Thwarting Cache Side-Channel Attacks Through Dynamic Software Diversity

Stephen Crane, Andrei Homescu, Stefan Brunthaler, Per Larsen, Michael Franz

University of California, Irvine

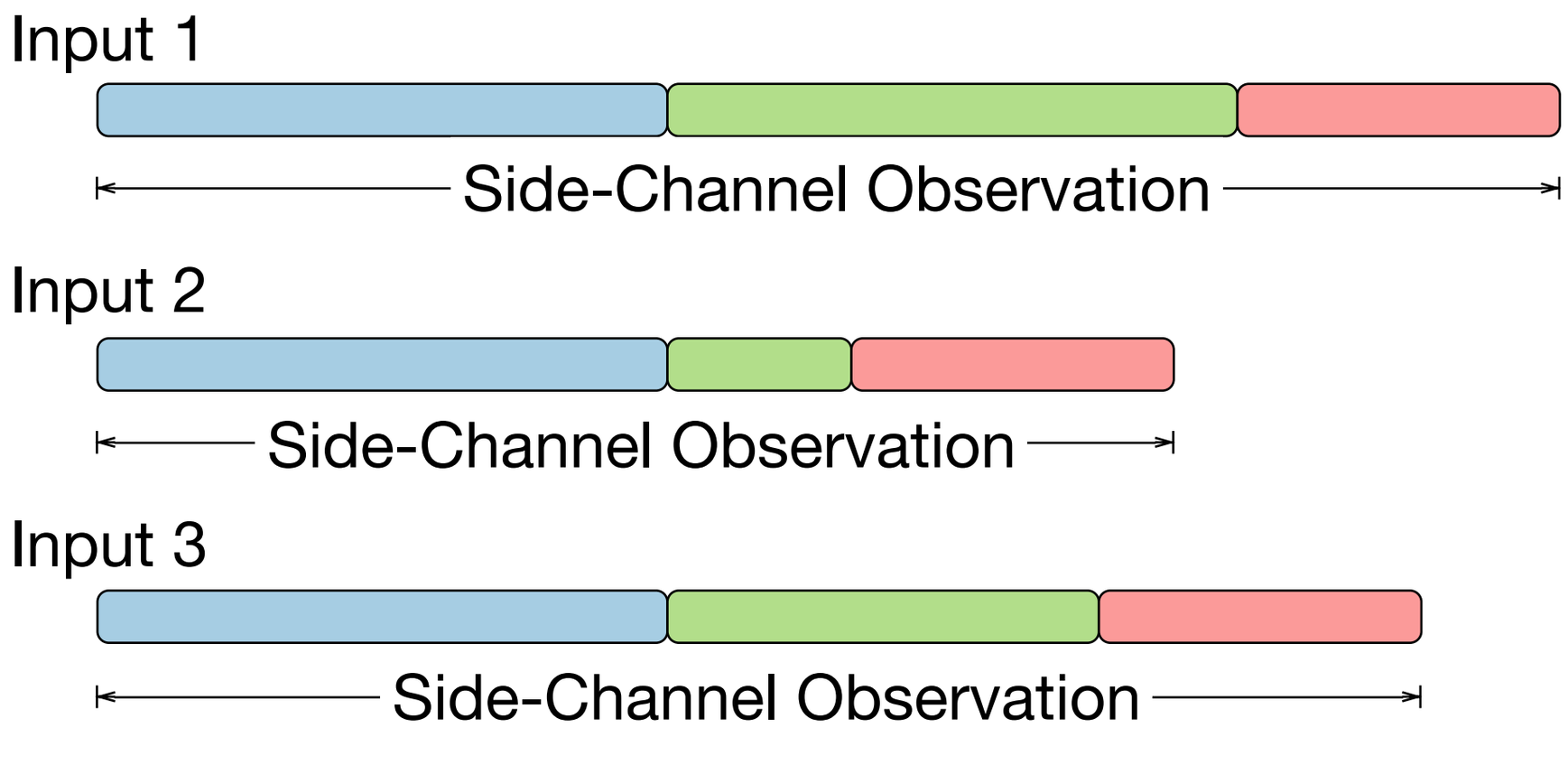
NDSS 2015

Side-Channel Attacks

THE PROBLEM

The attacker:

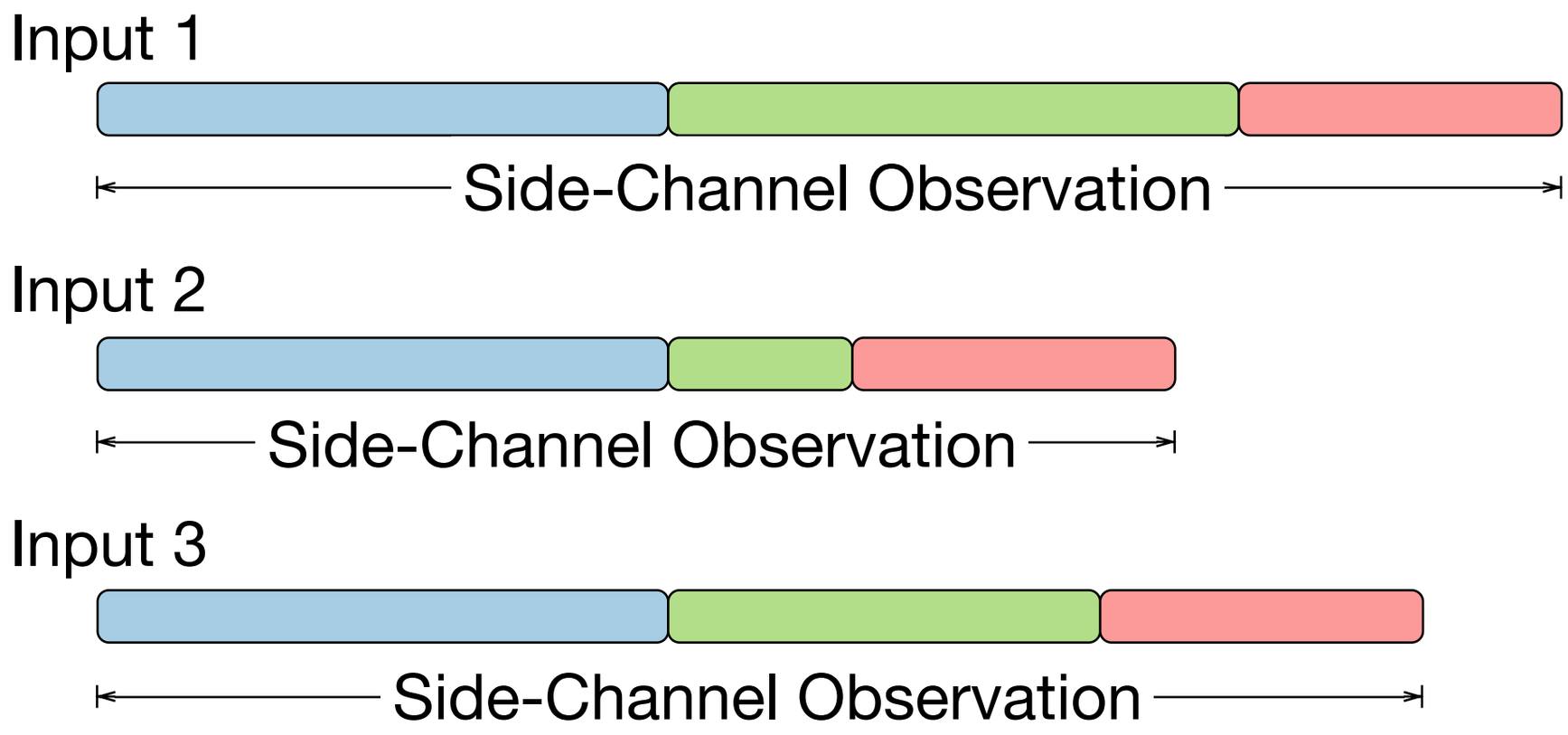
- Observes dynamic side-effects of computation
  - timing, cache footprint, power consumption, …
- Derives secret information from side-channel observations



