Solving Systems of Boolean Polynomials Using Binary Decision Diagrams

Oleksandr Kazymyrov\textsuperscript{1} \hspace{1cm} Håvard Raddum\textsuperscript{2}

\textsuperscript{1} University of Bergen \textsuperscript{2} Simula Research Laboratories
Norway
Solving equation systems

• Solving (non-linear) system of equations is NP-hard in general

• Several solving algorithms exist, which is the best?

• Equations may be represented as
  ✦ Boolean polynomials
  ✦ SAT formulas
  ✦ MRHS
  ✦ Binary Decision Diagrams (BDDs)
Binary Decision Diagrams (in this talk)

- Directed acyclic graph starting in one source node and ending in one sink node
- Drawn top to bottom, nodes in horizontal levels
- No edges between nodes on the same level
- At most two out-going edges from each node, called 0-edge and 1-edge
- Nodes on same level associated to some linear combination of variables
Examples
Constructing BDD systems
Constructing BDDs

• Easy construction of BDD from any Boolean polynomial

• May also construct BDD directly from non-linear components (S-boxes, mod $2^n$, bitwise AND …)
Boolean Equation to BDD

• $f(l_1(x),...,l_n(x)) = 1$

• Assign $f$ to source node, 1 to sink node and associate $l_1(x)$ to level 1 (top level)

• For $i=2,...,n$
  ✦ For each node $A$ on level $i-1$ (ass. to func. $g \neq 0$)
    • make two nodes on level $i$, connected to $A$ by 0-edge and 1-edge
    • assign $g|_{l_{i-1}(x)=0}$ and $g|_{l_{i-1}(x)=1} (\neq 0)$ to new nodes on level $i$
  ✦ Associate $l_i(x)$ to level $i$
Example

$$f(l_1, l_2, l_3, l_4) = l_1l_3 + l_1l_4 + l_2l_4 + l_2 + l_3 = 1$$
Example

\[ f(l_1, l_2, l_3, l_4) = l_1 l_2 l_3 + l_1 l_2 l_4 + l_2 l_4 + l_2 + l_3 = 1 \]
Example

\[ f(l_1, l_2, l_3, l_4) = l_1 l_2 l_3 + l_1 l_2 l_4 + l_2 l_4 + l_2 + l_3 = 1 \]
Example

\[ f(l_1, l_2, l_3, l_4) = l_1 l_2 l_3 + l_1 l_2 l_4 + l_2 l_4 + l_2 + l_3 = 1 \]
Example

\[ f(l_1, l_2, l_3, l_4) = l_1 l_2 l_3 + l_1 l_2 l_4 + l_2 l_4 + l_2 + l_3 = 1 \]
Example

$$f(l_1, l_2, l_3, l_4) = l_1l_2l_3 + l_1l_2l_4 + l_2l_4 + l_2 + l_3 = 1$$
Example

\[ f(l_1, l_2, l_3, l_4) = l_1l_2l_3 + l_1l_2l_4 + l_2l_4 + l_2 + l_3 = 1 \]
Example

\[ f(l_1, l_2, l_3, l_4) = l_1 l_2 l_3 + l_1 l_2 l_4 + l_2 l_4 + l_2 l_3 = 1 \]
Example

\[ f(l_1, l_2, l_3, l_4) = l_1 l_2 l_3 + l_1 l_2 l_4 + l_2 l_4 + l_2 + l_3 = 1 \]
Example

\[ f(l_1, l_2, l_3, l_4) = l_1 l_2 l_3 + l_1 l_2 l_4 + l_2 l_4 + l_2 + l_3 = 1 \]
Example

\[ f(l_1, l_2, l_3, l_4) = l_1 l_2 l_3 + l_1 l_2 l_4 + l_2 l_4 + l_2 + l_3 = 1 \]
Example

\[ f(l_1, l_2, l_3, l_4) = l_1l_2l_3 + l_1l_2l_4 + l_2l_4 + l_2 + l_3 = 1 \]
Example

\[ f(l_1, l_2, l_3, l_4) = l_1l_2l_3 + l_1l_2l_4 + l_2l_4 + l_2 + l_3 = 1 \]
Example

