ECE 443/518 – Computer Cyber Security
Lecture 26 Hardware Security

Professor Jia Wang
Department of Electrical and Computer Engineering
Illinois Institute of Technology

November 20, 2024

Outline

Reading Assignment

- This lecture: Hardware Security
- Next lecture: Side-Channel Attacks

Outline

Hardware Security

- Confidentiality
  - Program obfuscation
  - Logic Encryption
- Integrity
  - Hardware Trojan prevention and detection
- Authentication
  - Physical unclonable function
- Trusted computing base
  - Practically, to what extent can we trust the computer hardware we are using?

Outline

Program Obfuscation

- Threats: end users of your program as attackers
  - The attacker can run the program as many times as desired to observe input/output relationship.
- Defense mechanism: to implement the program in a way such that the attacker learns nothing other than the input/output relationship.
  - Not always possible, but to obfuscate certain families of functions would be useful.
- Applications: hide constant values in a program.
  - Password verification.
  - Decrypt with hidden key (digital rights management).
  - Encrypt with hidden key.

Point Function

- A boolean function $f(x)$ where x is of N bits.
- A secret s of N bits such that
  - $f(x) = 1$ for $x = s$
  - $f(x) = 0$ for $x \neq s$
- An obvious implementation of $f(x)$: $x == s$
  - XNOR each bit of x and s.
  - AND the result bits together.
  - But the attacker can easily recover s from such implementation.
- How to implement $f(x)$ so that the attacker cannot recover s?

Point Function Obfuscation

- The attacker will find s if all 2^N possible inputs are tried.
  - The actual goal of obfuscation is to prevent a computationally bounded attacker to recover s.
- Use a hash function H.
  - Compute $h = H(s)$ and implement $f(x)$ as $H(x) == h$.
- Use discrete logarithm with parameters (p, α).
  - Compute $k = \alpha^s \bmod p$ and implement $f(x)$ as $\alpha^x \bmod p == k$.
- What could the attacker learn in these two implementations?

Outline

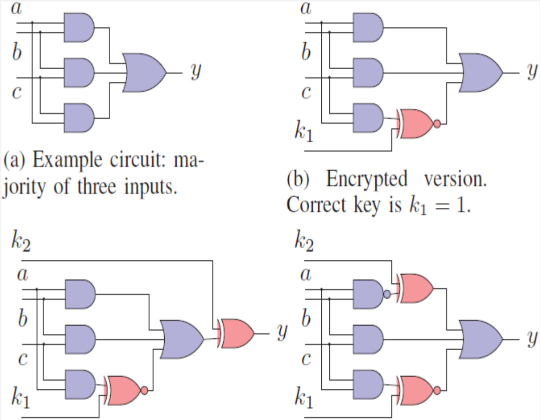
Logic Encryption

- Threats: manufacturers of your IC as attackers.
  - The attacker knows the circuit netlist of your IC and has full control of IC fabrication.
- Defense mechanism: “lock” the hardware design with a key.
  - A ROM supplies the key at runtime to “unlock” the hardware.
  - Once the hardware is manufactured, you update the ROM by yourself to include the key before sending them to end users.
  - The ROM is temper-proof so end users cannot read the key to collude with manufacturers.
- Applications: prevent unauthorized access.
  - IP protection/production control: unauthorized copy and execution
  - Program obfuscation: unauthorized reverse engineering
  - Hardware Trojan prevention: unauthorized modification

Ad-Hoc Mechanism

- Generate a random key.
- Pick up a net from the circuit netlist for each key bit.
- If the key bit is 1, replace the net with XNOR of input key bit and itself.
- If the key bit is 0, replace the net with XOR of input key bit and itself.
- Obfuscate the circuit netlist so attackers cannot tell the type of the gate the key input connect to.
  - Usually by synthesizing the circuit netlist again.

Example



(a) Example circuit: majority of three inputs.

(b) Encrypted version. Correct key is $k_1 = 1$.

(c) Another encrypted version. Correct key is $(k_1, k_2) = (1, 0)$.

(d) Yet another encrypted version with key $(k_1, k_2) = (1, 1)$.

Fig. 2: Simple logic encryption example.

(Subramanyan et al., Evaluating the Security of Logic Encryption Algorithms)

Analysis

- $g(x, k)$: the encrypted circuit netlist.
- Attackers: find k^* such that $f(x) == g(x, k^*)$ for all x.
  - Assumption: attackers know the correct input/output relationship as $f(x)$.
- Error rate: for an incorrect k, how many x are there such that $f(x)$ and $g(x, k)$ are different?
  - Some choices of x may mask incorrect k's.
  - Depend on the choice of wires and synthesis algorithm.
- A very challenge problem.
  - Achieve a proper error rate.
    - Low error rate: the attacker may simply use g and ignore errors.
    - High error rate: there are efficient algorithms solving for k^*.
  - Synthesis algorithm also need to obfuscate the circuit netlist.

Outline

Hardware Trojan

- Threats: malicious modification of IC, e.g. by manufacturers.
  - Leak sensitive information.
  - Sabotage critical computations.
- Isolation and containment can't solve the problem.
  - E.g. when the firewall is running on top of hardware with trojan.

Trojan Detection

- Physical inspection: imaging the layout and interconnects.
  - Concerns: being destructive, costly, cannot scale to large quantity of chips.
- Functional testing: detect behavioral differences.
  - Concern: not quite effective if the trojan is only activated upon very specific conditions.
- Power monitoring: detect extra power usages.
  - Concern: not quite effective if the trojan only contributes to a small fraction of power consumption.

Trojan Prevention

- Based on discussions of trojan detection, strong trojans will be those
  - Incur minimal changes to the original circuit.
  - Only activate on very specific conditions.
- Program obfuscation and logic encryption may help.
  - Both approaches make it difficult to correlate internal signals to desired functionality.
  - Trojan cannot decide when to activate, and what to leak.

Outline

Physical Unclonable Function (PUF)

- Threats: after deploying a piece of hardware, an attacker may replace it with a compromised one.
- Conceptually, this could be solved by adding identity to chips.
  - A naive approach is to store a private key into a temper-proof ROM.
  - However, the private key may leak – either when generating it, or when powerful attackers crack the ROM.
  - From another perspective, one may write the same private key to multiple chips, defeating the purpose of identification.
- Applications: secrets that no other knows
  - Smartcard based authentication.
  - Private storage.

PUF Construction

- Use unpredictable and uncontrollable physical structure.
  - Even the manufacturers have no control over the secret.
  - No two chips will have the same identity.
- Use challenge-response authentication since the secret is a unique physical structure instead of a single key.
  - The owner of the chip will need to generate and store a few challenge-response pairs before deploying the chip.
- Optical PUFs: use a transparent optical medium containing random bubbles.
  - A laser beam (challenge) shining through the medium produces a unique speckle pattern (response).
- Silicon PUFs: use transistors and interconnects.
  - An input (challenge) leads to a unique path delay (response) due to variations.

Challenges

- Environmental variations lead to different measurements even for the same chip.
- The owner should use each challenge-response pair no more than once.
  - Either the owner need to generate a lot of pairs in the beginning.
  - Or more pairs need to be generated remotely and sent back securely.

Summary

- Hardware security concerns a lot of challenge problems that we would like to research further.