Using MaxSAT to Correct Errors in AES Key Schedule Images

Xiaojuan Liao, Miyuki Koshimura, Hiroshi Fujita, Ryuzo Hasegawa

Hasegawa-Fujita Laboratory
Kyushu University

The 3rd CSPSAT2 Meeting

2013-7-26

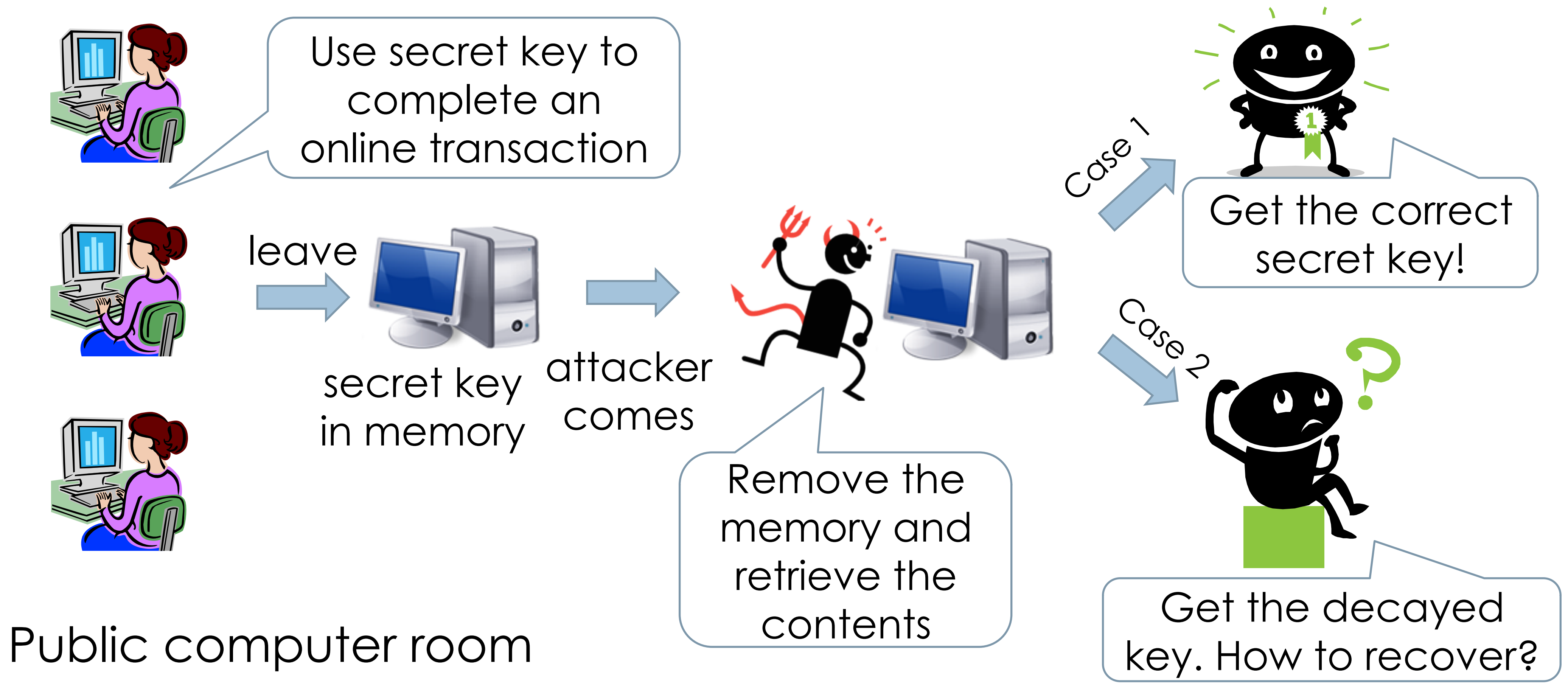
Outline

- Cold Boot Attack
- Advanced Encryption Standard (AES)
- Recover AES key Schedule
  - Using SAT solvers
  - Using MaxSAT solvers
  - Comparison
- Experiment
- Conclusion

Cold Boot Attack (1/2)

- DRAM:
  - Dynamic Random Access Memory
- DRAM cell:
  - a capacitor that is either 0 or 1
  - 0: ground state
  - 1: charged state
- DRAM remanence:
  - DRAM retains its contents for a period (seconds) after power is lost
  - As time goes on, data may decay and eventually disappear
- Cold boot attack:
  - Exploit DRAM remanence to access sensitive data (e.g., encryption keys)

A Scenario of Cold Boot Attack


Use secret key to complete an online transaction
leave
secret key in memory
attacker comes
Remove the memory and retrieve the contents
Case 1
Get the correct secret key!
Case 2
Get the decayed key. How to recover?
Public computer room

Cold Boot Attack (2/2)

- Decay patterns [1]
  - Decay aggravates as time goes on
  - Most bits decay to ground states ($1 \rightarrow 0$)
  - Only a small fraction (0.1%) flips to charged states ($0 \rightarrow 1$)
- This work
  - Recover AES keys from decayed bits

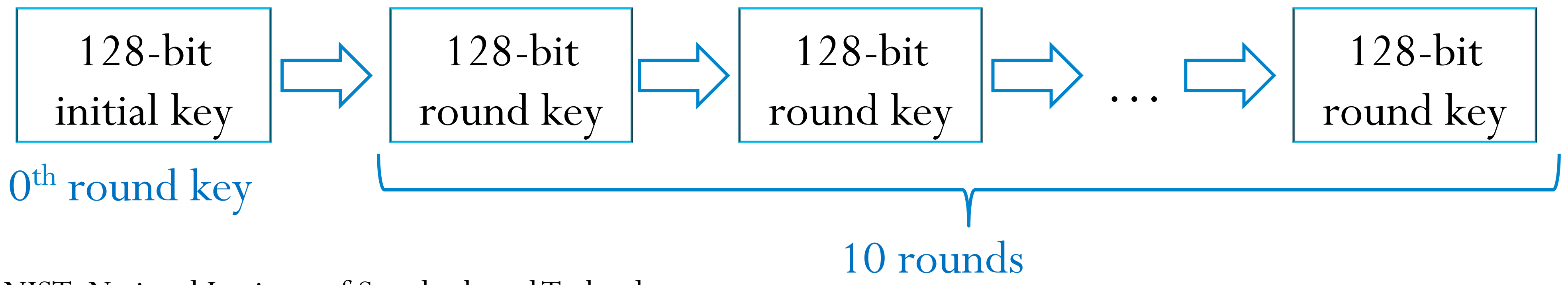
[1] J. Alex Halderman et al., Lest We Remember: Cold Boot Attacks on Encryption Keys. USENIX08.

