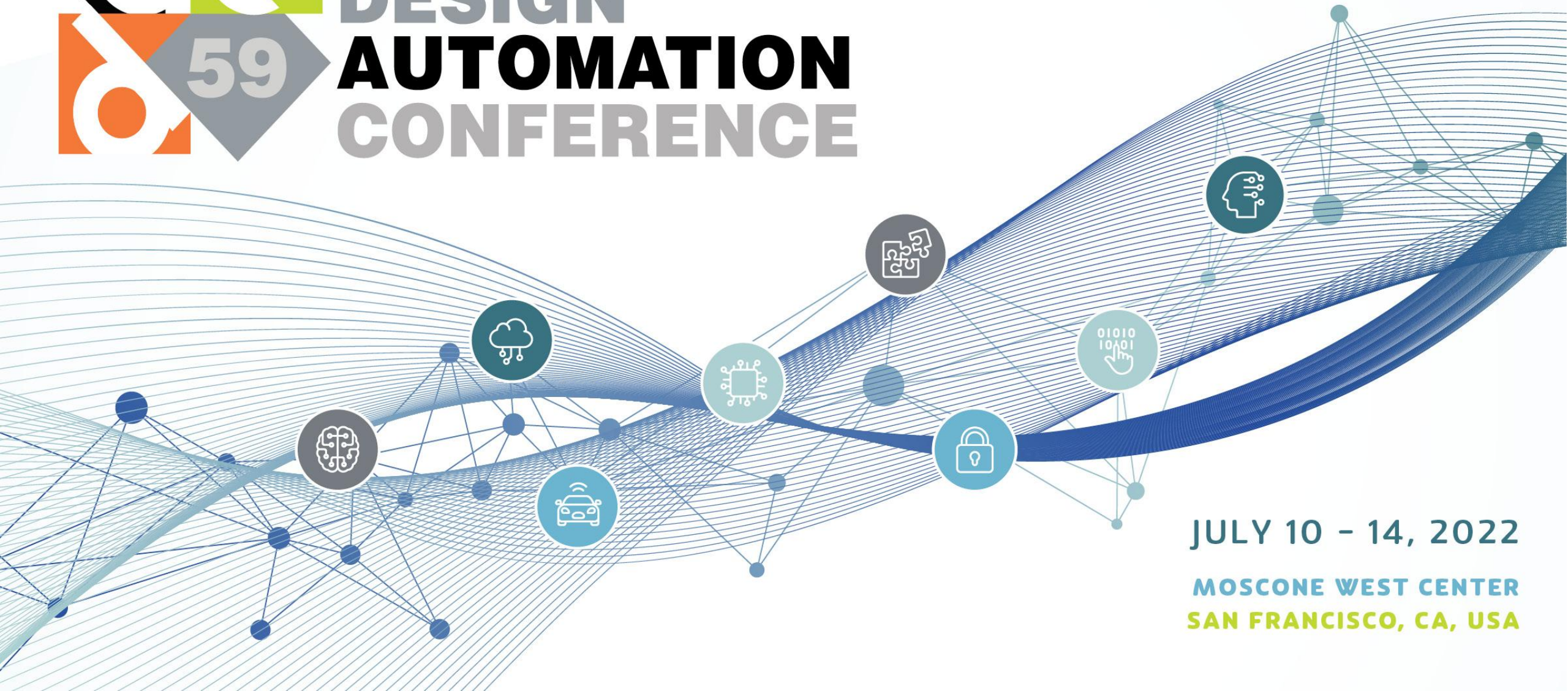
59 DESIGN AUTOMATION CONFERENCE



JULY 10 - 14, 2022

MOSCONE WEST CENTER
SAN FRANCISCO, CA, USA

天津大学 Tianjin University

UF UNIVERSITY of FLORIDA

PathFinder: Side Channel Protection through Automatic Leaky Paths Identification and Obfuscation

Haocheng Ma[1], Qizhi Zhang[1], Ya Gao[1], Jiaji He[1], Yiqiang Zhao[1] and Yier Jin[2]

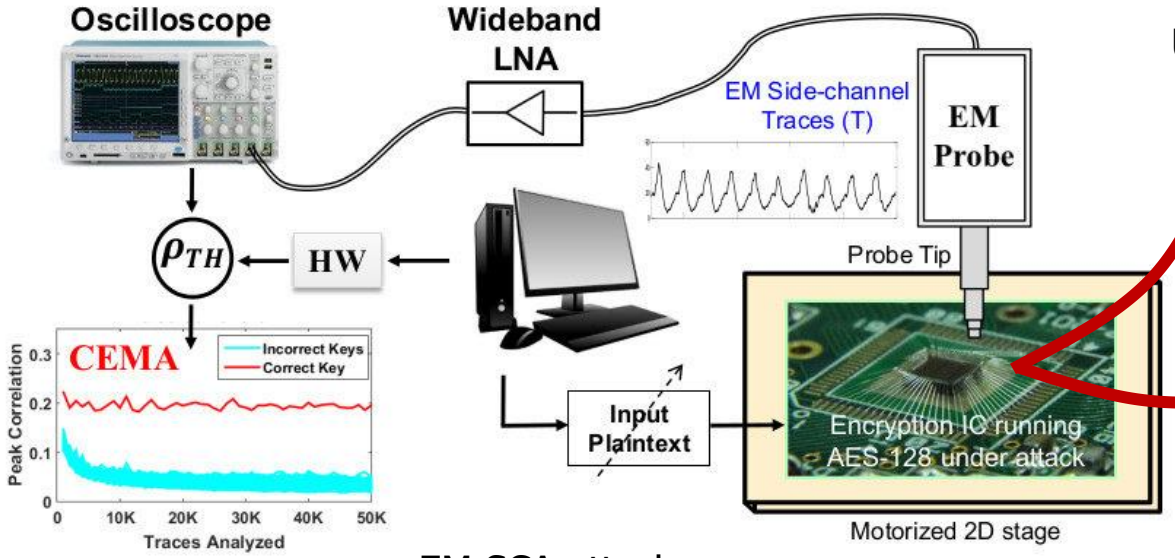
[1]Tianjin University, [2]University of Florida,

