SAT Solver Attacks on CubeHash

Ben Bloom
Hash Functions

- Variable length input, fixed length output.
- $H(m) = h$
Cryptographic Hash Functions

- Preimage resistance
- Collision resistance
Applications of Cryptographic Hash Functions

- Data Integrity
- Digital Signatures
Collision Application

• Create two messages M1 and M2
• Have the Authority sign M1
• Take the signature on M1 and put it with M2
• Send M2 with the signature; others see M2 as authentic.
Satisfiability
Satisfiability

- NP-Complete problem
- Boolean expressions
- Conjunctive Normal Form
SAT Solvers

- Take a CNF expression, attempt to satisfy it.
- Complicated heuristics.
CubeHash
Overall Structure

1. Initialization
2. State i
3. XOR
4. r rounds
5. State i+1
6. More Blocks?
   - yes
   - no
7. Finalization
8. Hash is in first h/8 bytes
Round Function

- Made of simple operations
- Addition and Exclusive Or
- Rotation and Swap
Round Function

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
8 9 10 11 12 13 14 15 0 1 2 3 4 5 6 7 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 34 35 32 33 38 39 36 37 42 43 40 41 46 47 44 45
52 53 54 55 48 49 50 51 60 61 62 63 56 57 58 59 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79
80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 65 64 67 66 69 68 71 70 73 72 75 74 77 76 79 78 78
The Attack
The Attack

• Collision Attack
  – Two random messages, same hash.
Attack Design

Message 1

M1_B1  M1_B2  M1_B3

M1_B1 xor IV  IV

r rounds

M1_S2  M1_S2

M1_B2 xor M1_S2  M1_S2

r rounds

M1_S3  M1_S3

M1_B3 xor M1_S3  M1_S3

r rounds

M1_S4  M1_S4

Message 2

M2_B1  M2_B2  M2_B3

M2_B1 xor IV  IV

r rounds

M2_S2  M2_S2

M2_B2 xor M2_S2  M2_S2

r rounds

M2_S3  M2_S3

M2_B3 xor M2_S3  M2_S3

r rounds

M2_S4  M2_S4

!="
Building the CNF
CNF for the Round Function

- Each operation is a Gate.
- Two inputs, one output.
- Addition has three inputs, two outputs.
- Each simple gate is four CNF clauses.
CNF for the Round Function

- **XOR**
  - \((\neg A \lor \neg B \lor \neg C) (A \lor B \lor \neg C) (A \lor \neg B \lor C) (\neg A \lor B \lor C)\)

- **AND**
  - \((\neg A \lor \neg B \lor C) (\neg A \lor B \lor \neg C) (A \lor \neg B \lor \neg C) (A \lor B \lor \neg C)\)

- **OR**
  - \((\neg A \lor \neg B \lor C) (A \lor B \lor \neg C) (A \lor \neg B \lor C) (\neg A \lor B \lor C)\)
## CNF for the Round Function

- **Addition:** $A, B, Ci, Co, S, M, P, Q$
  - $M = A \text{xor} B$
  - $P = A \text{and} B$
  - $Q = M \text{and} Ci$

- **Carry-in of zero**
  - $(\text{xor}(A, B, S)) (\text{and}(A, B, Co))$

- **Ignore Carry-out**
  - $(\text{xor}(A, B, M)) (\text{xor}(M, Ci, S))$

- **General Case**
  - $(\text{xor}(A, B, M)) (\text{xor}(M, Ci, S)) (\text{and}(A, B, P)) (\text{and}(M, Ci, Q)) (\text{or}(P, Q, Co))$

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$Ci$</th>
<th>$Co$</th>
<th>$S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Variable Structure

- Round Function groups of 32.
- Rotations and Swaps taken care of here.
- Temporary Variables for Addition
CNF Variables

- \(((224 + 192 \times (r - 1)) \times 32 + 8 \times (128 - b)) \times num\_blocks \times 2 + 8 \times b \times num\_blocks + 8 \times (128 - b)\)
- About 7500 per round block.
CNF Clauses

- \((23808 \times r \times \text{num}_\text{blocks}) \times 2 + 1 + 4 \times (8 \times b \times \text{num}_\text{blocks}) + 7 \times 8 \times (128 - b) + 8 \times (128 - b) \times (\text{num}_\text{blocks} - 1) \times 5\)
- About 25,000 per round block.
Byte Order

- Little and Big Endian at the same time.
- Keeping byte order straight
The Solution
SAT Solver Output

• Solver provides the variable assignments which satisfy the expression.
• Variables must be reconstructed in the proper order.
Colliding the SAT solver output

- SAT solver doesn't provide everything.
- Must perform final operation ourselves.
- Make the states equal at the end.
The Programs

• Three total programs.
  − RoundCNFProducer
  − glucose_static
  − CNFSolutionToHex

• Two java built by me, one C++ built for a SAT competition.
Results
CubeHash 1/b-512, 1 block

Solve Time
1 round, 1 block

- Unsatisfiable
- Satisfiable

Seconds

Block size
CubeHash r/1-512

![Solve Time Graph](image.png)
Solve Time: CubeHash 1/b-512

Solve Time
1 round, 1-4 blocks

block size in bytes

Graph showing solve time for CubeHash 1/b-512 with varying block sizes.
Brute Force

- Expects to take $2^{n/2}$ hash function evaluations
- Figure out how long one evaluation takes, multiply by expected number of evaluations.
Compare to Brute Force
Compare to Brute Force
Future Work

- Variable numbers of message blocks
- Improved SAT solver technology
Conclusions

- SAT solvers show strong potential for Cryptographic applications.
- The SAT attack looks to do better than a brute force attack.
Questions?