Outline

- Cold Boot Attack
- Advanced Encryption Standard (AES)
- Recover AES key Schedule
  - Using SAT solvers
  - Using MaxSAT solvers
  - Comparison
- Experiment
- Conclusion

Advanced Encryption Standard(AES)

- What is AES
  - A specification for the encryption of electronic data established by the U.S. NIST [1]
  - Adopted by the U.S. government and used worldwide
- AES key [2]
  - Initial key length: 128 bits (options:192 bits, 256 bits)
  - AES-128 key schedule:



[1] NIST: National Institute of Standards and Technology

[2] Federal Information Processing, Announcing the ADVANCED ENCRYPTION STANDARD (AES), 2001.

Example of Key Schedule

Original key schedule

0th:
0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0
0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0
1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0
0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0

↓

Key Expansion Algorithm (public algorithm)

↓

1st:
1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1
1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1
0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1
0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1

↓

Key Expansion Algorithm (public algorithm)

↓

:

↓

10th:
1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0
1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1
1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0
1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0

Decayed key schedule

0th:
0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0
0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0

1st:
1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1
1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1
0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1

reverse flipping error

10th:
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0
0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0

Goal: correct errors in the decayed key schedule

Key Expansion Algorithm

Given the 128-bit initial key, the following equations characterize bit-relations among the bits in the last 10 round keys:

$$\begin{tikzcd}
\boxed{\text{(r-1)th round key bit}} \arrow[r, "\text{AES-128 key expansion algorithm}"] & \boxed{\text{rth round key bit}}
\end{tikzcd}$$

$$b_i^r = b_i^{r-1} \oplus S_{i \bmod 8}\left(B_{105+8\cdot\lfloor i/8 \rfloor}^{r-1}\right) \oplus R_i(r), \qquad 0 \le i \le 31$$

$$b_i^r = b_i^{r-1} \oplus b_{i-32}^r, \qquad 32 \le i \le 127$$

b_i^r : ith bit of the rth round key, 1≦r≦10, 0≦i≦127

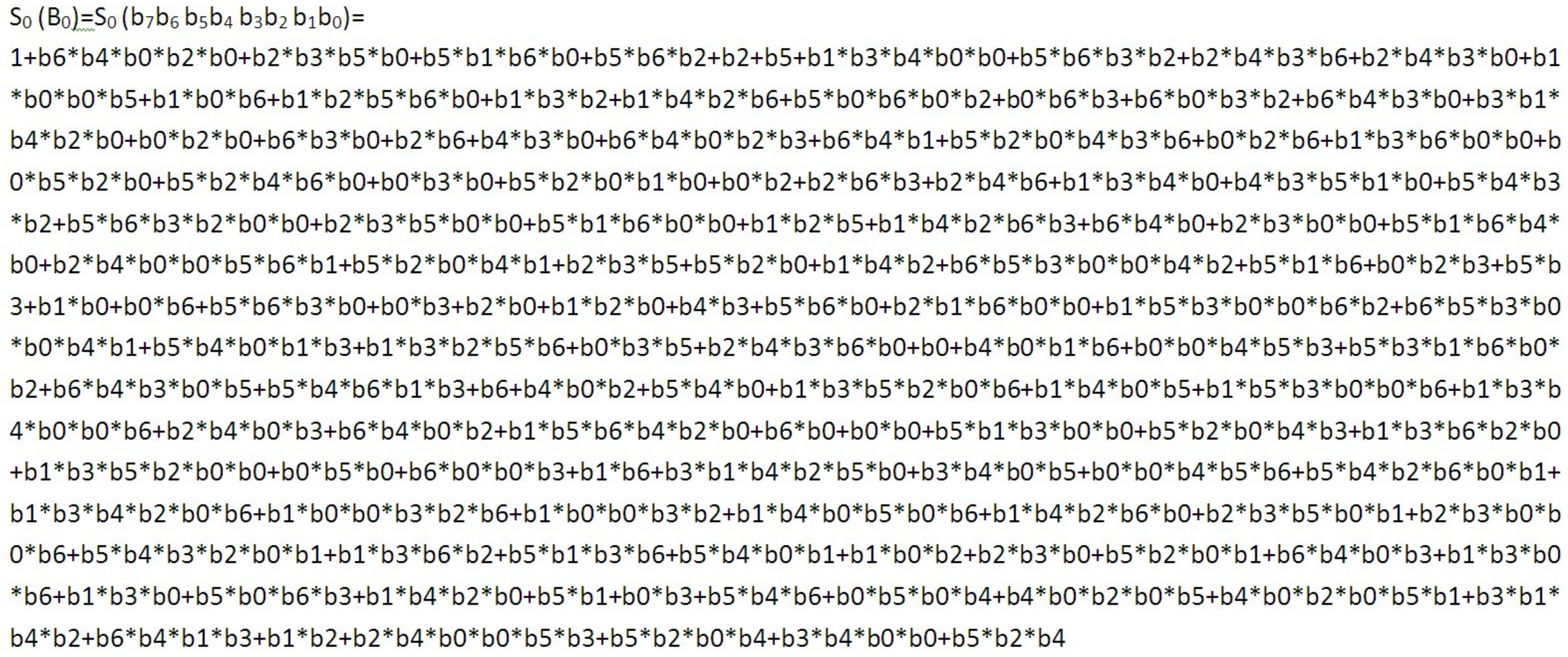
b_i^0 : ith bit of the 0th round key, copied from the initial key, 0≦i≦127

