Towards A Formal Treatment Of Logic Locking

Pierluigi Nuzzo, Peter Beerel, University of Southern California
Alex Malozemoff, Marios Georgiou, Ben Hamlin, Galois
Motivation: Decentralized Manufacturing

- Designer comes up with new chip design
- Sends the description to the fab
- Fab prints the chip and sends it back to the designer
- A malicious fab can
  - Overproduce for its own benefit
  - Extract intellectual property from chip (improved algorithms, ML models, etc)
  - Extract sensitive data hardcoded in the chip
  - Extract secret keys
Motivation: Decentralized Manufacturing

- Designer comes up with new chip design
- Sends the description to the fab
- Fab prints the chip and sends it back to the designer
- A malicious fab can
  - Overproduce for its own benefit
  - Extract intellectual property from chip (improved algorithms, ML models, etc)
  - Extract sensitive data hardcoded in the chip
  - Extract secret keys
Common Pitfalls and Cryptography Misuse

● Lack of Formal Threat Model
  ○ Limited set of assumed attacks; e.g., SAT attacks
  ○ Implicit or informal assumptions

● Lack of Rigorous Security Definitions
  ○ What is the adversary's task? E.g., original circuit, predicate of circuit, …
  ○ What are the adversary’s resources? E.g., description of chip, black box access to chip, …
  ○ How much (computational) power does the adversary have? E.g., polynomial time/space, unbounded, …

● Confusion with Software Obfuscation
Formalizing Logic Locking: Syntax and Correctness

What is Logic Locking? At its core:

A procedure **Lock** that on input circuit **C** produces “locked circuit” **L** and key **k**

**Correctness:**

**L** should function exactly as **C** when given **k** as input: \( L(k,x) = C(x) \)
Formalizing Logic Locking: Security - Most Prior Efforts

- Only consider SAT attacks, structural attacks, removal attacks, etc.
  - In practice, adversaries can do a lot more than SAT attacks, etc.
  - Cryptography quantifies security over well-defined classes of adversaries, and the class of "adversaries who perform SAT attacks" is not well defined.
- Don’t formally capture what adversary has access to
- Don't formally characterize adversary's goals
  - What if the adversary can recover an important part of the circuit?

**Goal**: Formalize logic locking and address these issues from past formalizations
Formalizing Logic Locking: Security - New Definitions

- **Ideally**: Adversary should “learn nothing” about $C$ from $L$ when not given $k$
- **Approach**: Consider an interactive game between an Adversary and a Challenger

A logic locking scheme is IND-LL-secure if all adversaries win with $\sim \frac{1}{2}$ probability
Formalizing Logic Locking: Security

Comparison to prior approaches:

- We do not constrain Adversary to any specific attack
- We consider a particularly strong setting: Adversary knows everything about the two circuits (it chose them), but still cannot figure out which circuit was locked
- Captures many concrete security goals: An adversary who can't distinguish also can't...
  - Use the SAT attack (or SMT, AI, or any other means) to recover the key
  - Recover the locked circuit (or even a significant part of it)
- Can be extended to security with leakage/side-channel attacks
Formalizing Logic Locking: Simulation Security

A “game based” definition is not as intuitive.

A more intuitive definition can be given using simulators.

Idea: Imagine an entity (the "simulator") that does not have access to the locked circuit

- If we can show that the adversary (with access to the locked circuit) cannot "learn more" than the simulator, the scheme is secure
- Why? Because the simulator doesn't even have the locked circuit!

"Simulation security" is another common approach towards defining security
Formalizing Logic Locking: Simulation Security

Anything that can be computed by the adversary given $L$ and oracle access to $C$, can also be computed by a "simulator" given only oracle access to $C$ (and in particular, no $L$)

Logic locking is SIM-LL secure if $b$ and $b'$ have (nearly) equal distributions
IND-LL implies SIM-LL

Proof Sketch.

By IND-LL, the Adversary cannot distinguish between $L$ and $L'$.