Side-Channel Attacks

THE PROBLEM

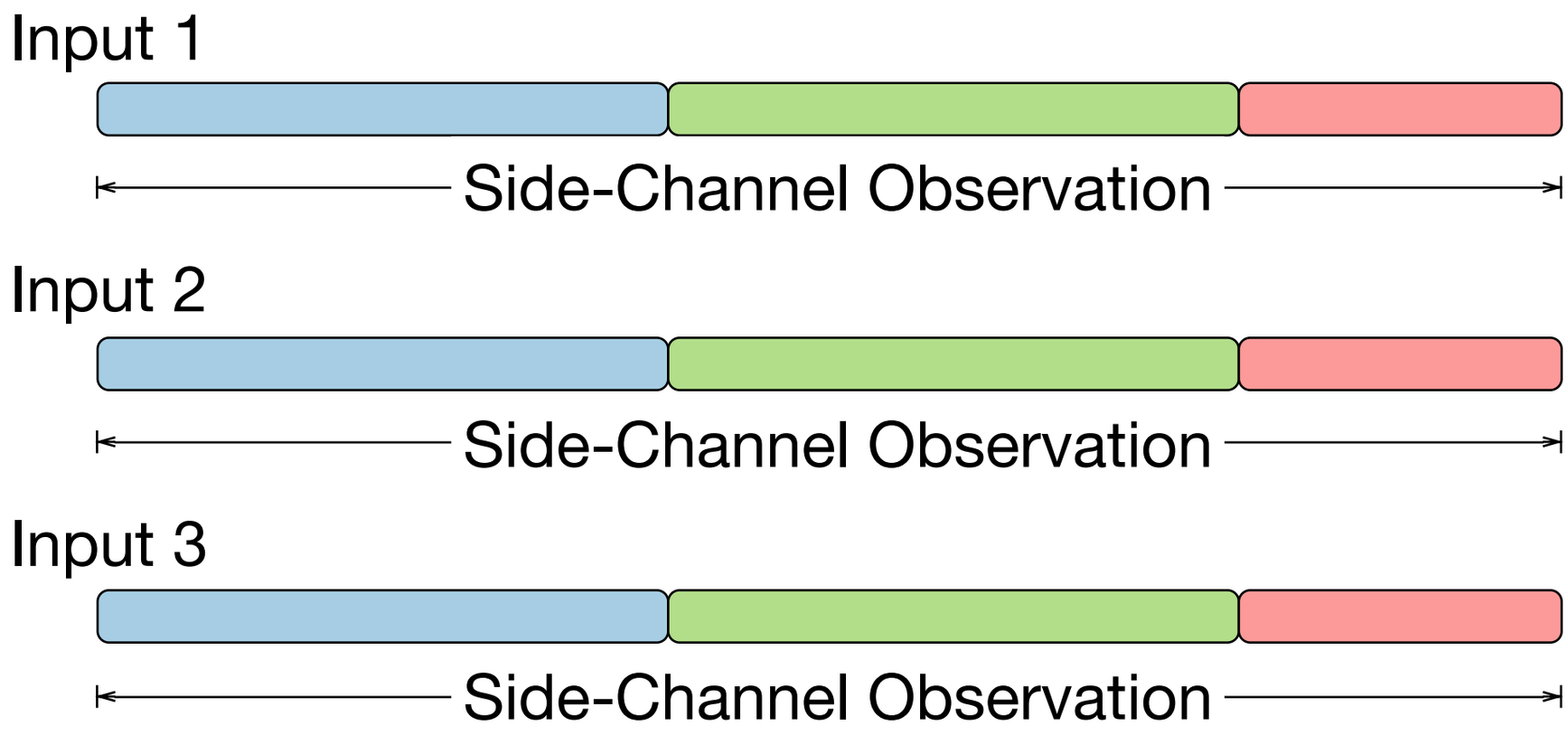
- Ideal defense decouples all side-channel observations from input
  - Usually requires manual programmer effort or custom hardware for each possible side channel