$R_i(r)$: ith bit of a round-dependent word, 0≦i≦31, 1≦r≦10

$S_x(B_i^r)$: an S-Box function in algebraic normal form (ANF), 0≦x≦7

input: $B_i^r = \{b_i^r, b_{i+1}^r, \mathrm{K}\ , b_{i+7}^r\}$, output: one bit

Example of $S_0(B_0)$



$S_0\ (B_0)=S_0\ (b_7b_6\ b_5b_4\ b_3b_2\ b_1b_0)=$

```
1+b6*b4*b0*b2*b0+b2*b3*b5*b0+b5*b1*b6*b0+b5*b6*b2+b2+b5+b1*b3*b4*b0*b0+b5*b6*b3*b2+b2*b4*b3*b6+b2*b4*b3*b0+b1
*b0*b0*b5+b1*b0*b6+b1*b2*b5*b6*b0+b1*b3*b2+b1*b4*b2*b6+b5*b0*b6*b0*b2+b0*b6*b3+b6*b0*b3*b2+b6*b4*b3*b0+b3*b1*
b4*b2*b0+b0*b2*b0+b6*b3*b0+b2*b6+b4*b3*b0+b6*b4*b0*b2*b3+b6*b4*b1+b5*b2*b0*b4*b3*b6+b0*b2*b6+b1*b3*b6*b0*b0+b
0*b5*b2*b0+b5*b2*b4*b6*b0+b0*b3*b0+b5*b2*b0*b1*b0+b0*b2+b2*b6*b3+b2*b4*b6+b1*b3*b4*b0+b4*b3*b5*b1*b0+b5*b4*b3
*b2+b5*b6*b3*b2*b0*b0+b2*b3*b5*b0*b0+b5*b1*b6*b0*b0+b1*b2*b5+b1*b4*b2*b6*b3+b6*b4*b0+b2*b3*b0*b0+b5*b1*b6*b4*
b0+b2*b4*b0*b0*b5*b6*b1+b5*b2*b0*b4*b1+b2*b3*b5+b5*b2*b0+b1*b4*b2+b6*b5*b3*b0*b0*b4*b2+b5*b1*b6+b0*b2*b3+b5*b
3+b1*b0+b0*b6+b5*b6*b3*b0+b0*b3+b2*b0+b1*b2*b0+b4*b3+b5*b6*b0+b2*b1*b6*b0*b0+b1*b5*b3*b0*b0*b6*b2+b6*b5*b3*b0
*b0*b4*b1+b5*b4*b0*b1*b3+b1*b3*b2*b5*b6+b0*b3*b5+b2*b4*b3*b6*b0+b0+b4*b0*b1*b6+b0*b0*b4*b5*b3+b5*b3*b1*b6*b0*
b2+b6*b4*b3*b0*b5+b5*b4*b6*b1*b3+b6+b4*b0*b2+b5*b4*b0+b1*b3*b5*b2*b0*b6+b1*b4*b0*b5+b1*b5*b3*b0*b0*b6+b1*b3*b
4*b0*b0*b6+b2*b4*b0*b3+b6*b4*b0*b2+b1*b5*b6*b4*b2*b0+b6*b0+b0*b0+b5*b1*b3*b0*b0+b5*b2*b0*b4*b3+b1*b3*b6*b2*b0
+b1*b3*b5*b2*b0*b0+b0*b5*b0+b6*b0*b0*b3+b1*b6+b3*b1*b4*b2*b5*b0+b3*b4*b0*b5+b0*b0*b4*b5*b6+b5*b4*b2*b6*b0*b1+
b1*b3*b4*b2*b0*b6+b1*b0*b0*b3*b2*b6+b1*b0*b0*b3*b2+b1*b4*b0*b5*b0*b6+b1*b4*b2*b6*b0+b2*b3*b5*b0*b1+b2*b3*b0*b
0*b6+b5*b4*b3*b2*b0*b1+b1*b3*b6*b2+b5*b1*b3*b6+b5*b4*b0*b1+b1*b0*b2+b2*b3*b0+b5*b2*b0*b1+b6*b4*b0*b3+b1*b3*b0
*b6+b1*b3*b0+b5*b0*b6*b3+b1*b4*b2*b0+b5*b1+b0*b3+b5*b4*b6+b0*b5*b0*b4+b4*b0*b2*b0*b5+b4*b0*b2*b0*b5*b1+b3*b1*
b4*b2+b6*b4*b1*b3+b1*b2+b2*b4*b0*b0*b5*b3+b5*b2*b0*b4+b3*b4*b0*b0+b5*b2*b4
```

ANF formula: $(b_1 \wedge b_2) \oplus (b_2 \wedge b_3 \wedge b_4) \oplus L$

+: xor operation

*: and operation

Example of Bit Representation

b_0^0

0th: 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0
0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0
1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0
0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0

$b_{105}^0, b_{106}^0, b_{107}^0, b_{108}^0, b_{109}^0, b_{110}^0, b_{111}^0, b_{112}^0$ ↓

Key Expansion Algorithm (public algorithm)

↓

b_0^1

1st: 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1
1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1
0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1
0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1

$$b_0^1 = b_0^0 \oplus S_0\left(b_{105}^0, b_{106}^0, \ldots, b_{112}^0\right) \oplus R_0(1) \qquad b_{127}^1 = b_{127}^0 \oplus b_{95}^1$$

Summary: each bit in the latter 10 rounds is computed from its former bits → bit-relations