Outline

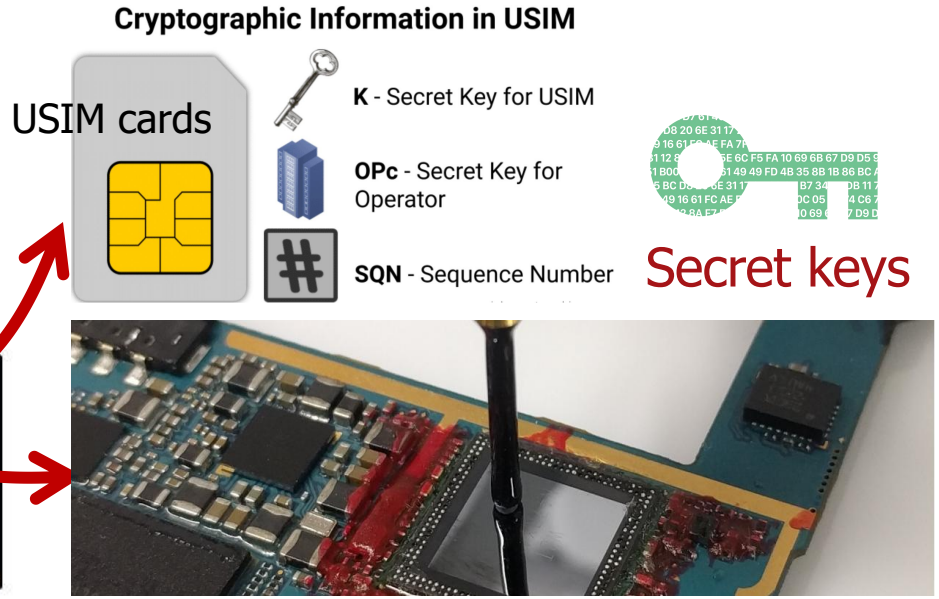
- Motivation
- Framework
  - Path Identification
  - Path Obfuscation
- Case Studies
  - CPA Results
  - CEMA Results
- Conclusions

Motivation

- Side-channel attacks on cryptographic ICs
  - Timing, Power, Electromagnetic (EM), etc.
  - Break USIM Cards[1], Secure Cryptoprocessor[2], etc.



EM SCA attacks

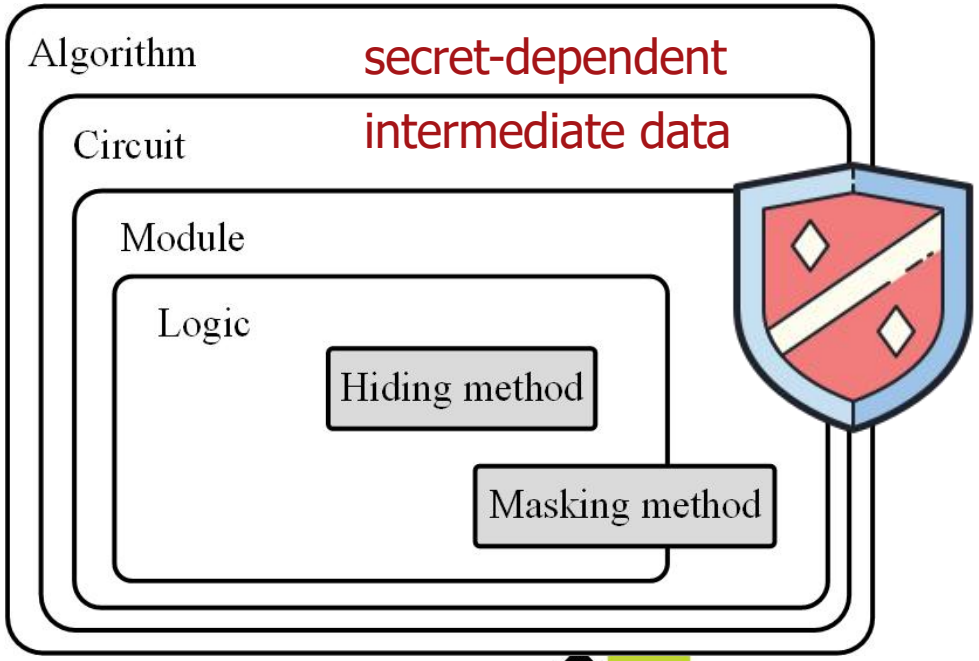


Hardware cryptoprocessor of a smartphone

[1] Liu J, et al. Small tweaks do not help: Differential power analysis of milenage implementations in 3G/4G USIM cards, 2015.
[2] Vasselle A, et al. Breaking mobile firmware encryption through near-field side-channel analysis. 2019.

Motivation

- Countermeasures to defend SCA attacks
  - Underlying concepts include hiding and masking
  - Breach or isolate the correlation between sensitive variables and side channel information
  - Different hardware hierarchies
    - whole design or submodules[1-3]
    - secure logic styles[4]



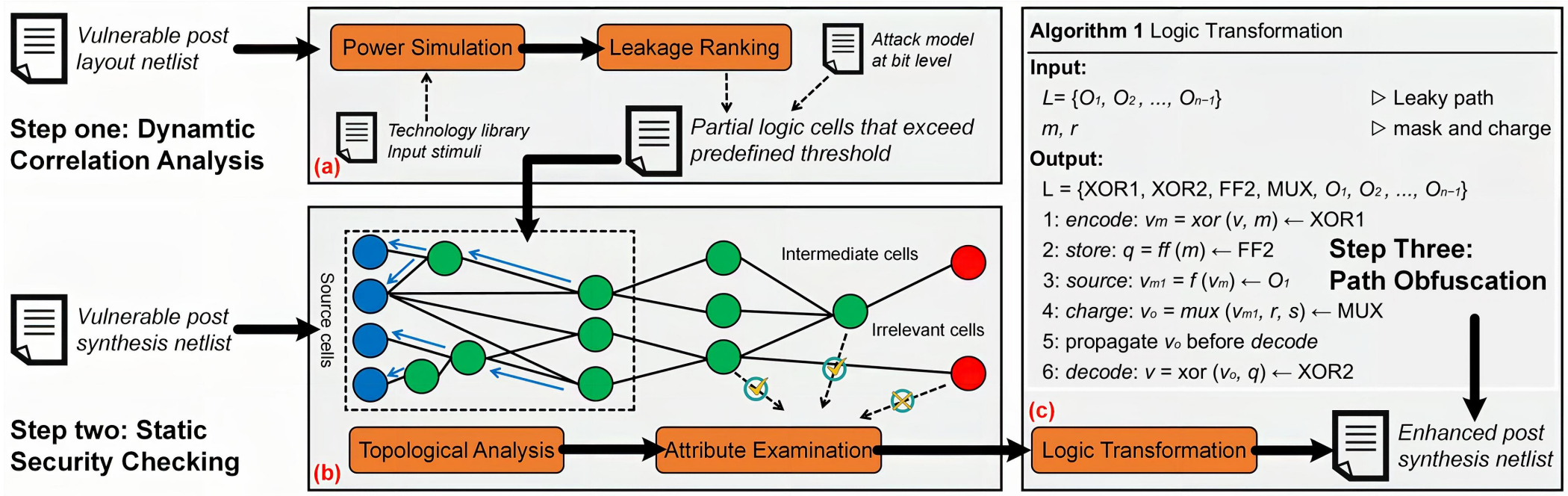
[1] noise injection
[2] randomize switching behaviors
[3] integrated voltage regulators
[4] wave dynamic differential logic