Side-Channel Attacks

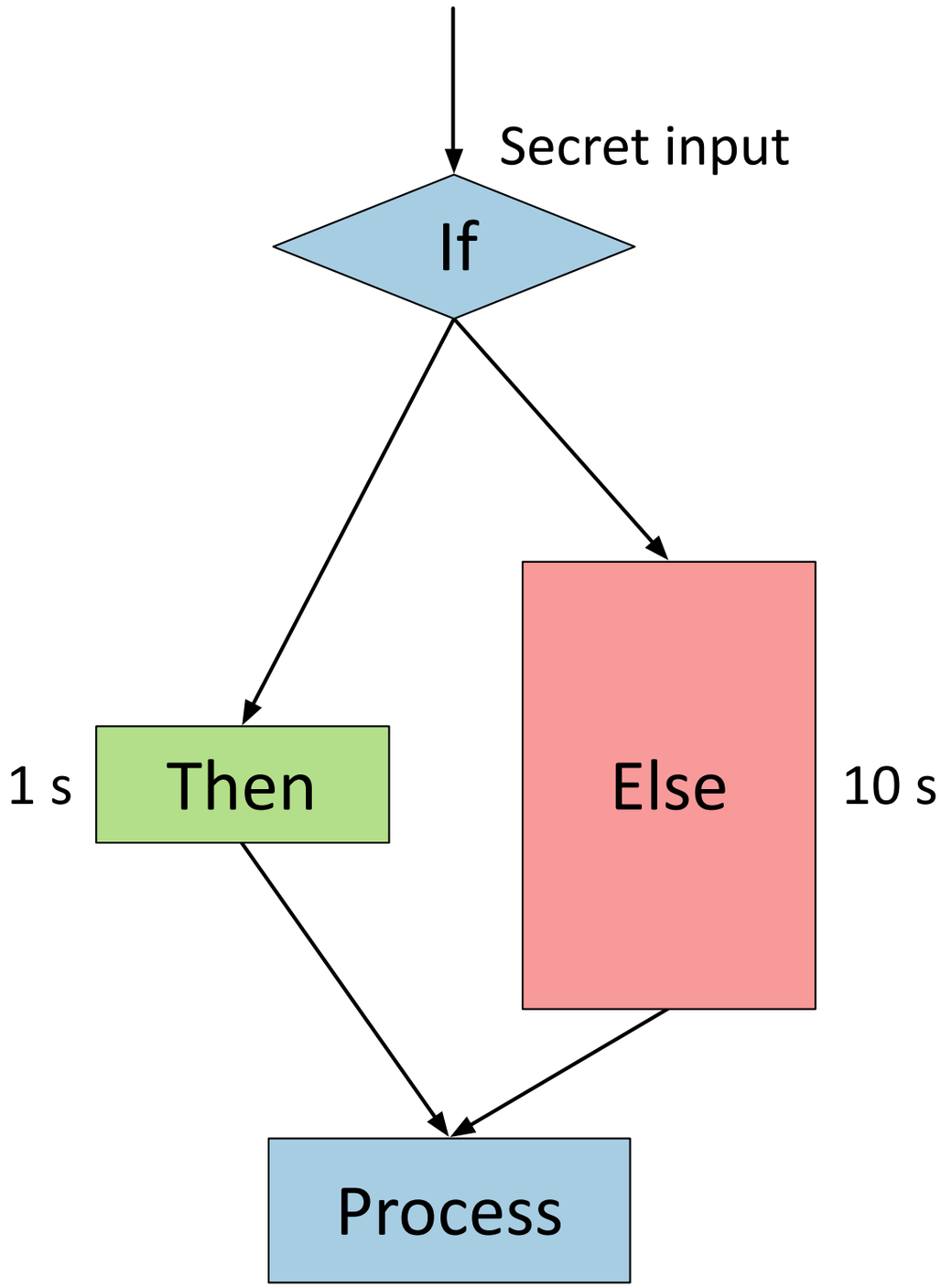
THE PROBLEM

- Ideal defense decouples all side-channel observations from input
  - Usually requires manual programmer effort or custom hardware for each possible side channel