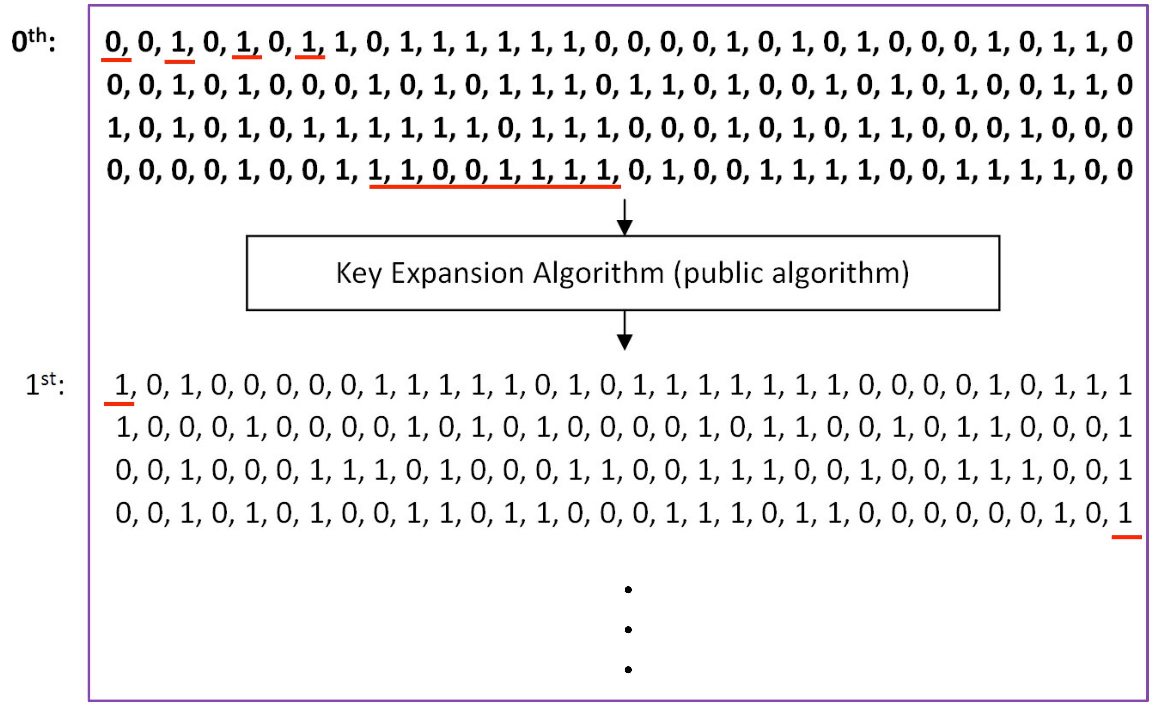
Outline

- Cold Boot Attack
- Advanced Encryption Standard (AES)
- Recover AES key Schedule
  - Using SAT solvers
  - Using MaxSAT solvers
  - Comparison
- Experiment
- Conclusion

Assumption of AES Key Recovery

- Assumption
  - Perfect assumption
    - Decay occurs only on 1s
    - No reverse flipping errors: no 0s flip to 1
  - Realistic assumption
    - Decay occurs mainly on 1s
    - A few reverse flipping errors: 0.1% of 0s flip to 1
- This work
  - Recover AES-128 key schedule based on the realistic assumption

Recover AES keys using SAT solvers



0th:

0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0
0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0
1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0
0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0

Key Expansion Algorithm (public algorithm)

1st:

1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1
1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1
0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1
0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1

⋮

Perfect assumption [1]

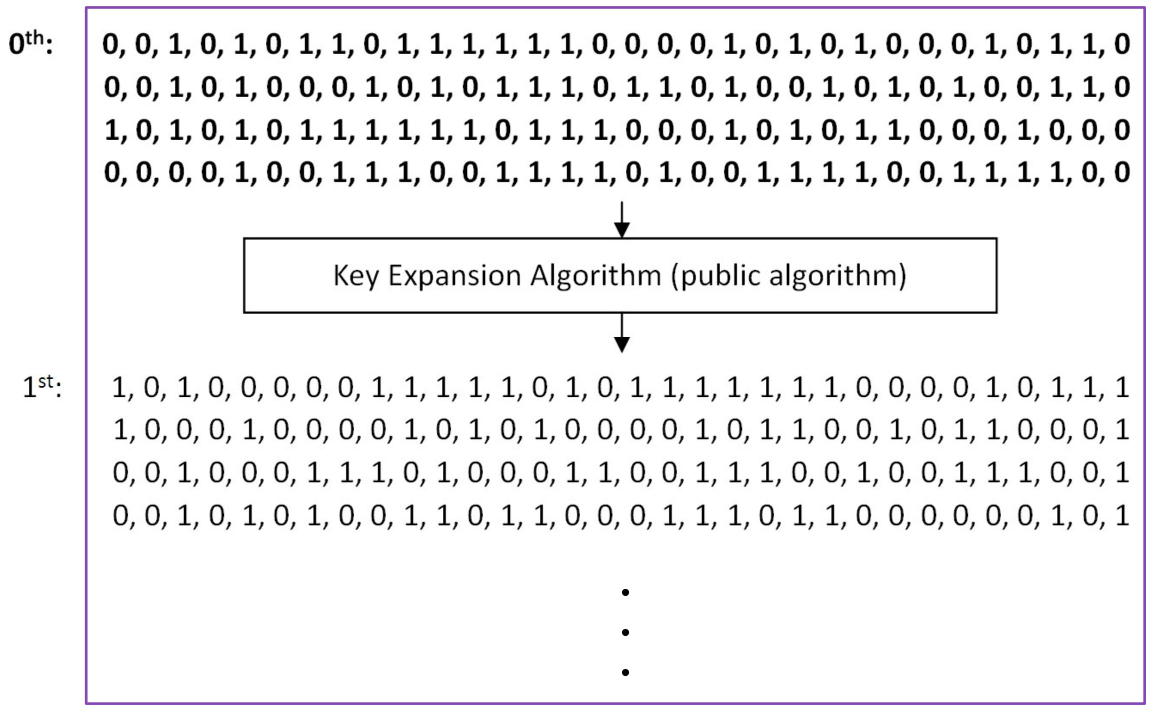
➢No bit flips from 0 to 1

➢All 1s in the key schedule are correct

1. All 1s in the key schedule are treated as hard constraints
   $\underline{b_2^0 = 1}, \underline{b_4^0 = 1}, \underline{b_6^0 = 1}, \mathrm{K}\ \underline{b_{127}^1 = 1}, \mathrm{K}$
2. All bit-relations are treated as hard constraints
   $$\left.\begin{array}{l} \underline{b_0^1 = b_0^0 \oplus S_0\left(B_{105}^0\right) \oplus R_0(1)} \\ b_1^1 = b_1^0 \oplus S_1\left(B_{105}^0\right) \oplus R_1(1) \\ \mathrm{L} \\ b_{127}^1 = b_{127}^0 \oplus b_{95}^1 \end{array}\right\}$$
   No. of formulas: 128*10=1280
   L
3. CryptoMiniSat: support XOR operation natively
   e.g., understand $x_1 \oplus x_2 \oplus x_3 = 1$

[1] Kamal et al., Applications of SAT solvers to AES key recovery from decayed key schedule images, 2010.