Motivation

- Main challenges of existing countermeasures
  - Significant power, area and speed overhead
  - Proper implementations require
    - non-traditional EDA toolchains
    - full-custom circuit design
    - manual optimizations
- Our contribution
  - PathFinder to support automated side-channel protection
  - Identify sources of leakage, e.g., leaky paths
  - Apply specific protections to obfuscate leaky paths

Framework

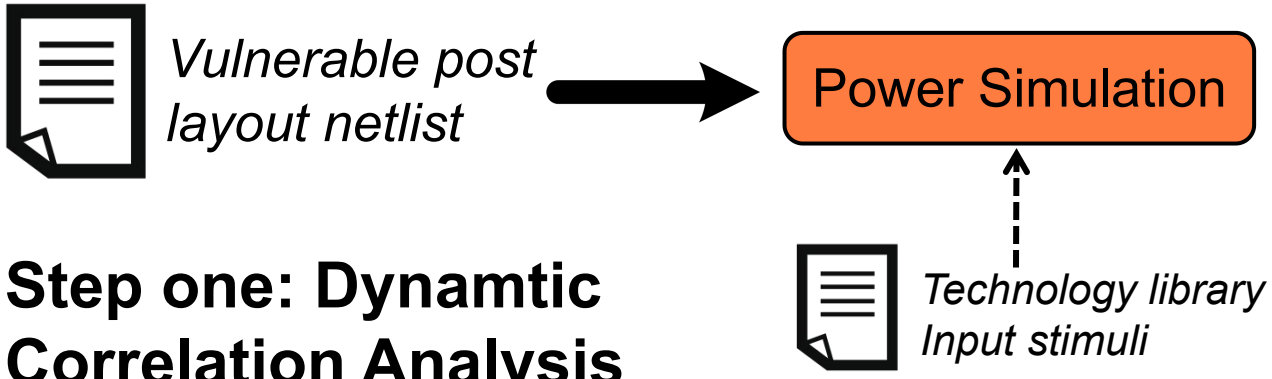
- Overall workflow of PathFinder
  - Dynamic correlation analysis, static security checking
  - Path obfuscation



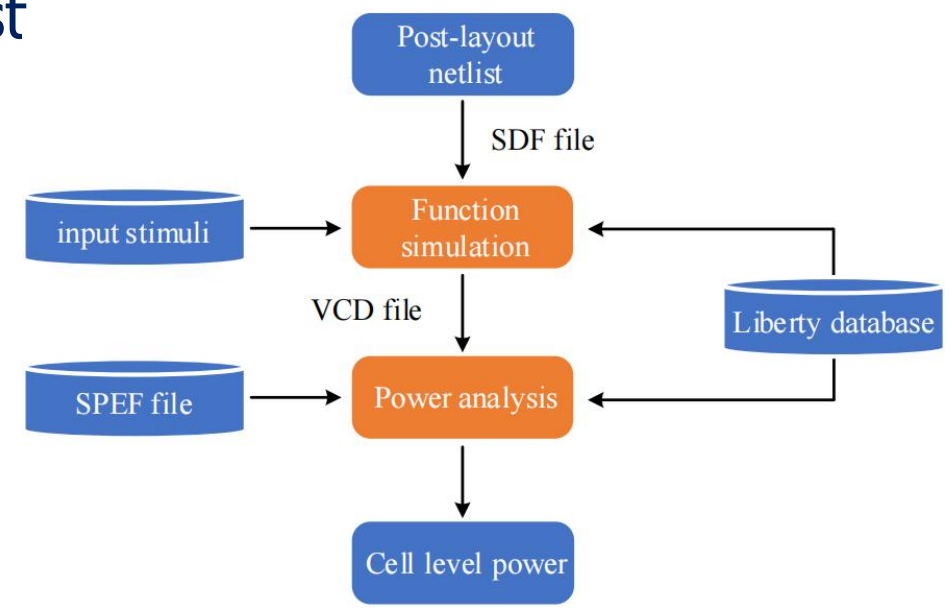
The overall framework of PathFinder

Framework

- Dynamic correlation analysis
  - Power simulation collects the dynamic power of each logic cell
    - function simulation to record switching activities
    - power analysis on post-layout netlist



