CS 613200 Advanced Logic Synthesis Homework 2 (2023, Spring)

Due Date: 2023/5/22

OBJECTIVE

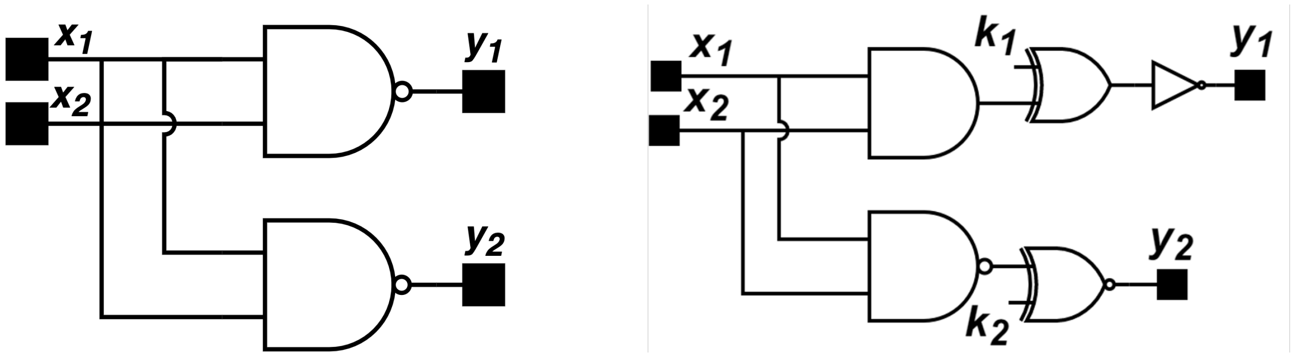
Based on the knowledge given in the following, using SAT Attack to **evaluate** different logic encryption algorithms.

Upload the report (max 4 pages) to eeclass by **2023-05-22**.

INTRODUCTION

Logic Locking

Logic locking is a technique used to protect the intellectual property of digital circuits by making the design of the circuit only usable with an authorized key. It is a form of hardware security that makes the circuit behave differently with incorrect key. The key is connected to a tamper-proof memory[1] and used to unlock the desired behavior of the circuit.



(a) Original Circuit (b) XOR-based Locked Circuit

Fig. 1 Example of logic locking

Fig. 1 demonstrates how logic locking works by using XOR and XNOR gates as **key gates** to obscure the signal. The output of the locked circuit is only correct when the values of k_1 and k_2 are set to ***01***. By using logic locking, the design of the circuit is protected and only those with the correct key can access its intended functionality.

In addition to XOR/XNOR, there are several other kinds of key gates:

[1] **Tamper proof memory:** Unreadable, ensuring attackers cannot obtain the key by reading memory.

- AND/OR [IOLTS’14]
- MUX [TCAD’12, TC’15]
- LUT [IDT’10, ASPDAC’15]

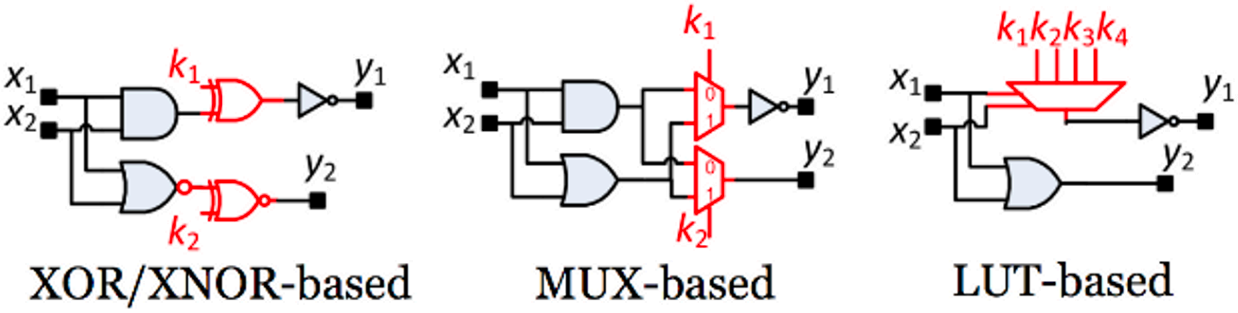


Fig. 2 Various key gate types

Early logic locking research primarily focused on preventing key values from propagating to primary outputs (PO) by ATPG tools. Consequently, the main goal of such research was to choose a suitable encryption position to prevent key values from effectively propagating to PO:

- Random [DATE’08]
- Interference analysis [DAC’12]
- Controllability [IOLTS’14]

Some studies also explore how to achieve an optimal 50% error rate for functions generated by incorrect keys:

- Fault analysis (Hamming distance) [TC’15]

Nonetheless, the SAT Attack introduced by Pramod Subramanyan in 2015 rendered previous logic locking techniques ineffective.

Attack Model

Logic locking presumes that the attacker can access the netlist of encrypted circuits and the correctly activated circuits (black box), known as the **oracle**, purchased from the market.

SAT Attack

In a SAT attack, the attacker aims to acquire the correct key.

SAT attack iteratively eliminates incorrect keys to obtain the correct one. It transforms the following two steps into a SAT problem, provides it to a SAT solver, and ultimately acquires the correct key:

1. Search for a distinguish input pattern (**DIP**) that can cause 2 different keys to generate different outputs.
2. Query oracle to obtain the correct output so that we can eliminate incorrect keys.

Fig. 3 illustrates the actual execution process of a SAT Attack. The circuit in Fig. 3 (a) is called a miter and is used to find DIPs. It consists of two identical encrypted circuits connected with a comparator. These encrypted circuits share the same input but have different key inputs. When the circuit outputs 1, it indicates a DIP has been found, where different keys produce different outputs under this input pattern.

For a SAT solver, finding a DIP is equivalent to finding a solution that makes the miter (in CNF form) satisfiable.

Fig. 3 (b) demonstrates the process of searching for the oracle. The DIP (DI_0) found in Fig. 3 (a) is input into the oracle, yielding a correct output (DO_0) as shown in Fig.3 (c). This IO pair will be added as a condition to the previous round's CNF to eliminate incorrect keys for the next round.