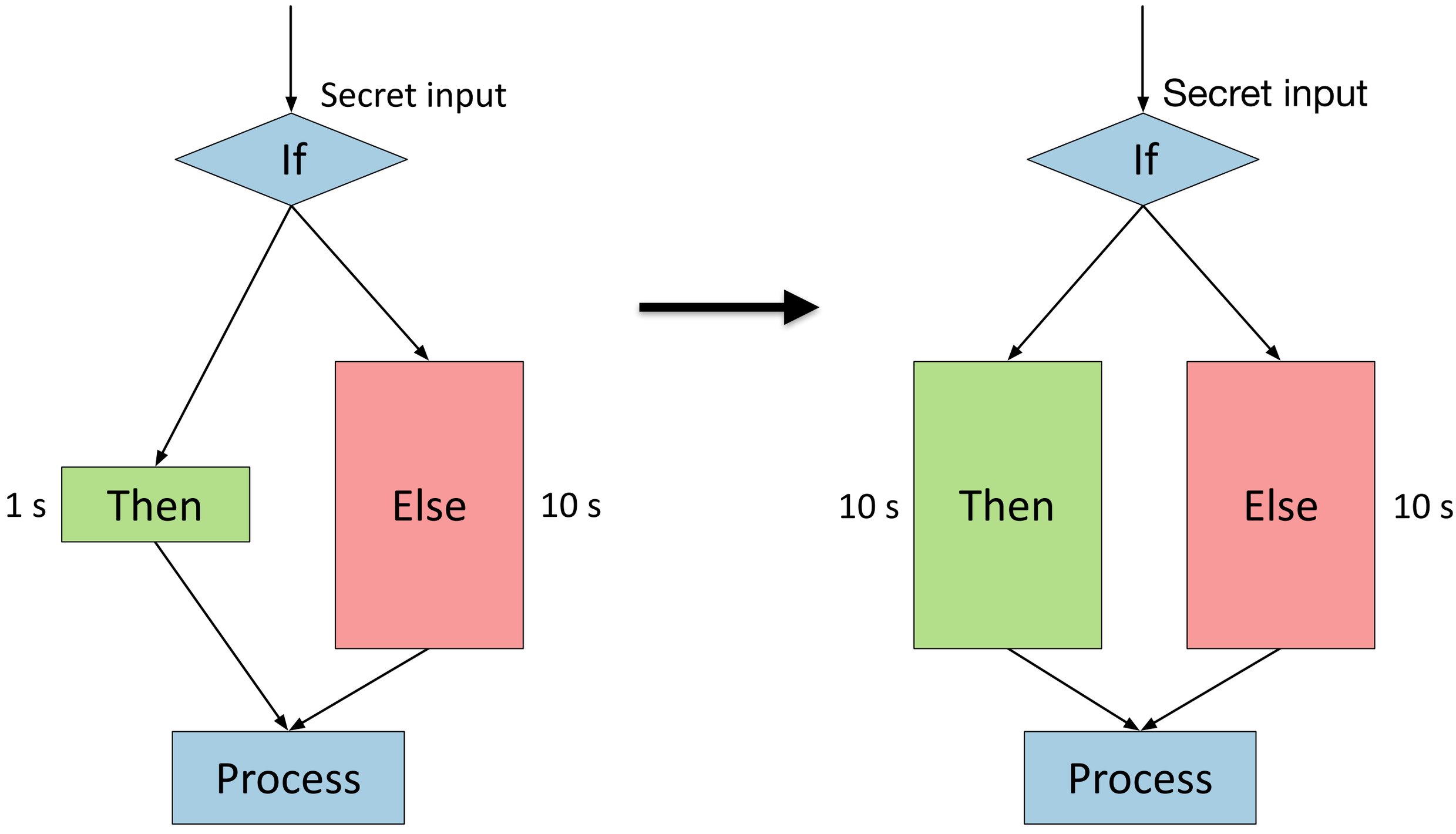
Manual Side-Channel Mitigation

THE PROBLEM



Manual Side-Channel Mitigation

THE PROBLEM



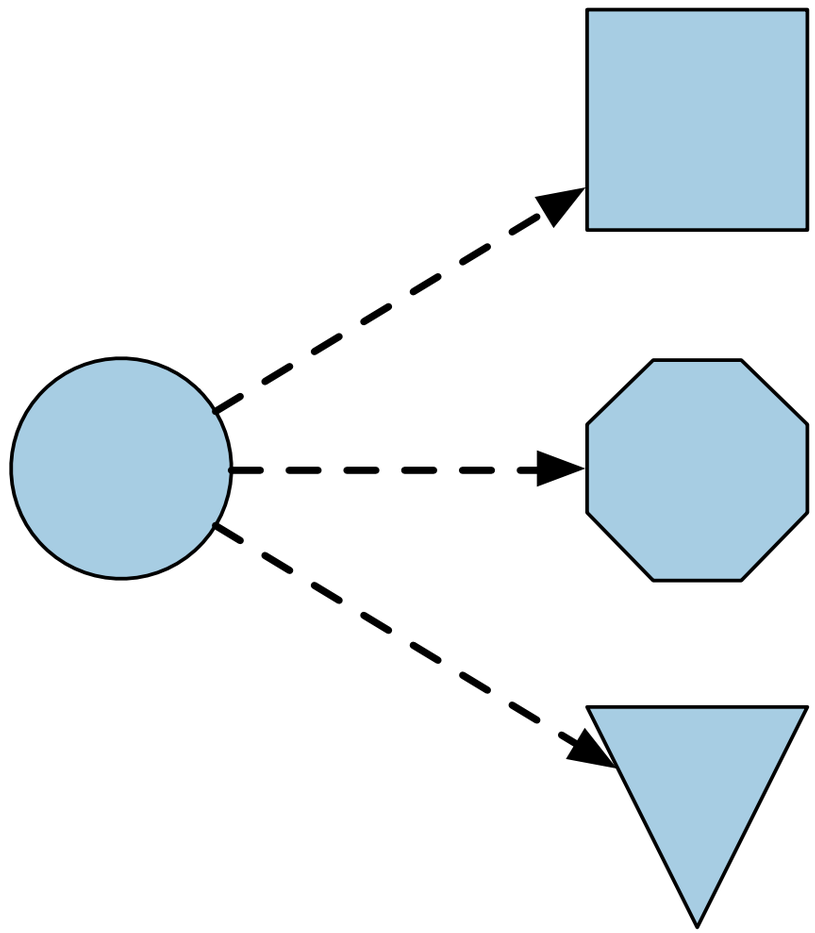
Automated Software Diversity

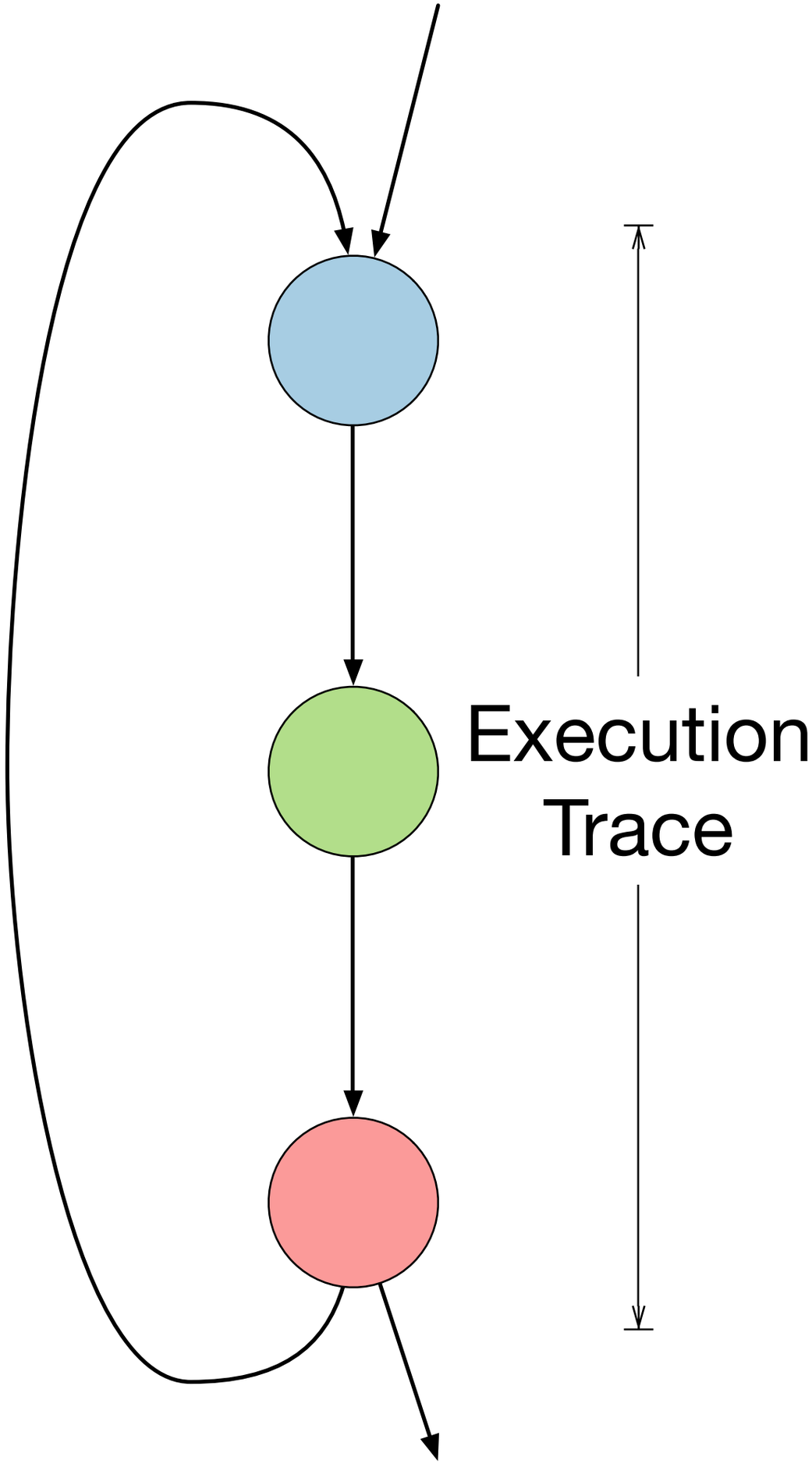
APPROACH

Multiple functionally equivalent copies which vary in implementation details

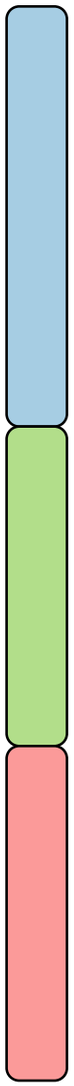
Techniques:

- NOP insertion
- Function reordering
- Register randomization
- Instruction substitution

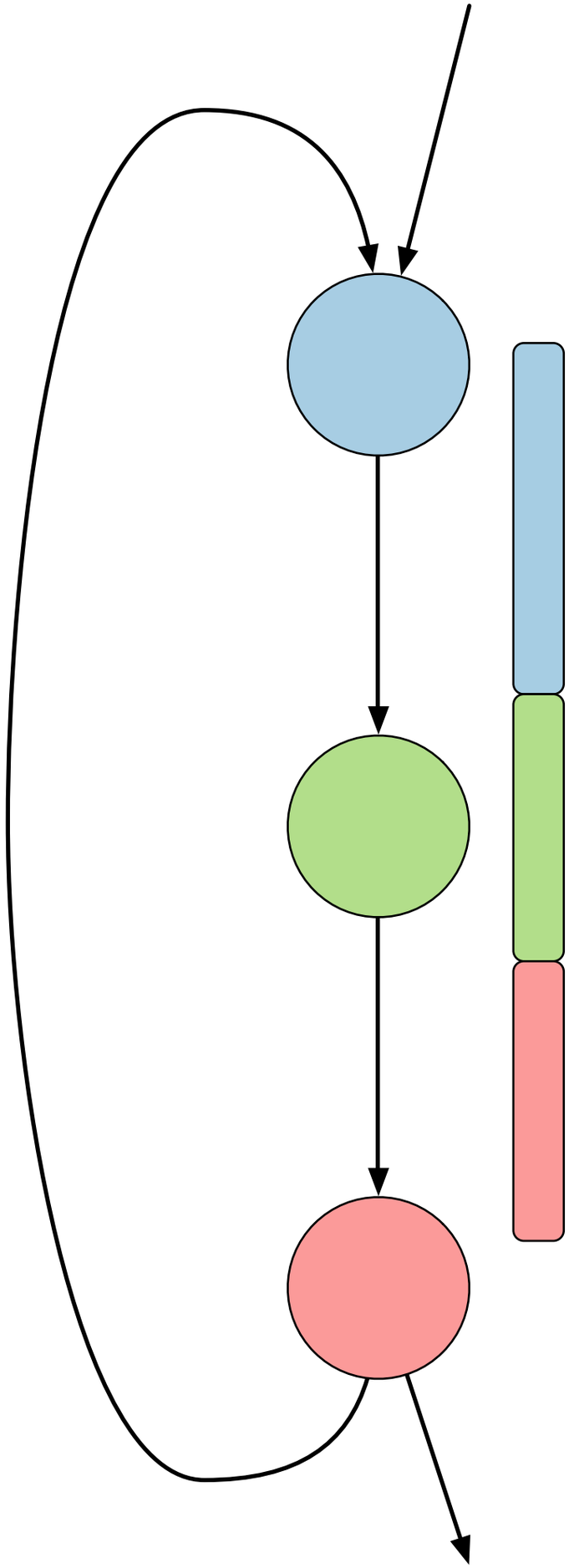