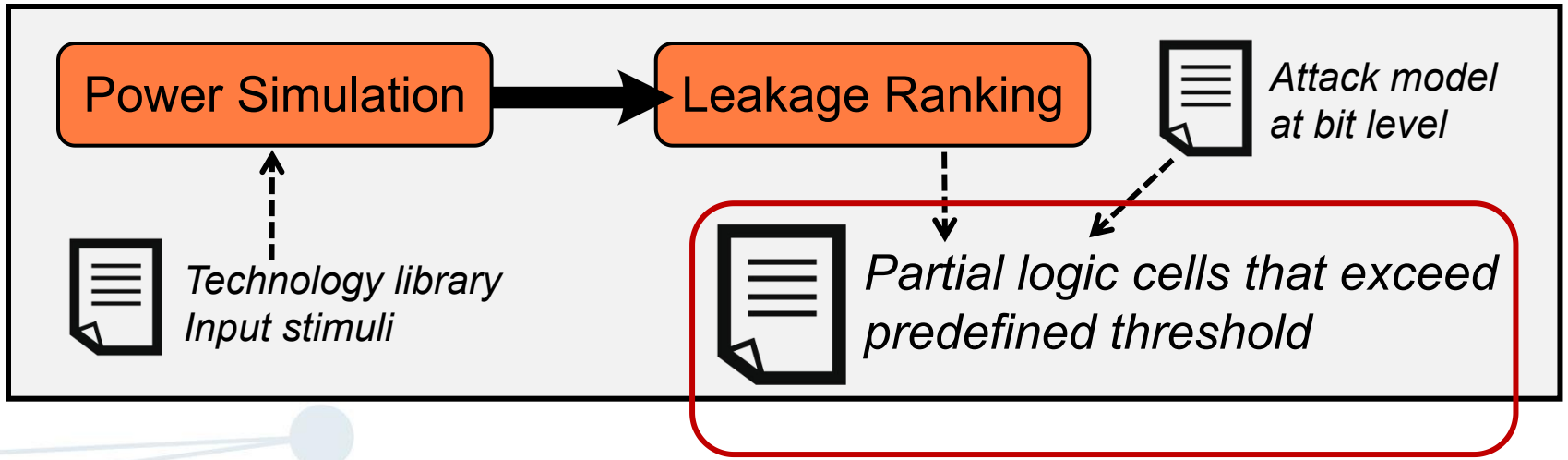
Step one: Dynamtic Correlation Analysis



Detailed flow of power analysis

Framework

- Dynamic correlation analysis
  - Leakage ranking quantifies the information leakage of each logic cell
    - maximum Pearson correlation serves as the leakage criterion
  - Ranks logic cells from highest leakage criterion to lowest
    - partial logic cells that exceed the predefined threshold

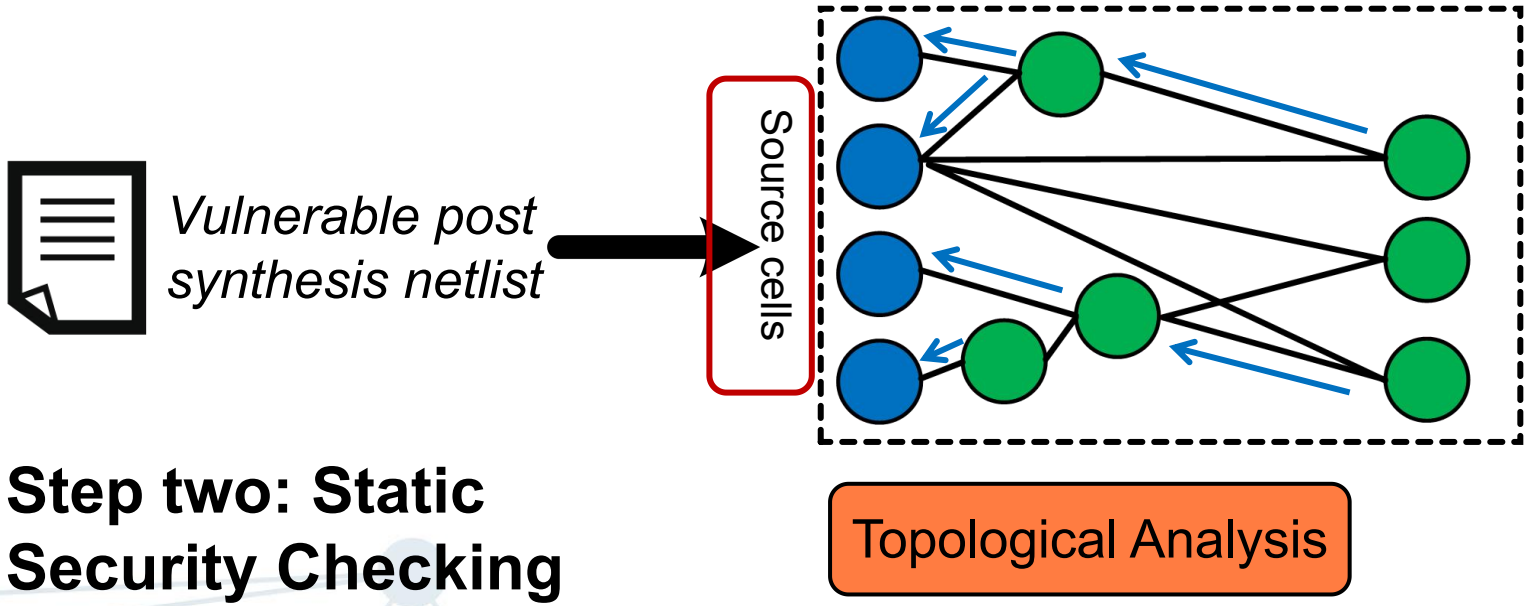


$$C = max(|\rho(P, L)|)$$

P: power traces
L: leakage model

Framework

- Static security checking
  - Topological analysis locates source cells in partial logic cells
    - CELL and NET describe the intrinsic connectivity
    - inference head of antecedent networks within the bound



