2438	7809	65.1
		max	577	2763	8922	242.9
30	**Bard**	avg	1503	34986	204778	15930.8
		max	2560	69042	409206	175840
	Park	avg	574	2753	8808	278.6
		max	652	3204	10414	1856.2