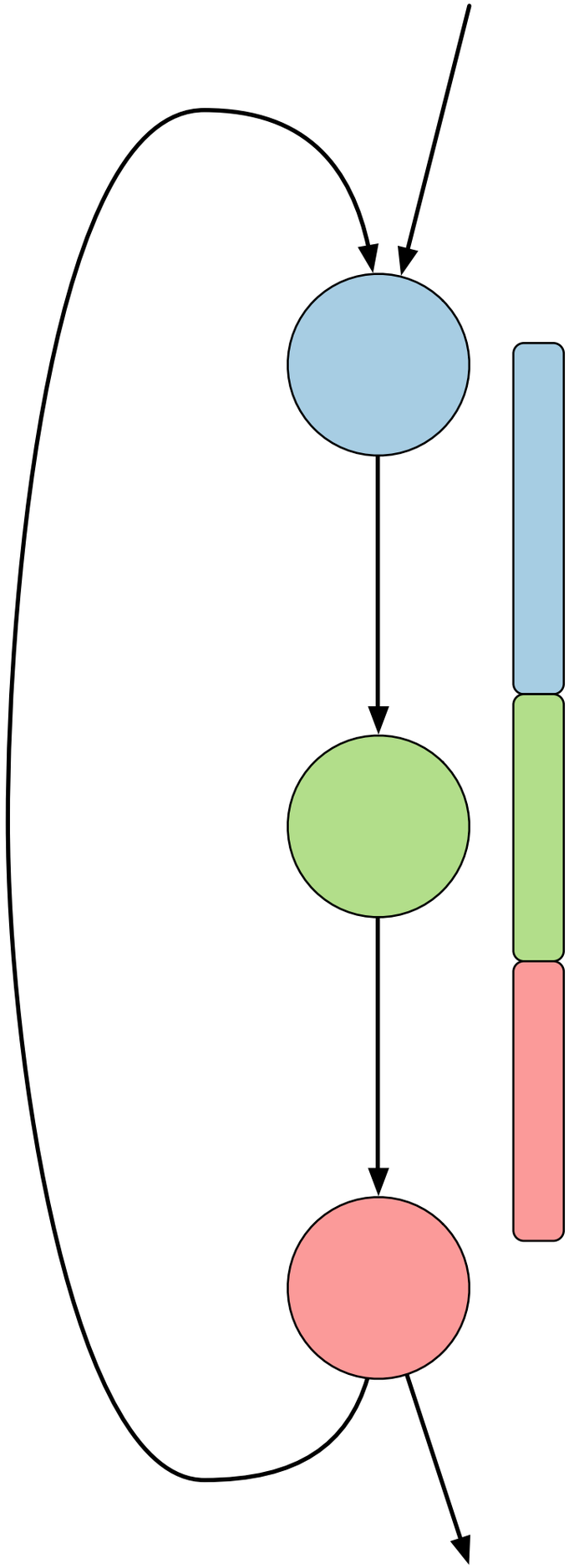


Execution
Trace

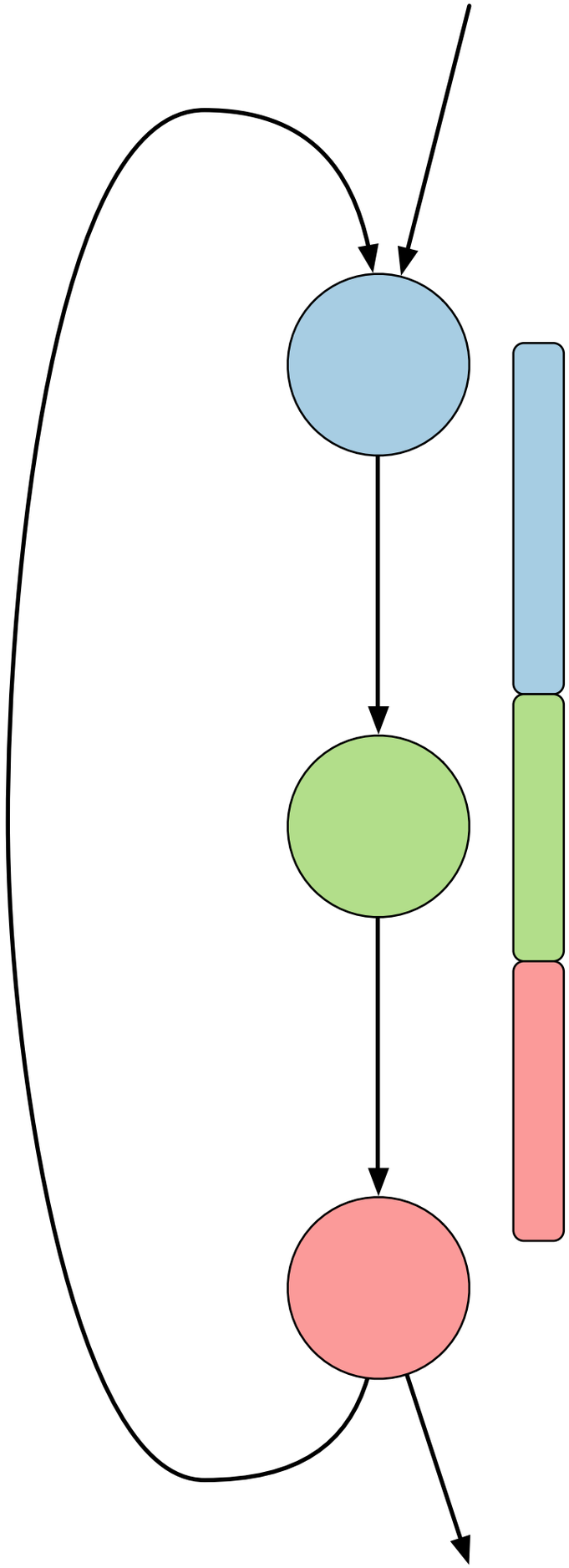


Timing
Side Channel