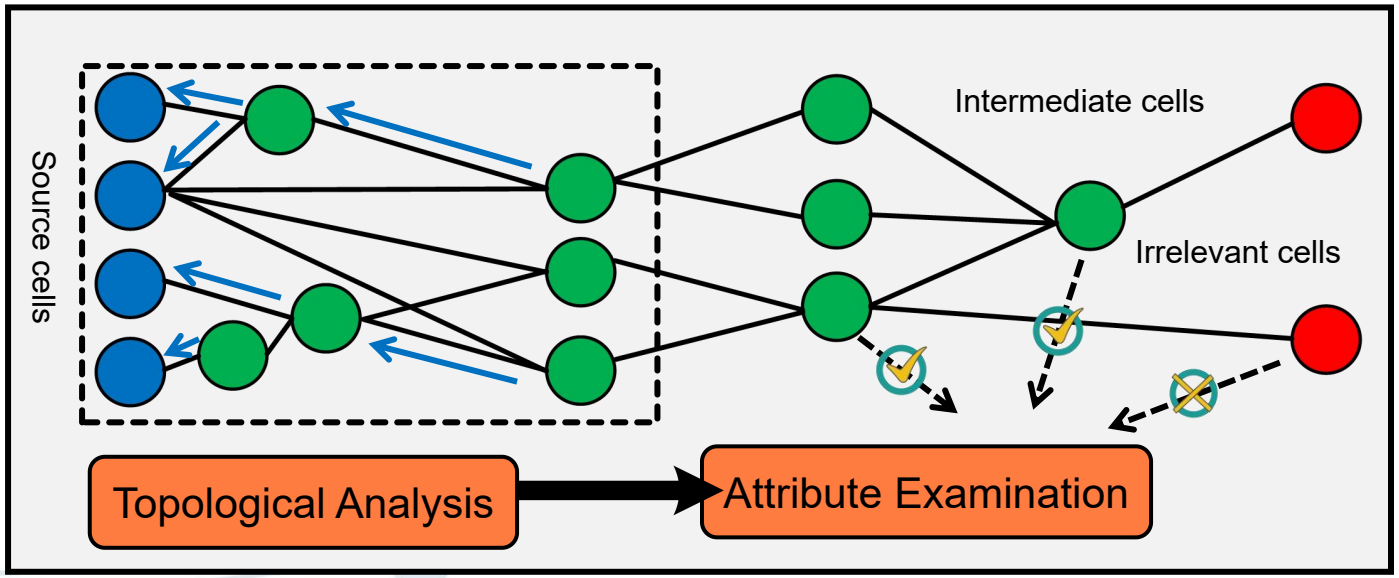
CELL lists signal nets passing in and out per logic cell

NET records logic cells that connect before and after per signal net

Step two: Static Security Checking

Framework

- Static security checking
  - Attribute examination constructs consequent leaky paths
    - similar power profile as the known source cell or intermediate cell
    - check whether logic cells inherit leakage attributes



$$\forall(a_1, a_2, \ldots a_{n-1}), \left|O_c - O_p\right| = \left|v_c - v_p\right|$$

- spread state transitions of sensitive variables to its output net
- no matter how other input nets change

sensitive variable: vc → vp, other input: a1, a2, ..., an-1
output state: Oc → Op
Here we neglect the slight divergence between power profile of state transition 0 → 1 (0 → 0) and 1 → 0 (1 → 1)

Framework

- Path obfuscation
  - Logic transformation translates protections on post-synthesis netlist
  - Boolean masking and random precharge

Algorithm 1 Logic Transformation

Input:

$L = \{O_1, O_2, ..., O_{n-1}\}$ ▷ Leaky path

m, r ▷ mask and charge

Output:

L = {XOR1, XOR2, FF2, MUX, $O_1, O_2, ..., O_{n-1}$}

1: *encode*: $v_m = xor\ (v, m)$ ← XOR1

2: *store*: $q = ff\ (m)$ ← FF2

3: *source*: $v_{m1} = f\ (v_m)$ ← O_1

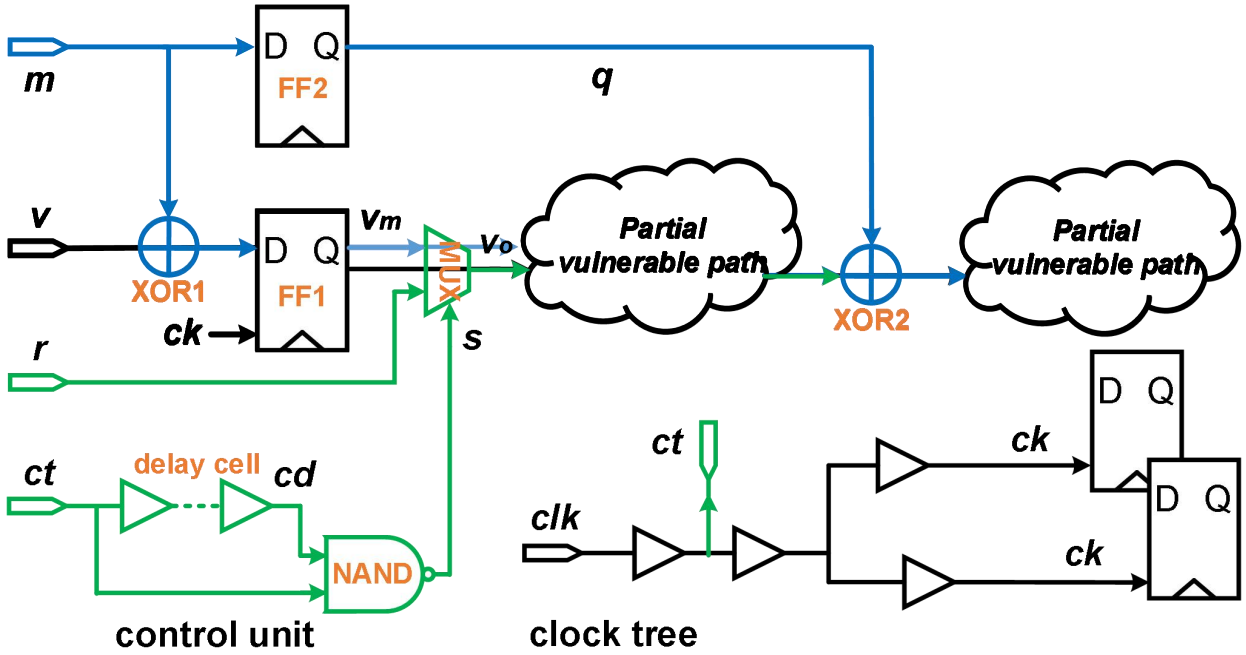
4: *charge*: $v_o = mux\ (v_{m1}, r, s)$ ← MUX