Recover AES keys using SAT solvers



0th:

0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0
0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0
1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0
0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0

1st:

1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1
1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1
0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1
0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1

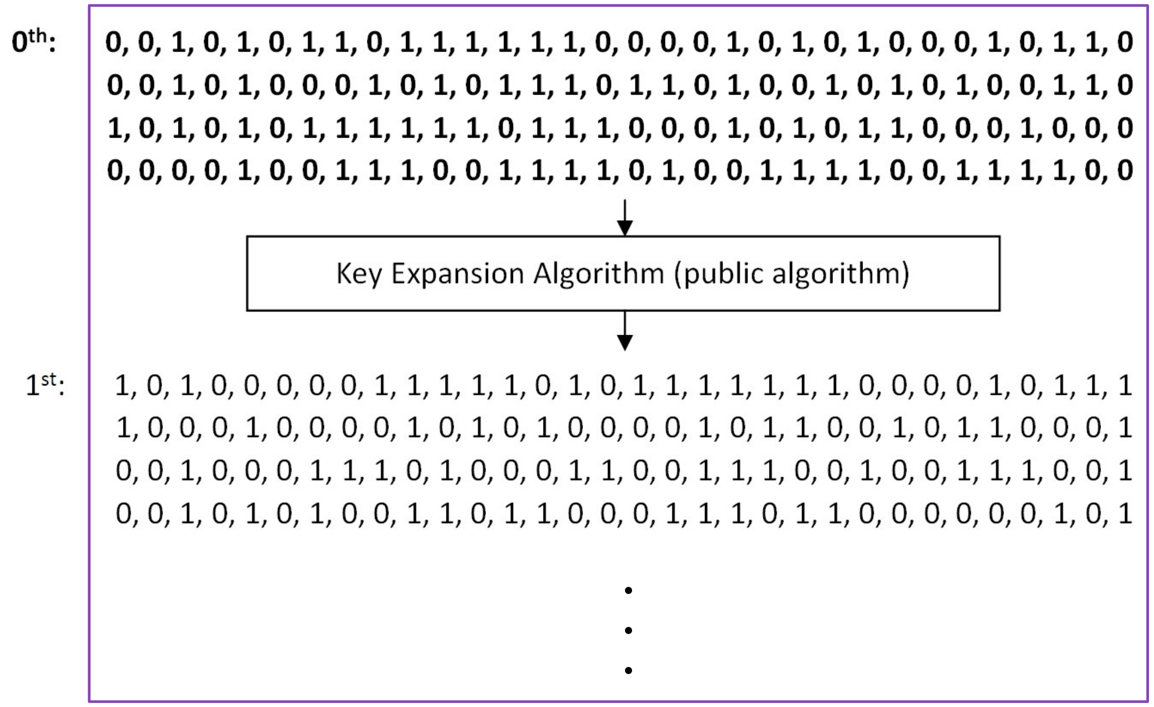
$\vdots$

$S_x(B_i)$: ANF formula, XORing conjunctions of variables, e.g., $\underline{(b_1 \wedge b_2)} \oplus \underline{(b_2 \wedge b_3 \wedge b_4)} \oplus \mathrm{L}$

$$\Rightarrow \begin{cases} x_1 \Leftrightarrow b_1 \wedge b_2,\quad x_2 \Leftrightarrow b_2 \wedge b_3 \wedge b_4, \mathrm{L} \\ x_1 \oplus x_2 \oplus \mathrm{L} \end{cases}$$

1. All 1s in the key schedule are treated as hard constraints
   $b_2^0 = 1, b_4^0 = 1, b_6^0 = 1, \mathrm{K}\ b_{127}^1 = 1, \mathrm{K}$
2. All bit-relations are treated as hard constraints
   $$b_0^1 = b_0^0 \oplus S_0\left(B_{105}^0\right) \oplus R_0(1)$$
   $$b_1^1 = b_1^0 \oplus S_1\left(B_{105}^0\right) \oplus R_1(1)$$
   $$\mathrm{L}$$
   $$b_{127}^1 = b_{127}^0 \oplus b_{95}^1$$
   $$\mathrm{L}$$
3. CryptoMiniSat: support XOR functions natively
   e.g., $x_1 \oplus x_2 \oplus x_3 = 1$

Recover AES keys using SAT solvers


0th:

0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0
0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0
1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0
0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0

Key Expansion Algorithm (public algorithm)

1st:

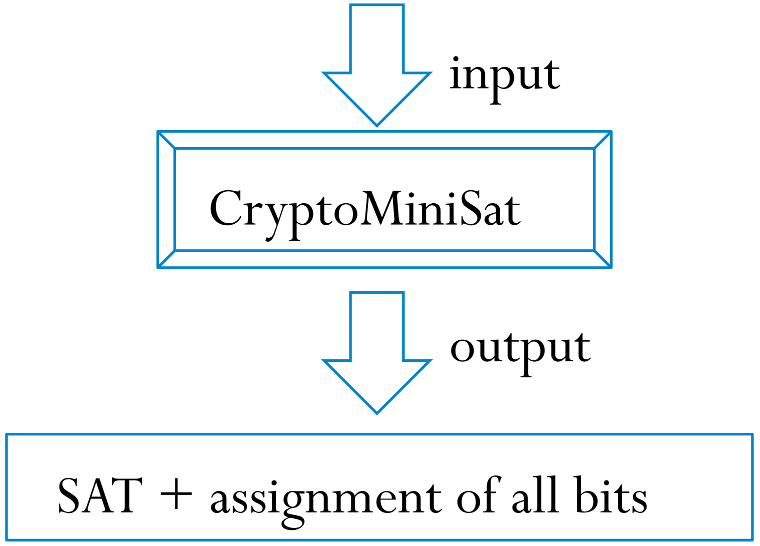
1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1
1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1
0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1
0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1