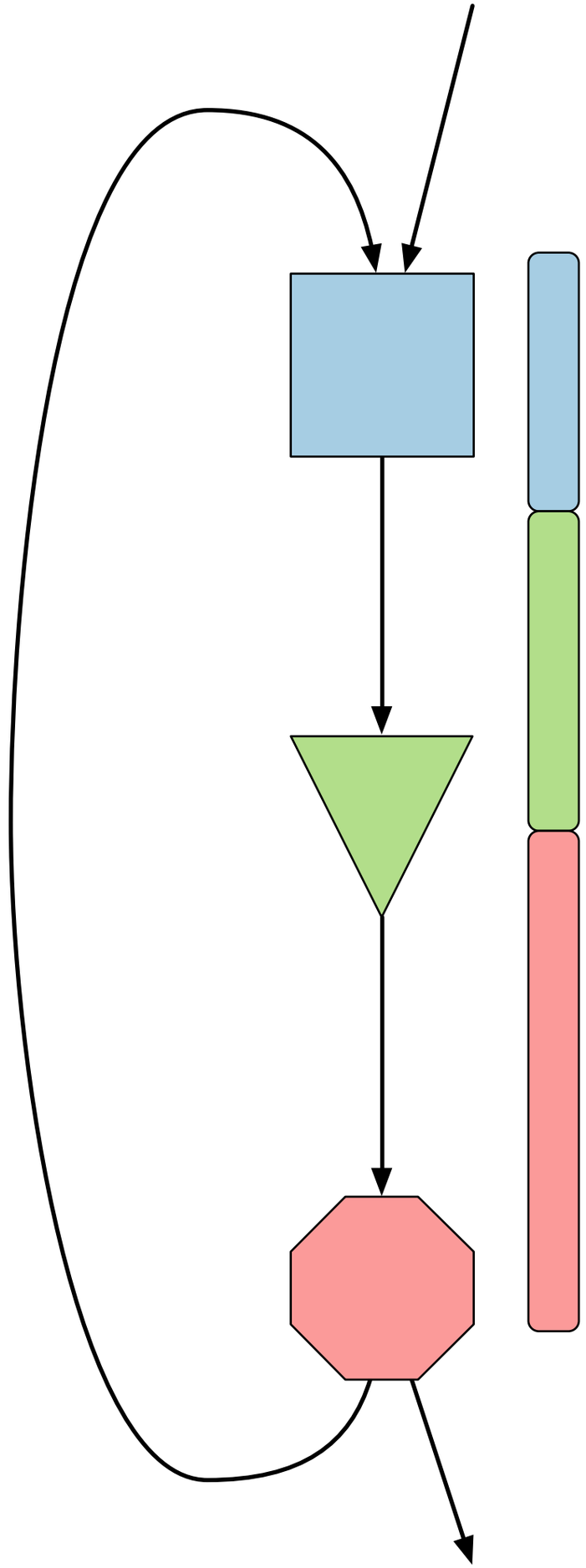


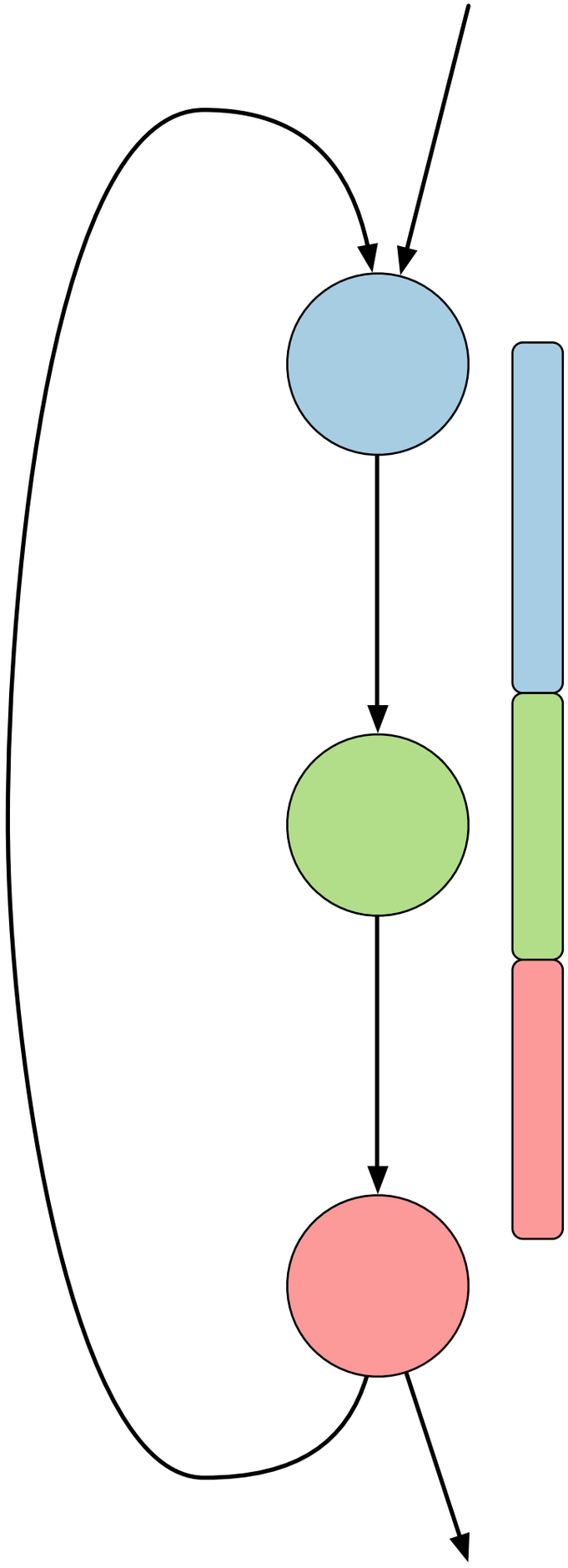


Diversity

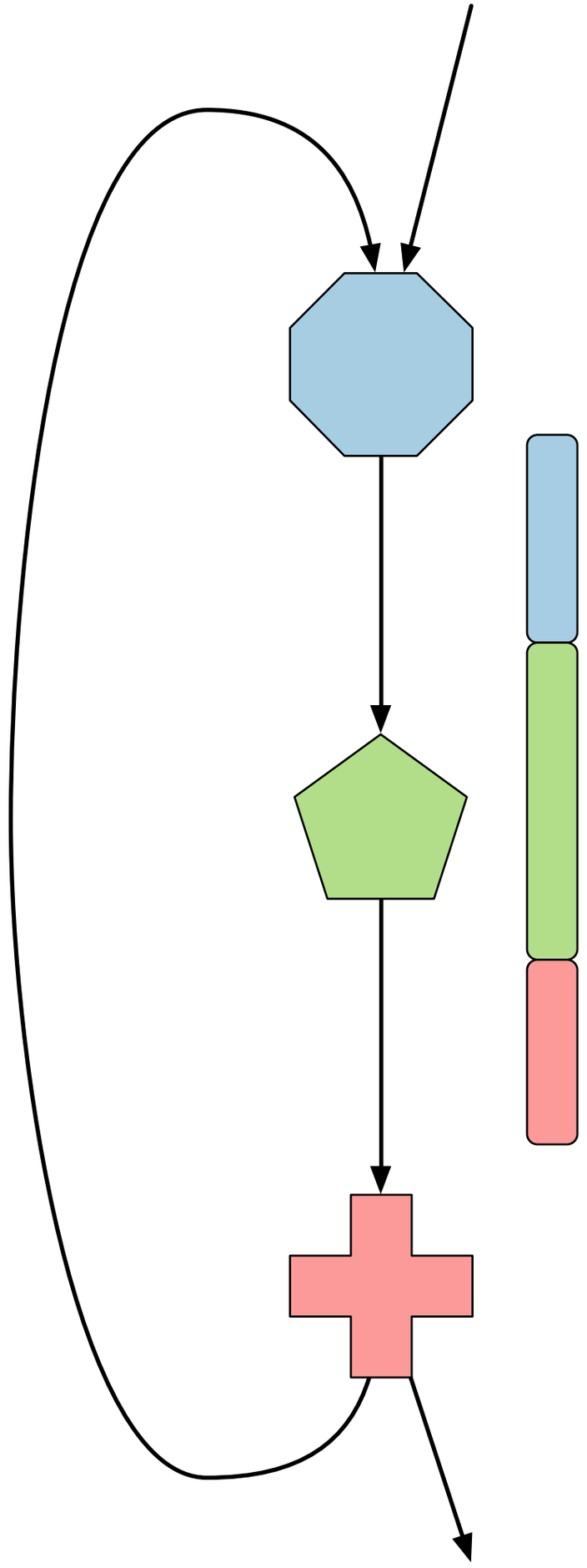