5: propagate v_o before *decode*

6: *decode*: $v = xor\ (v_o, q)$ ← XOR2

Step Three: Path Obfuscation

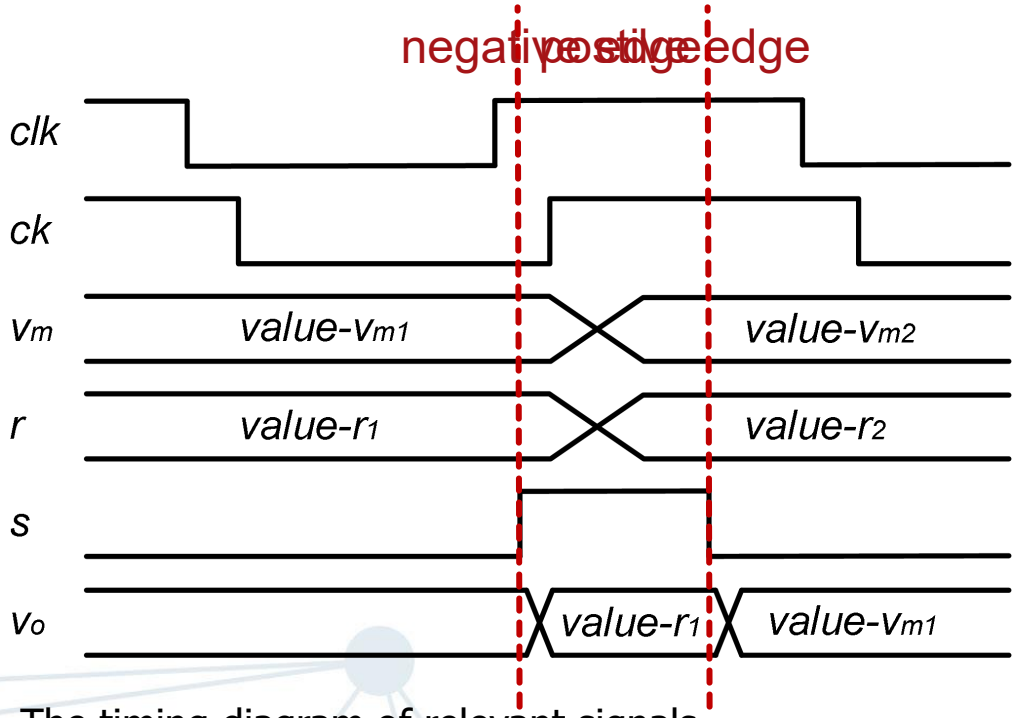
Logic Transformation → *Enhanced post synthesis netlist*



Hardware scheme of protection solutions
Boolean masking (blue)
random precharge (green).

Framework

- Path obfuscation
  - the control signal and delay cells have a strong impact on both security and performance

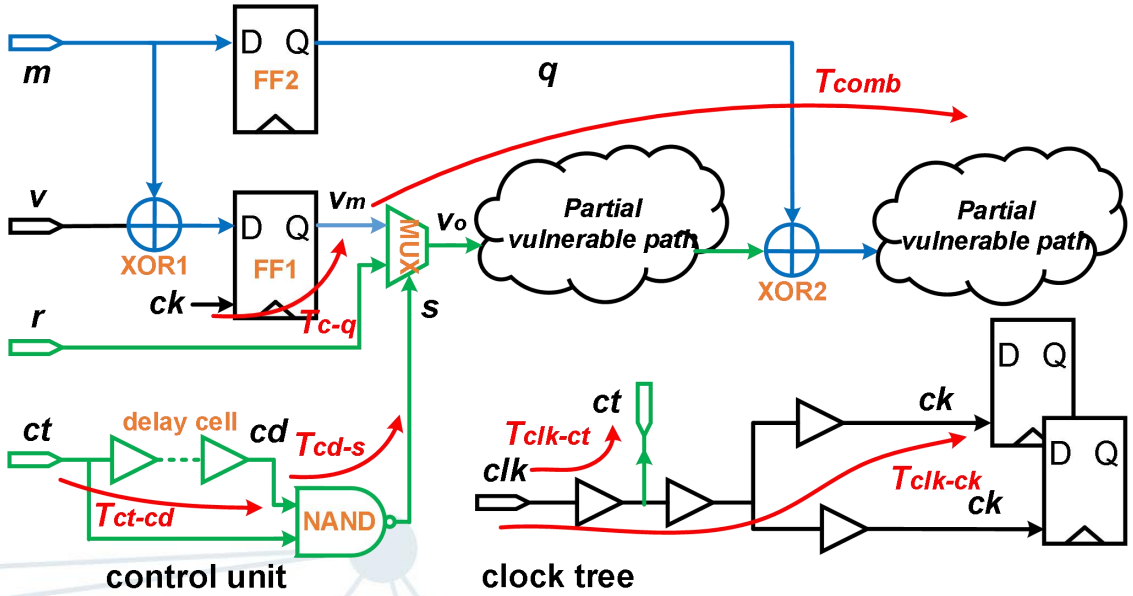


The timing diagram of relevant signals

- random charges will deliver to the next level FFs, which results in wrong results (earlier)
- random charges do not take effect on partial leaky paths when the positive edge arrives later (later)
- sampling errors occur when transferring values of source cells (earlier)
- setup time violations due to extra delay (later)

Framework

- Path obfuscation
  - the control signal and delay cells have a strong impact on both security and performance
  - formulate the timing demand to the placement and routing stage