⋮

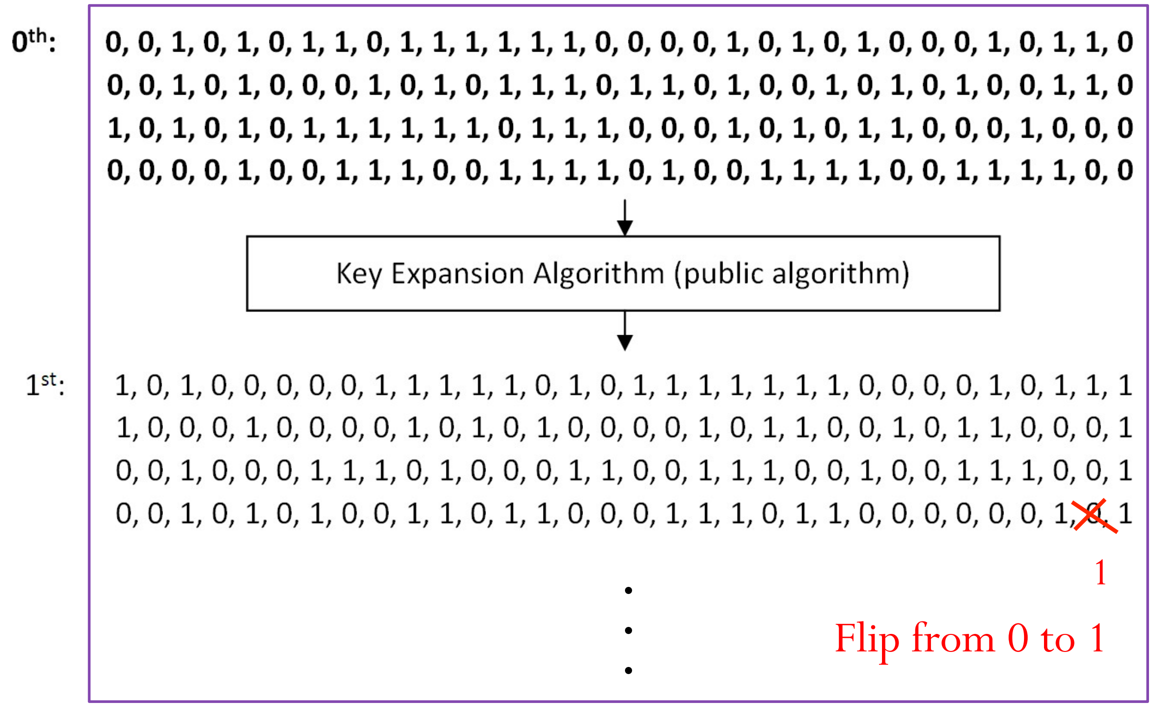
Perfect assumption

- ➢No bit flips from 0 to 1
- ➢All 1s in the key schedule are correct

1. All 1s in the key schedule are treated as hard constraints
   $b_2^0 = 1, b_4^0 = 1, b_6^0 = 1, \mathrm{K}\ b_{127}^1 = 1, \mathrm{K}$
2. All bit-relations are treated as hard constraints



Recover AES keys using SAT solvers



0th:

0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0
0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0
1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0
0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0

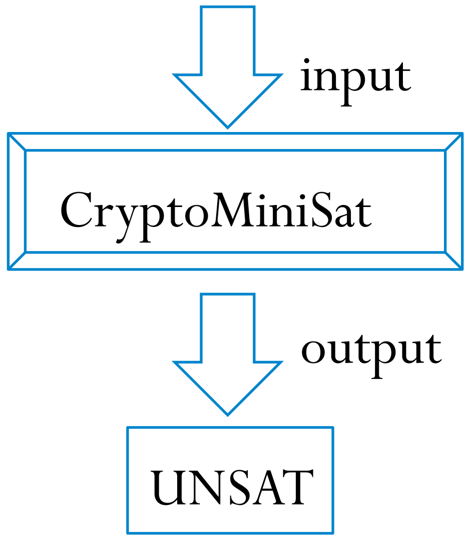
1st:

1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1
1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1
0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1
0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, $\overset{}{\underset{1}{\cancel{0}}}$, 1

⋮

Flip from 0 to 1

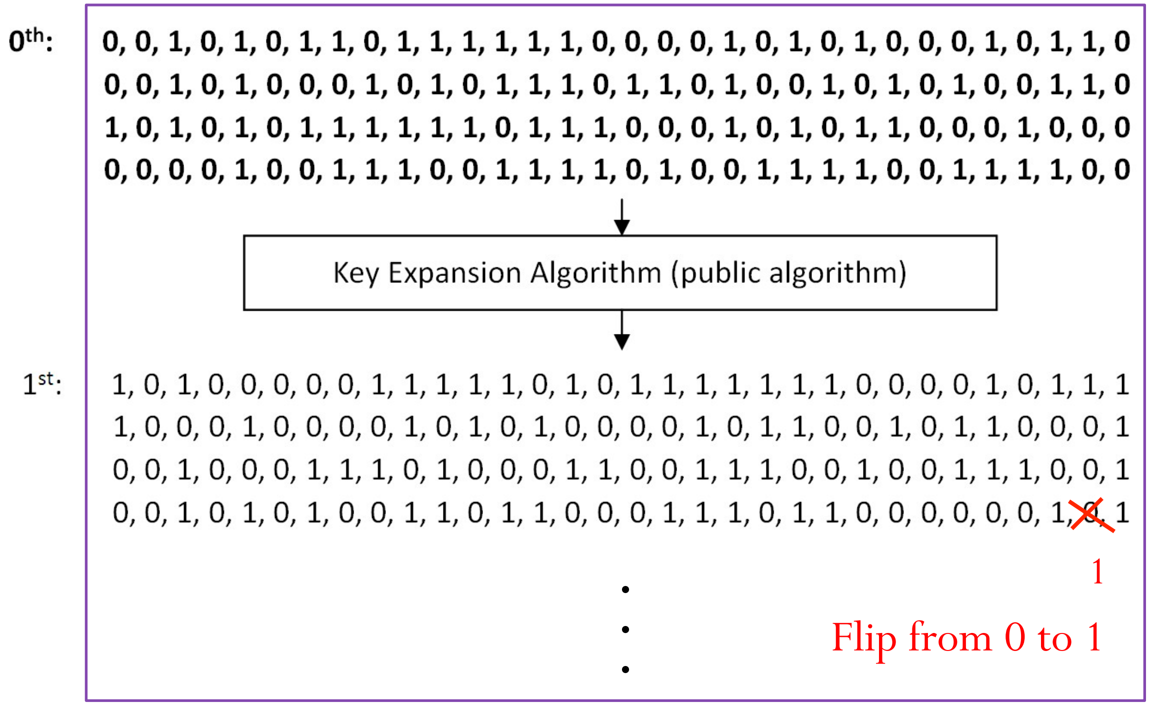
1. All 1s in the key schedule are treated as hard constraints
$$b_2^0 = 1, b_4^0 = 1, b_6^0 = 1, \ldots\ \underline{b_{126}^1 = 1}, \ldots$$
incorrect
2. All bit-relations are treated as hard constraints