\[ f(l_1, l_2, l_3, l_4) = l_1 l_2 l_3 + l_1 l_2 l_4 + l_2 l_4 + l_2 + l_3 = 1 \]
Example

\[ f(l_1, l_2, l_3, l_4) = l_1 l_2 l_3 + l_1 l_2 l_4 + l_2 l_4 + l_2 + l_3 = 1 \]
Example

\[ f(l_1, l_2, l_3, l_4) = l_1 l_2 l_3 + l_1 l_2 l_4 + l_2 l_4 + l_2 + l_3 = 1 \]
Example

\[ f(l_1, l_2, l_3, l_4) = l_1l_2l_3 + l_1l_2l_4 + l_2l_4 + l_2 + l_3 = 1 \]
Example

$$f(l_1, l_2, l_3, l_4) = l_1 l_2 l_3 + l_1 l_2 l_4 + l_2 l_4 + l_2 + l_3 = 1$$
Constructing system

\[ \begin{align*}
    f_1(l_{11}, \ldots, l_{1k}) &= 1 \\
    f_2(l_{21}, \ldots, l_{2k}) &= 1 \\
    \vdots \\
    f_n(l_{n1}, \ldots, l_{nk}) &= 1
\end{align*} \]

- k relatively small, 

\[ l_{ij} = l_{ij}(x_1, \ldots, x_n) \]

- Build one BDD for each \( f_i \) (or non-linear component)

- Set of BDDs = representation of equation system (cryptographic primitive)
BDD representing S-box

<table>
<thead>
<tr>
<th>x</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>5</td>
<td>C</td>
<td>8</td>
<td>F</td>
<td>9</td>
<td>7</td>
<td>2</td>
<td>B</td>
<td>6</td>
<td>A</td>
<td>0</td>
<td>D</td>
<td>E</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>
BDD representing S-box

<table>
<thead>
<tr>
<th>x</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>5</td>
<td>C</td>
<td>8</td>
<td>F</td>
<td>9</td>
<td>7</td>
<td>2</td>
<td>B</td>
<td>6</td>
<td>A</td>
<td>0</td>
<td>D</td>
<td>E</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>
BDD representing S-box

<table>
<thead>
<tr>
<th>x</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>5</td>
<td>C</td>
<td>8</td>
<td>F</td>
<td>9</td>
<td>7</td>
<td>2</td>
<td>B</td>
<td>6</td>
<td>A</td>
<td>0</td>
<td>D</td>
<td>E</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

The diagram represents a BDD (Binary Decision Diagram) for an S-box. The input variables are $x_0, x_1, x_2, x_3$, and the output variables are $y_0, y_1, y_2, y_3$. The table maps these variables to hexadecimal values.
BDD representing S-box

\[ y = S(x) \]
Example - 4 round AES
Example - 4 round AES
Example - 4 round AES
Example - 4 round AES
Solving BDD systems
Paths = valid assignments

- Set of paths from source to sink nodes in BDD describes constraint of equation
- Selecting a path assigns values to linear combinations
- The edge out from a node on a level gives value to linear combination associated with that level
- One path gives right-hand side linear system
0. $x_{12} + x_{20} + x_{28} + x_{36} + x_{44} + x_{125} + x_{128}$

1. $x_{13} + x_{21} + x_{29} + x_{37} + x_{45} + x_{126} + x_{129}$

2. $x_{14} + x_{22} + x_{30} + x_{38} + x_{46} + x_{124} + x_{127} + x_{130}$

3. $x_{15} + x_{23} + x_{31} + x_{39} + x_{47} + x_{124} + x_{131}$

4. $x_{49} + x_{51} + x_{52} + x_{54}$

5. $x_{48} + x_{50} + x_{52} + x_{53} + x_{55}$

6. $x_{48} + x_{49} + x_{51} + x_{52} + x_{53} + x_{54}$

7. $x_{48} + x_{50} + x_{51} + x_{53} + x_{55}$
0. x12 + x20 + x28 + x36 + x44 + x125 + x128