$$T_{clk-ct} + T_{comb} + T_{cd-s} > T_{clk-ck} + T_{c-q} + T_{hold} \quad (3)$$

$$T_{clk-ct} + T_{cd-s} < T_{clk-ck} + T_{c-q} \quad (4)$$

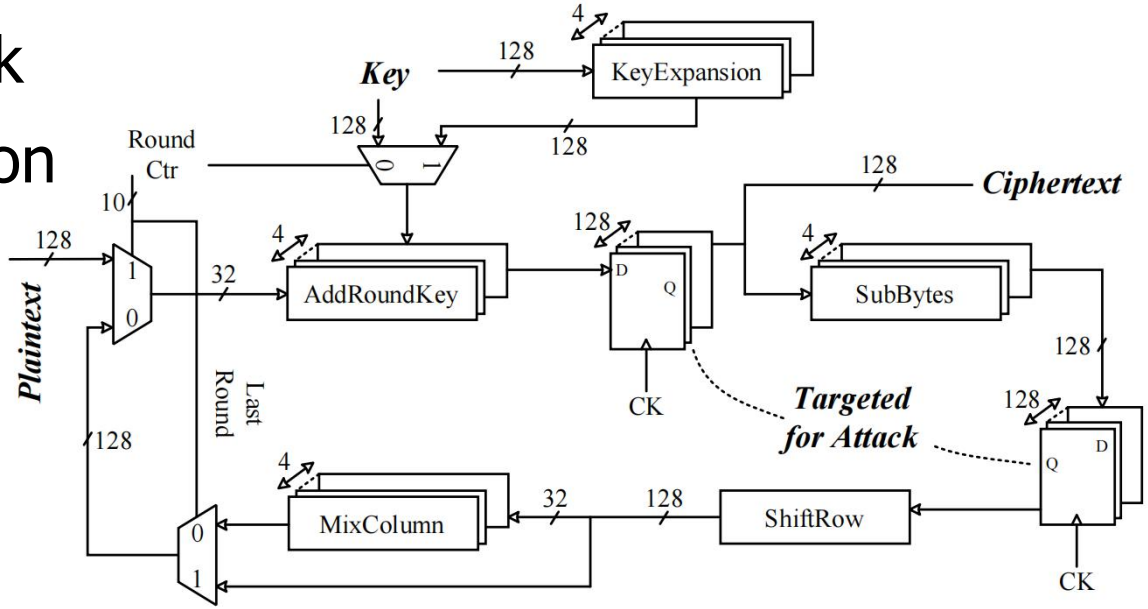
$$T_{clk-ct} + T_{ct-cd} + T_{cd-s} > T_{clk-ck} + T_{c-q} \quad (5)$$

$$T_{clk-ct} + T_{ct-cd} + T_{cd-s} + T_{comb} < T_{clk-ck} + T_{cycle} - T_{setup} \quad (6)$$

T_{cycle}, T_{setup} and T_{hold} denote the clock period, setup time and hold time
Types of T_{a-b} denote the delay from signal a to signal b
T_{comb} is the total delay of combinational logic behind FF1

Case Studies

- Benchmark
  - 128-bit AES design
  - RS-232 serial communication block
  - RTL-to-GDS flow for implementation
    - 180 nm CMOS technology
    - 10 modules
    - 10083 logic cells
    - supply voltage 1.8 V
    - clock frequency 25 MHz



Hardware architecture of the 128-bit AES design

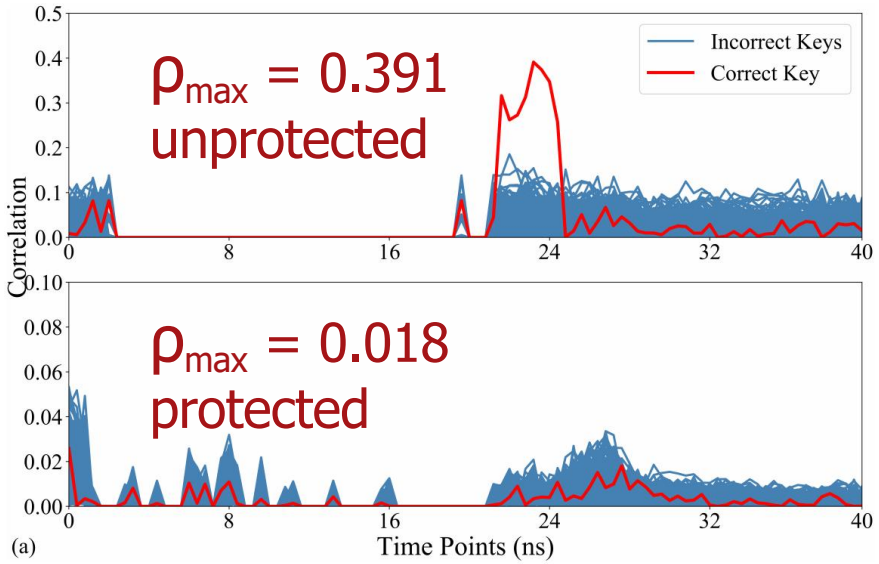
Case Studies

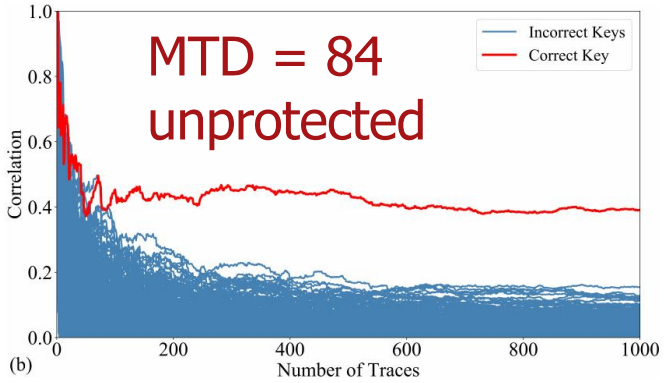
- Experiment Configuration
  - PathFinder Parameters
    - 1000 input stimuli
    - leakage criterion > 0.95
    - 32 random masks and random charges
  - PathFinder Results
    - selects 1082 partial logic cells (18.54 min)
    - 2120 logic cells compose complete leaky paths (9.12 s)
    - increase 659 cells for protection (0.13 s)

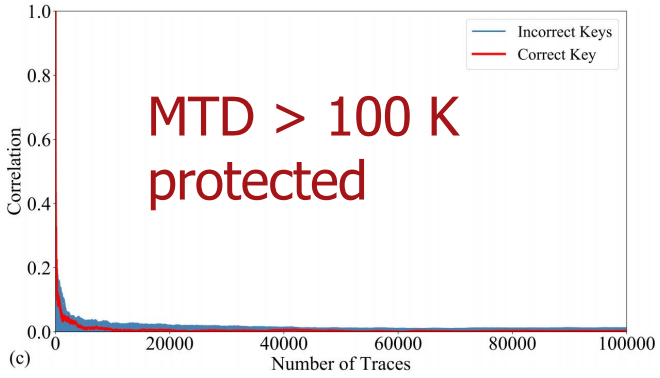
Case Studies

- CPA Attack
  - Simulated power trace using the Primetime PX
  - Total 100 K dynamic traces are collected
  - 1190× security inprovements
  - Overheads
    - 6.53 % area
    - 4.51 % power
    - 3.1 % performance

ρ_{max} : maximum Pearson correlation coefficient
MTD the minimum traces to disclosure the key