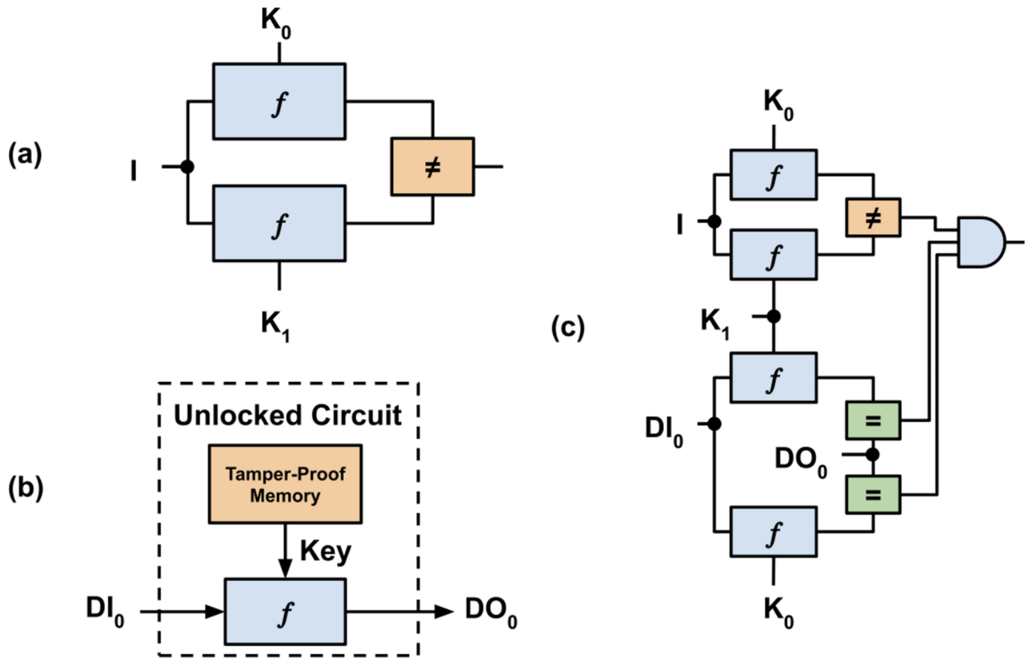
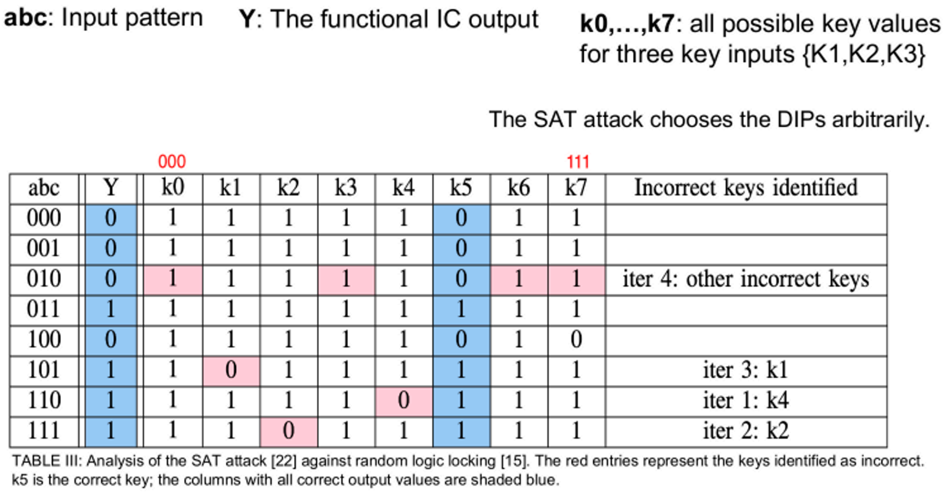


Fig. 3 Process of SAT Attack

Fig. 4 presents an example of a SAT attack on an encrypted circuit. The first column shows the circuit input, and the second column displays the correct circuit output. Note that from the attacker's perspective, the correct output can only be obtained by querying the oracle. The remaining columns demonstrate the function performance of the encrypted circuit under different keys. ***k₅*** is the correct key, as its function matches the correct key entirely.

The attack goes through five iterations:

1. In the 1st iteration, the SAT solver finds DIP: ***110***. By querying the oracle, it obtains the correct output: ***1***. IO Pair ***110:1*** is added as a constraint to the CNF, and incorrect key $\boldsymbol{k_4}$ is eliminated.
2. In the 2nd iteration, the SAT solver finds DIP: ***111***. By querying the oracle, it obtains the correct output: ***1***. IO Pair ***111:1*** is added as a constraint to the CNF, and incorrect key $\boldsymbol{k_2}$ is eliminated.
3. In the 3rd iteration, the SAT solver finds DIP: ***101***. By querying the oracle, it obtains the correct output: ***1***. IO Pair ***101:1*** is added as a constraint to the CNF, and incorrect key $\boldsymbol{k_1}$ is eliminated.
4. In the 4th iteration, the SAT solver finds DIP: ***010***. By querying the oracle, it obtains the correct output: ***0***. IO Pair ***010:0*** is added as a constraint to the CNF, and incorrect keys $\boldsymbol{k_0}$, $\boldsymbol{k_3}$, $\boldsymbol{k_6}$, and $\boldsymbol{k_7}$ are eliminated.
5. In the 5th iteration, the SAT solver can no longer find any DIPs (UNSAT), so the iteration ends. The remaining key $\boldsymbol{k_5}$ from the 4th iteration is the correct key.



abc: Input pattern **Y**: The functional IC output **k0,…,k7**: all possible key values for three key inputs {K1,K2,K3}

The SAT attack chooses the DIPs arbitrarily.