1. x13 + x21 + x29 + x37 + x45 + x126 + x129

2. x14 + x22 + x30 + x38 + x46 + x124 + x127 + x130

3. x15 + x23 + x31 + x39 + x47 + x124 + x131

4. x49 + x51 + x52 + x54

5. x48 + x50 + x52 + x53 + x55

6. x48 + x49 + x51 + x52 + x53 + x54

7. x48 + x50 + x51 + x53 + x55
0. \(x_{12} + x_{20} + x_{28} + x_{36} + x_{44} + x_{125} + x_{128} = 1\)

1. \(x_{13} + x_{21} + x_{29} + x_{37} + x_{45} + x_{126} + x_{129} = 1\)

2. \(x_{14} + x_{22} + x_{30} + x_{38} + x_{46} + x_{124} + x_{127} + x_{130} = 0\)

3. \(x_{15} + x_{23} + x_{31} + x_{39} + x_{47} + x_{124} + x_{131} = 1\)

4. \(x_{49} + x_{51} + x_{52} + x_{54} = 0\)

5. \(x_{48} + x_{50} + x_{52} + x_{53} + x_{55} = 1\)

6. \(x_{48} + x_{49} + x_{51} + x_{52} + x_{53} + x_{54} = 0\)

7. \(x_{48} + x_{50} + x_{51} + x_{53} + x_{55} = 1\)
Naive solving attempt
Naive solving attempt

• Select a path from each BDD
Naive solving attempt

• Select a path from each BDD
• Collect linear systems from each BDD into one big linear system
Naive solving attempt

• Select a path from each BDD
• Collect linear systems from each BDD into one big linear system
• Solve big linear system
Naive solving attempt

- Select a path from each BDD
- Collect linear systems from each BDD into one big linear system
- Solve big linear system
- Solution found :-(
Naive failure
Naive failure

- Big linear system is overdefined, with lots of dependencies among linear combinations
Naive failure

• Big linear system is overdefined, with lots of dependencies among linear combinations

• Selected paths will, in all likelihood, lead to an inconsistent system
Naive failure

• Big linear system is overdefined, with lots of dependencies among linear combinations

• Selected paths will, in all likelihood, lead to an inconsistent system

• No solution :-(

Operations on BDDs

- We may manipulate a BDD to:
  - Reduce the BDD (remove redundant nodes)
  - Swap the linear combinations of two adjacent levels
  - Add (xor) the linear combinations of two adjacent levels
BDD Operations

- BDD reduction runs in polynomial time
- Swapping/adding levels are local operations, only affecting two levels involved
- May swap/add repeatedly to perform Gaussian elimination on linear combinations of BDD
Joining BDDs

• Two or more BDDs may be joined into one BDD very easily
  ✦ To join two BDDs, replace the sink node of one with the source node of the other
Three joined BDDs
Linear absorption

x_1 + x_4

x_2

x_3 + x_7

x_3 + x_4 + x_6

x_1 + x_3 + x_6

x_5 + x_7

Operations on a BDD
Linear absorption

\[ x_2 \]
\[ x_1 + x_4 \]
\[ x_3 + x_7 \]
\[ x_3 + x_4 + x_6 \]
\[ x_1 + x_3 + x_6 \]
\[ x_5 + x_7 \]

Operations on a BDD
Linear absorption

\[ \begin{align*}
x_2 \\
x_3 + x_7 \\
x_1 + x_4 \\
x_3 + x_4 + x_6 \\
x_1 + x_3 + x_6 \\
x_5 + x_7
\end{align*} \]
Linear absorption

Operations on a BDD
Linear absorption

\[ x_2 \]
\[ x_3 + x_7 \]
\[ x_1 + x_4 \]
\[ x_1 + x_3 + x_6 \]
\[ 0 \]
\[ x_5 + x_7 \]
Linear absorption

\[ x_2 \]
\[ x_3 + x_7 \]
\[ x_1 + x_4 \]
\[ x_1 + x_3 + x_6 \]
\[ 0 \]
\[ x_5 + x_7 \]
Level with 0-vector