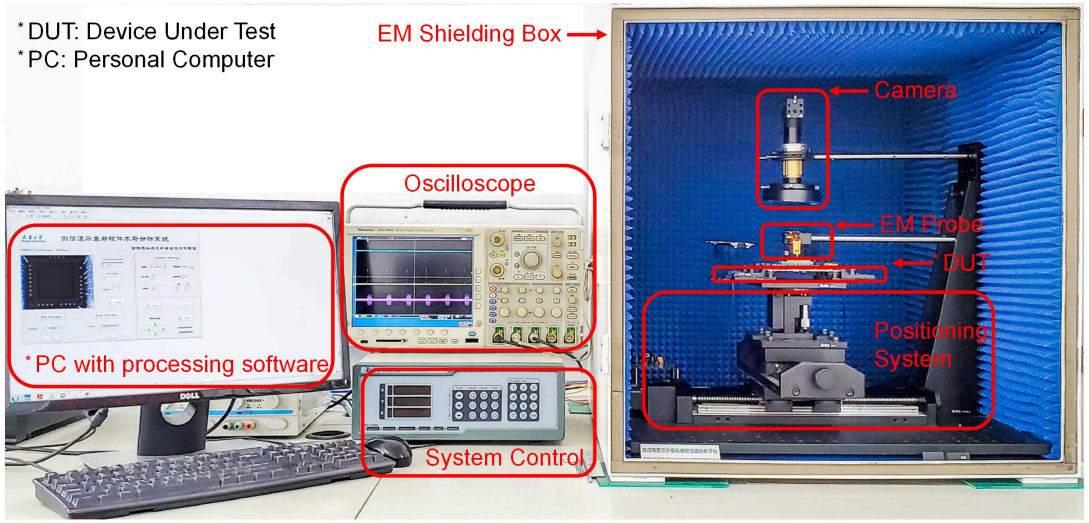




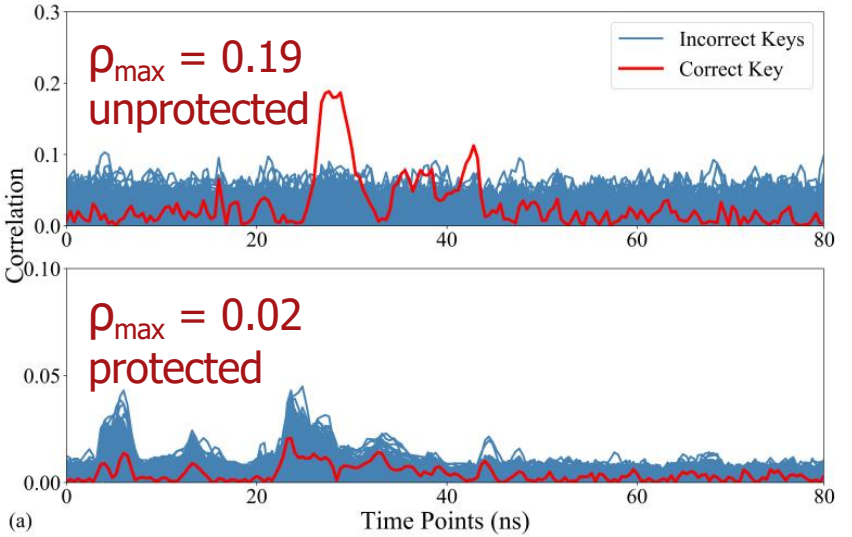
Comparison of CPA attack results between unprotected and protected designs

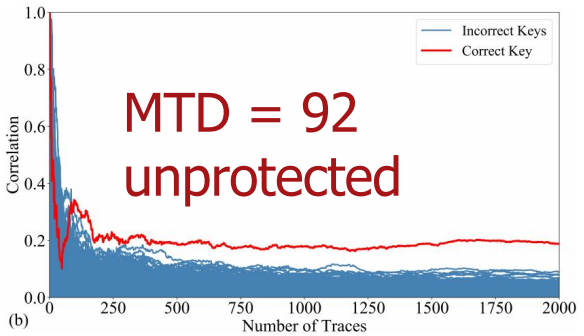
Case Studies

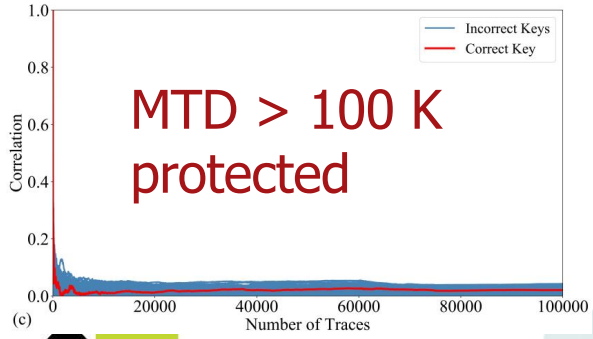
- CEMA Attack
  - Actual EM trace collected by EM probe
  - Total 100 K EM traces are collected
  - 1085× security inprovements



Measurement setup







CEMA attack results

Conclusion

- We propose PathFinder tool for automatic side-channel protection
- This tool identifies leaky paths by combining dynamic and static procedures
- Well-designed hardware solutions such as Boolean masking and random precharge are inserted to protect the design
- Enhance the side-channel resistance by at least 1000×
- Introduce slight impacts on the area, power and performance.

TABLE II: Comparison with existing works.

Works	MTD Improv.		Overheads		
	Power	EM	Area	Power	Perf.
Moradi [2]	100×	–	359 %	262 %	40^{a}
Moradi [10]	10000×	–	196 %	–	20^{a}
Yao [4]	$4\times^{b}$	–	10 %	–	–
SLPSK [11]	107×	–	0 %	0 %	0 %
KF [3]	16×	–	31.9 %	–	31.25 %
Singh [8]	4210×	136×	96.7 %^{c}	32 %	10.4 %
Das [12]	–	167×	23 %	49 %	0 %
Das [9]	125000×	83333×	36.7 %	49.8 %	0 %
This Work	$1190\times^{d}$	$1085\times^{d}$	6.53 %	4.51 %	3.1 %

- Data has not been reported,

a Increase clock cycles,

b Decrease the maximum correlation,

c Area overhead includes 1.9 nF load capacitor,

d Only 100 K traces are collected limited by experiment conditions.

Thank You!
Any Questions?