000 111

abc	Y	k0	k1	k2	k3	k4	k5	k6	k7	Incorrect keys identified
000	0	1	1	1	1	1	0	1	1	
001	0	1	1	1	1	1	0	1	1	
010	0	1	1	1	1	1	0	1	1	iter 4: other incorrect keys
011	1	1	1	1	1	1	1	1	1	
100	0	1	1	1	1	1	0	1	0	
101	1	1	0	1	1	1	1	1	1	iter 3: k1
110	1	1	1	1	1	0	1	1	1	iter 1: k4
111	1	1	1	0	1	1	1	1	1	iter 2: k2

TABLE III: Analysis of the SAT attack [22] against random logic locking [15]. The red entries represent the keys identified as incorrect. k5 is the correct key; the columns with all correct output values are shaded blue.

Fig. 4 Example of SAT attack

PROJECT DESCRIPTION

In this project, you will use SAT attack tools to decrypt locked circuits based on different algorithms. You need to deliver a report on your experimental results:

- Apply SAT attack to the locked benchmarks: **rnd**, **dac12**, **sarlock/dac12** and record the results
- The following information should be included
  - Information on CPU and Memory of the PC (or workstation)
  - For each benchmark
    - Name
    - #PIs
    - #Key Inputs
    - #POs
    - #Gates
    - #SAT iterations
    - CPU time
- You can set a time limit of 2 hours for unlocking a benchmark
- You need to tabulate your results (no restriction on format)
- You do not need to explain the results, but any opinion on them is welcome

BENCHMARK INFORMATION

Benchmark is located in the **benchmarks/** folder after unzipping the satAttackTool.zip source code.

There are eight subdirectories:

1. iolts14 (Controllability, XOR)
2. toc13mux (Fault analysis, MUX)
3. toc13xor (Fault analysis, XOR)
4. dtc10lut (Random, LUT)
5. **rnd** (Random, XOR)
6. **dac12** (Inteference analysis, XOR)
7. **sarlock** (Point Function, XOR)
8. **original**

The naming convention for these benchmarks is: _enc.bench. PCT (Percentage) is a 2 digit number which can be one of 05, 10, 25 and 50 and denotes the **percentage area overhead** of encryption for this benchmark

Note that the **sarlock** folder contains **dac12**, iolts14, and original subfolders. Files in **dac12** are encrypted using the **dac12** method and SAR-lock block, likewise for iolts14. When decrypting sarlock-related circuits, use the files in the original subfolder inside the sarlock folder as the oracle.

Benchmark Format

```
Netlist circuits in "bench" format
INPUT(PI_name)
OUTPUT(PO_name)
Gate_name = Gate_type(Fanin1, Fanin2)
…
```

Be aware that INPUT and OUTPUT positions can change and are not always at the top of the file.

TOOL INFORMATION

A. General Information

You can obtain information about tools on this website.

B. NTHU CAD Severs, using host: ic51, ic55~58

C. How to run:

1. Download the **source code** of SAT Attack Tool
2. unzip satAttackTool
3. Carefully read "**README.md**" first
4. Move to the **bin/** there are 2 binary executable file
   a. sld: Binaries of decryption tool
   b. lcmp: Binaries of verification tool
5. To use SAT attack to decryption the locked circuit, using the following command:
   ```
   $ ./sld <Locked circuit> <Original circuit>
   ```
   You can find out more about this tool with the **-h** command.
6. To verify the key, using the following command:
   ```
   $ ./lcmp <Original circuit> <Locked circuit> key=<Key value>
   ```

HINTS

About Tools,Benchmarks&Reports:

1. Evaluating the security of logic encryption algorithms

About Chatgpt:

Don't use NLP model to generate your report !!!
We will use AI Text Classifier to check.

If you have any questions, feel free to ask TA via email.
TA: 王睿杰
Email: jrw@gapp.nthu.edu.tw