Realistic assumption

- Some bits flip from 0 to 1
- Not all 1s in the key schedule are correct

Recover AES keys using SAT solvers

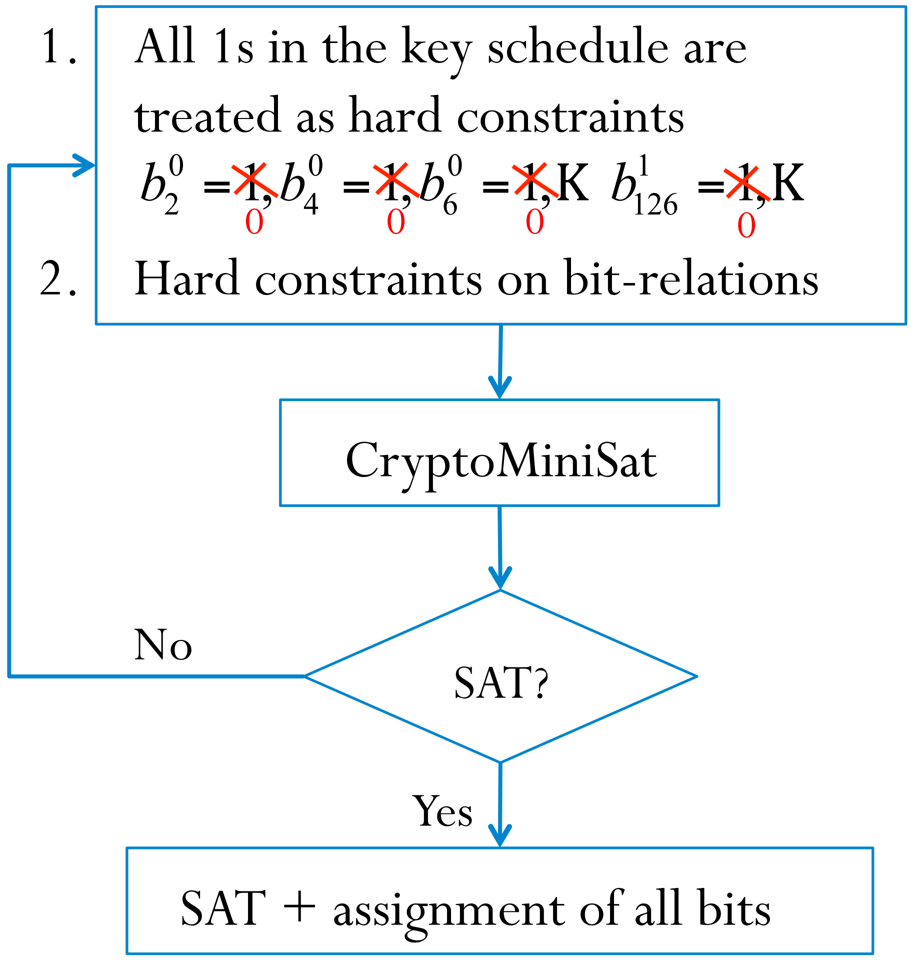


0th: 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0
0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0
1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0
0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0

1st: 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1
1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1
0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1
0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, $\underset{1}{\cancel{0}}$, 1

⋮

Flip from 0 to 1



1. All 1s in the key schedule are treated as hard constraints
$b_2^0 = \underset{0}{\cancel{1}}, b_4^0 = \underset{0}{\cancel{1}}, b_6^0 = \underset{0}{\cancel{1}}, \text{K}\ b_{126}^1 = \underset{0}{\cancel{1}}, \text{K}$
2. Hard constraints on bit-relations

Realistic assumption

- Some bits flip from 0 to 1
- Not all 1s in the key schedule are correct

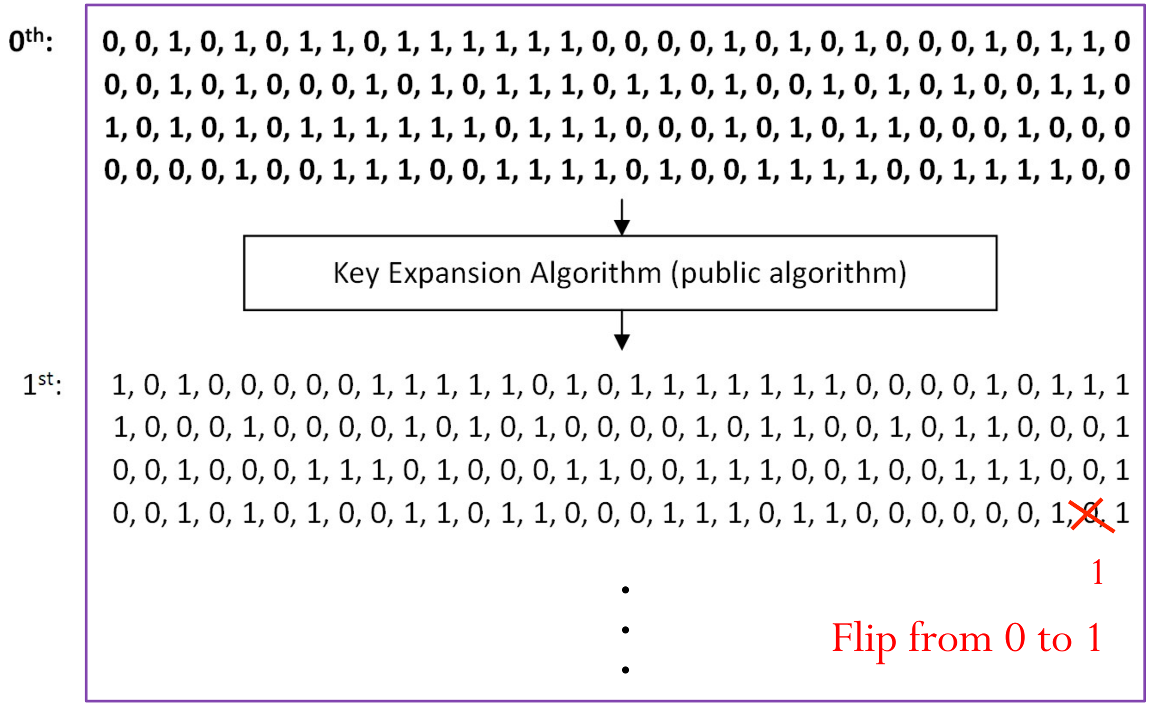
Recover AES keys using SAT solvers

- To recover a key schedule with n 1s and k reverse flipping errors
  - In the best case:
    - CryptoMiniSat needs to run $\left(\sum_{1 \le i \le k-1} C_n^i + 1 \right)$ times
  - In the worst case
    - CryptoMiniSat needs to run $\sum_{0 \le i \le k} C_n^i$ times

Recover AES keys using MaxSAT solvers

- Deal with two kinds of constraints
  - Hard constraints: must be satisfied
  - Soft constraints: may be unsatisfied
- MaxSAT solver
  - Try to satisfy all hard constraints and the maximum number of soft constraints