**Simulator:**
1. Pick arbitrary $C'$
2. $L', k = \text{Lock}(C')$

By IND-LL, the Adversary cannot distinguish between $L$ and $L'$. 
SIM-LL does not imply IND-LL

Proof Sketch.

Given SIM-LL scheme \( \text{Lock}(C) \), we create \( \text{Lock}' \):

1. \((L, k) \leftarrow \text{Lock}(C)\)
2. \(L' = (L, C(0))\)
3. Output \((L', k)\)

- **Claim 1.** \( \text{Lock}' \) is also SIM-LL
  - \( C(0) \) can be learned by a Simulator by querying \( C \)
- **Claim 2.** \( \text{Lock}' \) is *not* IND-LL
  - The adversary can pick \( C_0, C_1 \) s.t. \( C_0(0) \neq C_1(0) \)
Functional Secrecy (implicit in previous works)

The adversary and the simulator have to guess the whole circuit (not just 1 bit).

A Logic Locking Scheme is FS-secure if $\Pr[C_{\text{adv}} = C] \approx \Pr[C_{\text{sim}} = C]$

**Thm:** SIM-LL implies FS
Function Recovery [CS21]

FR considers **unlearnable** circuits. Consider a set of circuits $\mathcal{C}$.

$\mathcal{C}$ is **unlearnable** if the best adversary wins with $\sim 1/|\mathcal{C}|$ probability.

**Diagram:**
- Adversary
- Challenger

Choose $\mathcal{C}$ at random from $\mathcal{C}$

$x_1$ → $C(x_1)$ → $\vdots$ → $x_n$ → $C(x_n)$

Check if $C' = C$
A logic locking scheme is FR-secure if the best adversary wins with $\sim 1/|\mathcal{C}|$ probability.
FS implies FR for unlearnable $\mathcal{C}$

Proof Sketch.

If a logic locking scheme is FS but not FR then is $\mathcal{C}$ learnable:

1. An FR adversary $A_{FR}^{\mathcal{C}}(L)$ can guess $\mathcal{C}$
2. Let $A_{FS} = A_{FR}$ be an FS adversary
   a. $A_{FS}^{\mathcal{C}}(L)$ can also guess $\mathcal{C}$
3. By FS, there is a simulator $S_{FS}$ such that $S_{FS}^{\mathcal{C}}$ can also guess $\mathcal{C}$
4. Therefore, $S_{FS}$ can learn $\mathcal{C}$
More Relations

- IND-LL and SIM-LL imply CFS
  - Schemes secure against these retain security in a "compositional" setting
- Prior notions do not imply CFS
  - Schemes secure against these break down in a "compositional" setting!
Universal Circuits

A universal circuit $\mathbf{UC}_n$ can evaluate any circuit of size $n$:

$$\mathbf{UC}_n([C], x) = C(x),$$

where $[C]^*$ is the description of $C$.

The (input) size of $\mathbf{UC}_n$ has to be at least $\Omega(n \log n)$ since we need at least $n \log n$ bits to describe a circuit of size $n$.

* we often abuse notation and write $C$ for the description too.
Universal Circuits are IND-LL secure

\[ \text{Lock}(C) = (L,k) \text{ where:} \]

- \( L = UC \)
- \( k = C \)

\text{Lock} is correct: \( L(k,x) = UC(C,x) = C(x) \)

\text{Lock} is IND-LL with \textbf{perfect} security:

- \( L_0 \equiv UC \equiv L_1 \) where \( (L_b,k) \leftarrow \text{Lock}(C_b) \)

Construction mimics FPGA [MGM⁺22]
Open Problems

● Universal Circuits are expensive
  ○ Can we trade perfect security for more efficient constructions?
  ○ **Pseudo-UC**: Circuit that can evaluate only a small number of circuits
    ■ Versus UC, which can evaluate all circuits
  ○ **Goal**: Hard for adversary to learn which circuits can be evaluated
  ○ Succinct “hiding” of a circuit’s **topology** is sufficient

● Current work focuses on combinational circuits. Next steps:
  ○ Develop definitions for latch locking
  ○ Develop definitions for sequential circuits