• Level associated with 0-vector = 0-level
• Selecting 1-edge out from 0-level gives «0=1» assignment
• Remove all 1-edges out from nodes on 0-level
Linear absorption

\[ x_2 \]
\[ x_3 + x_7 \]
\[ x_1 + x_4 \]
\[ x_1 + x_3 + x_6 \]
\[ 0 \]
\[ x_5 + x_7 \]
Linear absorption

\[ x_2 \]
\[ x_3 + x_7 \]
\[ x_1 + x_4 \]
\[ x_1 + x_3 + x_6 \]
\[ 0 \]
\[ x_5 + x_7 \]
Linear absorption

\[ x_2 \]

\[ x_3 + x_7 \]

\[ x_1 + x_4 \]

\[ x_1 + x_3 + x_6 \]

\[ 0 \]

\[ x_5 + x_7 \]
Linear absorption

\[ x_2 \]

\[ x_3 + x_7 \]

\[ x_1 + x_4 \]

\[ x_1 + x_3 + x_6 \]

\[ x_5 + x_7 \]
General solving algorithm
General solving algorithm

- While more than 1 BDD in system
  - Join some BDDs (in some order) creating a BDD with linear dependencies
  - Absorb linear dependencies
General solving algorithm

• While more than 1 BDD in system
  ✦ Join some BDDs (in some order) creating a BDD with linear dependencies
  ✦ Absorb linear dependencies

• Any remaining path in final BDD gives right-hand side leading to consistent linear system
General solving algorithm

- While more than 1 BDD in system
  - Join some BDDs (in some order) creating a BDD with linear dependencies
  - Absorb linear dependencies
- Any remaining path in final BDD gives right-hand side leading to consistent linear system
- Solve linear system
Complexity

• Number of nodes on one level may (worst case) double when swapping or adding levels

• Absorbing one linear dependency may double the size of BDD

• In practice: very far from worst-case behavior
Practical results and examples
DES
DES

• 2007: an equation system for 6-round DES solved with MiniSat in 68 seconds (Courtois & Bard)
2007: an equation system for 6-round DES solved with MiniSat in 68 seconds (Courtois & Bard)

But… necessary to fix 20 bits of the key to correct values
• 2007: an equation system for 6-round DES solved with MiniSat in 68 seconds (Courtois & Bard)

• But… necessary to fix 20 bits of the key to correct values

• BDD system for 6-round DES solved in the same time without guessing (8 chosen plaintexts)
MiniAES

- There is no previous algebraic attacks on 10-round version (except CryptoMiniSAT)
MiniAES

• There is no previous algebraic attacks on 10-round version (except CryptoMiniSAT)

• The best previous attack is only for 2 rounds
MiniAES

• There is no previous algebraic attacks on 10-round version (except CryptoMiniSAT)

• The best previous attack is only for 2 rounds

• BDD approach allows to break full version of MiniAES using only 1 known plaintext
Determining EA-equivalence

- Two vectorial Boolean functions $F, G: \mathbb{GF}(2^n) \rightarrow \mathbb{GF}(2^n)$ are EA-equivalent if for all $x$

$$F(x) = M_1 \cdot G(M_2 \cdot x + V_2) + M_3 \cdot x + V_1$$

- $M_i$ are $n \times n$ matrices and $V_j$ are $n$-bit vectors, $M_1$ and $M_2$ are invertible

- May create equation system describing EA-equivalence, entries to $M_i$ and $V_j$ are variables (number of vars. is $3n^2 + 2n$)
## Finding EA-equivalence

- A few experiments for n=4 and n=5

<table>
<thead>
<tr>
<th>Instance</th>
<th>n</th>
<th>Number of solutions</th>
<th>BDD</th>
<th>Gröbner basis</th>
<th>CryptoMiniSat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>60</td>
<td>2</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>&gt;2</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>155</td>
<td>2</td>
<td>-</td>
<td>&gt;2</td>
</tr>
</tbody>
</table>

* Not finished after 78 hours
Conclusions

• New approaches to algebraic attacks development

• The BDD approach allows to reduce complexity of algebraic attack on DES by $2^{20}$

• Practical algebraic attack on 10-round MiniAES was presented for the first time

• In some cases the BDD method is more universal and shows the best results of all known methods