Recover AES keys using MaxSAT solvers



0th:

0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0
0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0
1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0
0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0

1st:

1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1
1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1
0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1
0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, ~~0~~, 1

1

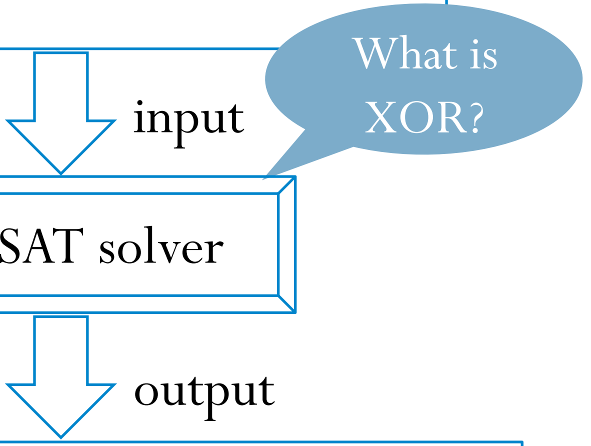
⋮

Flip from 0 to 1

Realistic assumption

- Some bits flip from 0 to 1
- Not all 1s in the key schedule are correct

1. All 1s in the key schedule are treated as *soft* constraints
   $b_2^0 = 1, b_4^0 = 1, b_6^0 = 1, \ldots \underline{b_{126}^1 = 1}, \ldots$
   unsatified
2. All bit-relations are treated as hard constraints



Maximum No. of satisfied soft constraints + assignment of all bits

Recover AES keys using MaxSAT solvers

- Convert XOR to clauses
  - Direct conversion: $B_i = B_j \oplus B_k \Rightarrow$

    $$(\neg B_i \vee \neg B_j \vee \neg B_k) \wedge$$
    $$(\neg B_i \vee B_j \vee B_k) \wedge$$
    $$(B_i \vee \neg B_j \vee B_k) \wedge$$
    $$(B_i \vee B_j \vee \neg B_k) \wedge$$

    - n variables → 2^{n-1} clauses
  - Cut-up conversion: Cut long formula into shorter ones

$$B_i = B_1 \oplus B_2 \oplus L \ \oplus B_{109} \Rightarrow \neg B_i \oplus B_1 \oplus B_2 \oplus L \ \oplus B_{109} = 1$$

$$\Rightarrow \boxed{\begin{array}{l} C_1 = \neg B_i \oplus B_1 \oplus B_2 \oplus B_3 \oplus B_4, \\ L \\ C_{22} = B_{105} \oplus B_{106} \oplus B_{107} \oplus B_{108} \oplus B_{109}, \\ \boxed{C_1 \oplus L \ \oplus C_{22} = 1} \end{array}} \Rightarrow \boxed{\begin{array}{l} D_1 = C_1 \oplus C_2 \oplus C_3 \oplus C_4 \oplus C_5, \\ L \\ D_4 = C_{16} \oplus C_{17} \oplus C_{18} \oplus C_{19} \oplus C_{20}, \\ D_1 \oplus L \ \oplus D_4 \oplus C_{21} \oplus C_{22} = 1 \end{array}}$$

$\Rightarrow$ direct conversion

Comparison

- Recover AES key schedule in the presence of reverse flipping errors

	SAT solver	MaxSAT solver
Treat 1s as	Hard constraints	Soft constraints
Need to run a solver multiple times for solving an instance?	Yes	No 🏆
Support XOR natively?	Yes 🏆	No

51,440 clauses and XOR formulas for representing bit-relations

372,240 clauses for representing bit-relations

Outline

Experiment(1/4)

- Solver
  - SAT: CryptoMiniSat
  - MaxSAT: Pwbo2.0
    - Better than Akmaxsat, WMaxSatz, QMaxSAT, QMaxSAT-g2, WPM1, PM2
- Environment
  - Core i5-2540 @ 2.6GHz / 8GB

Experiment (2/4)

Decay factor (%)	CryptoMiniSat (s)	Pwbo2.0 (s)
30	45.8	0.943
40	28.467	0.956
50	19.665	1.168
60	26.524	1.560
70	225.379	12.532
72	678.452	26.782
74	1004.161	231.610
76	1116.353	296.415

- Setting (real situation)
  - Decay factor (probability of $1 \rightarrow 0$): 30%-76%
  - Probability of flipping 0 to 1: **0.1%**
  - Number of instances for each decay factor: 100

Among100 instances, the average of No. of instances with 0, 1, 2 reverse flipping errors is 50, 36, 14, respectively

- Result
  - MaxSAT is superior to SAT approach

Experiment (3/4)

Decay factor (%)	CryptoMiniSat (s)	Pwbo2.0 (s)
30	2.045	1.037
40	1.310	1.088
50	1.971	1.345
60	5.026	2.252
70	43.532	14.945
72	47.603	28.354
74	280.825	161.740
76	480.101	384.348

- Setting
  - Decay factor (probability of 1→0): 30%-76%
  - Number of reverse flipping errors (0 →1): **1**
  - Number of instances for each decay factor: 100
- Result
  - The superiority of MaxSAT is not obvious when the number of reverse flipping errors is 1

Experiment (4/4)

Decay factor (%)	CryptoMiniSat (s)	Pwbo2.0 (s)
30	198.638	1.464
40	162.249	1.562
50	224.689	2.184
60	329.621	4.676
70	3047.821	47.725
72	4909.565	245.177
74	14715.607	2160.648

- Setting
  - Decay factor (probability of $1 \rightarrow 0$): 30%-74%
  - Number of reverse flipping errors ($0 \rightarrow 1$): **2**
  - Number of instances for each decay factor: 40
- Result
  - MaxSAT is far superior to SAT when the number of reverse flipping errors is 2

Outline

- Cold Boot Attack
- Advanced Encryption Standard (AES)
- Recover AES key Schedule
  - Using SAT solvers
  - Using MaxSAT solvers
  - Comparison
- Experiment
- Conclusion

Conclusion

- Recover AES key schedule in the presence of reverse flipping errors
  - SAT solver:
    - Treat all 1s as hard constraints
    - Run the solver repeatedly until it outputs SAT
    - CryptoMiniSat: support XOR natively
  - MaxSAT solver:
    - Treat all 1s as soft constraints
    - Solver needs to run only one time
    - Do no support XOR natively
    - Superior to the SAT approach

THE END

THANKS FOR YOUR ATTENTION