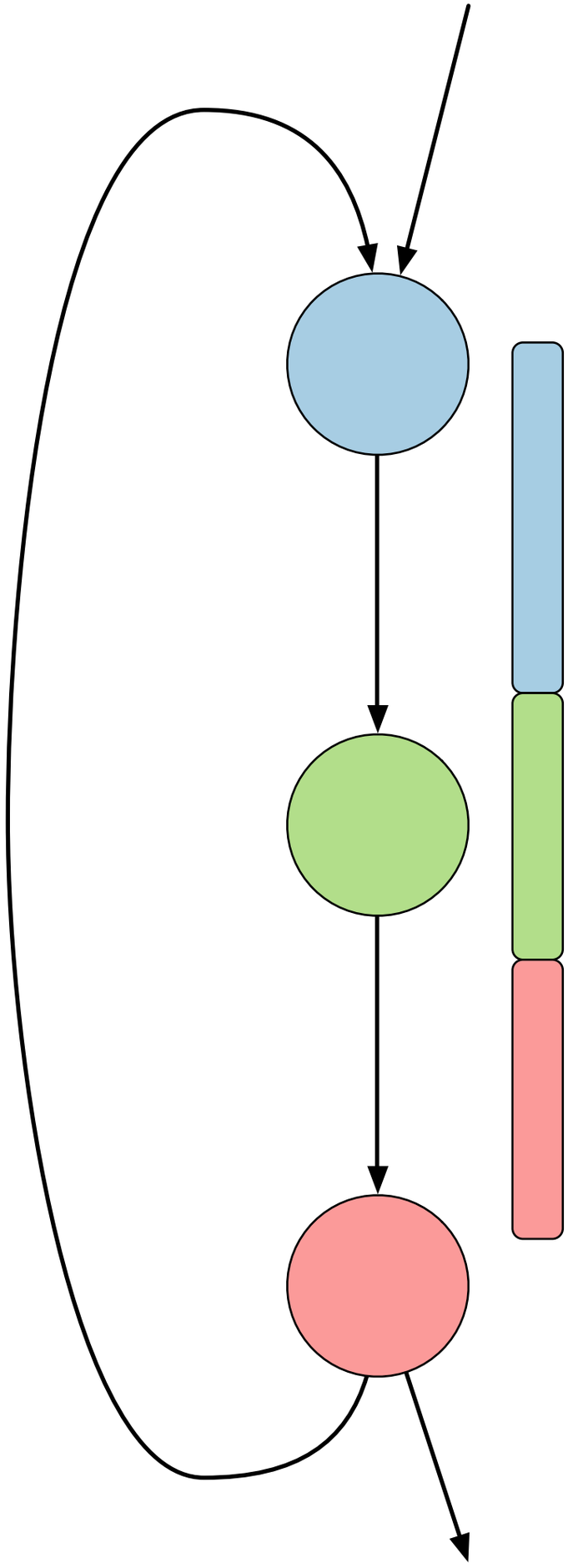
Diversity



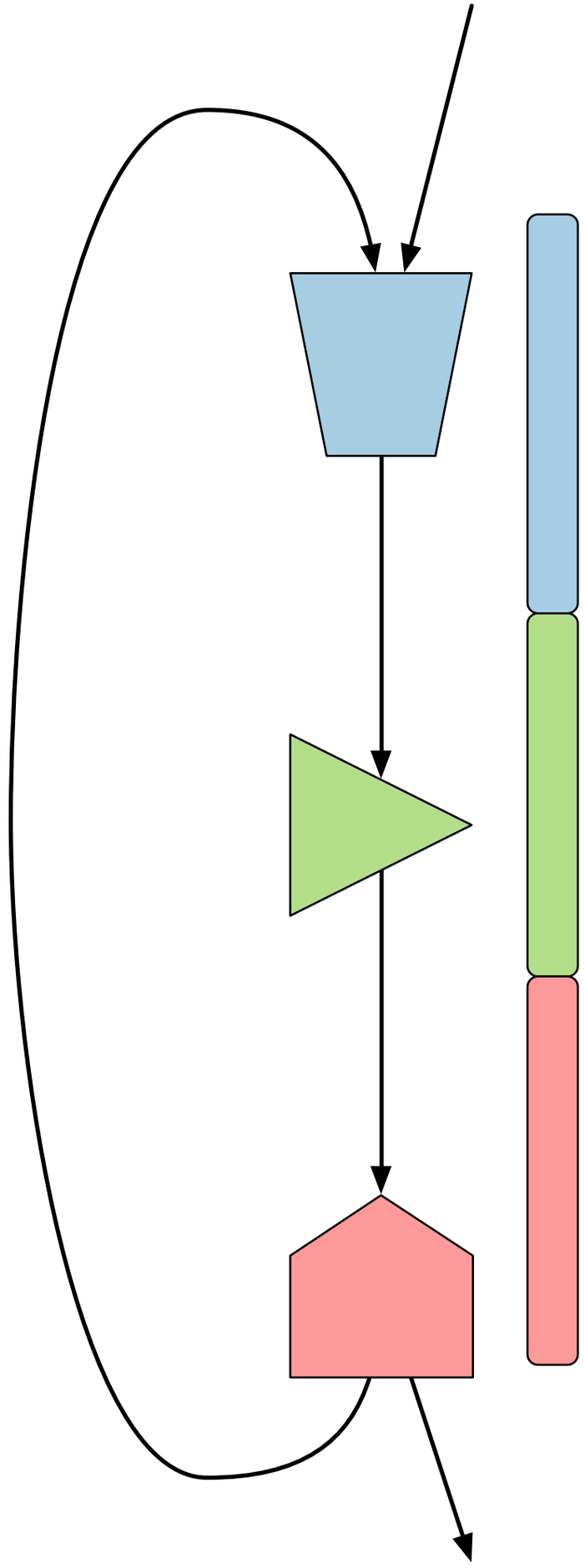


Diversity

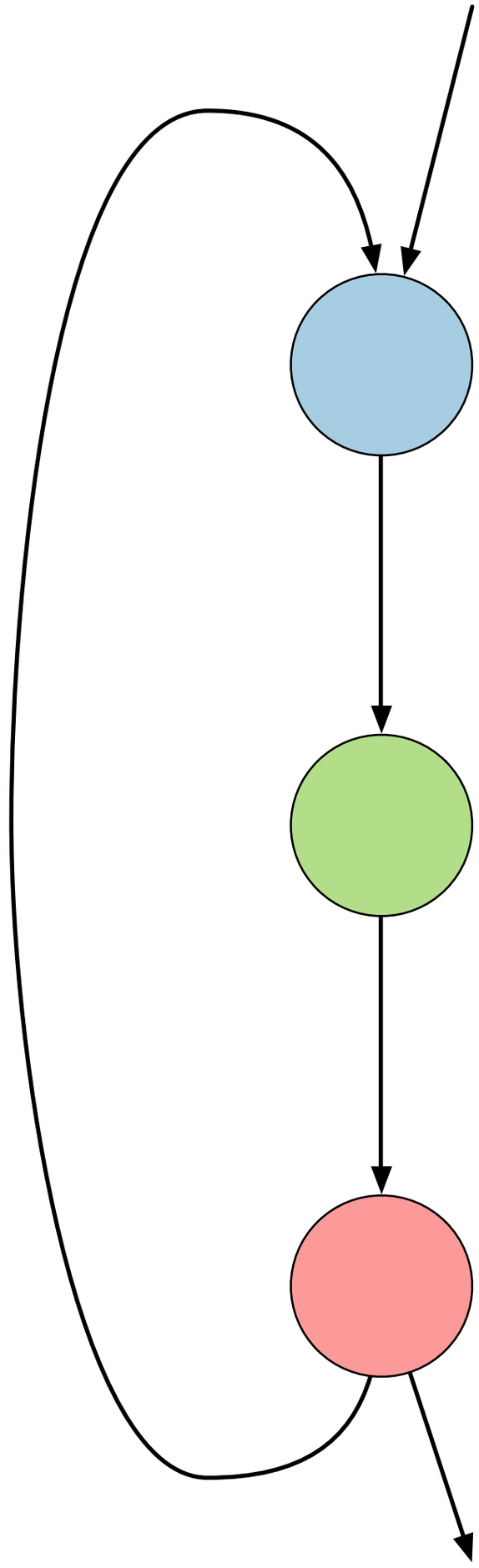




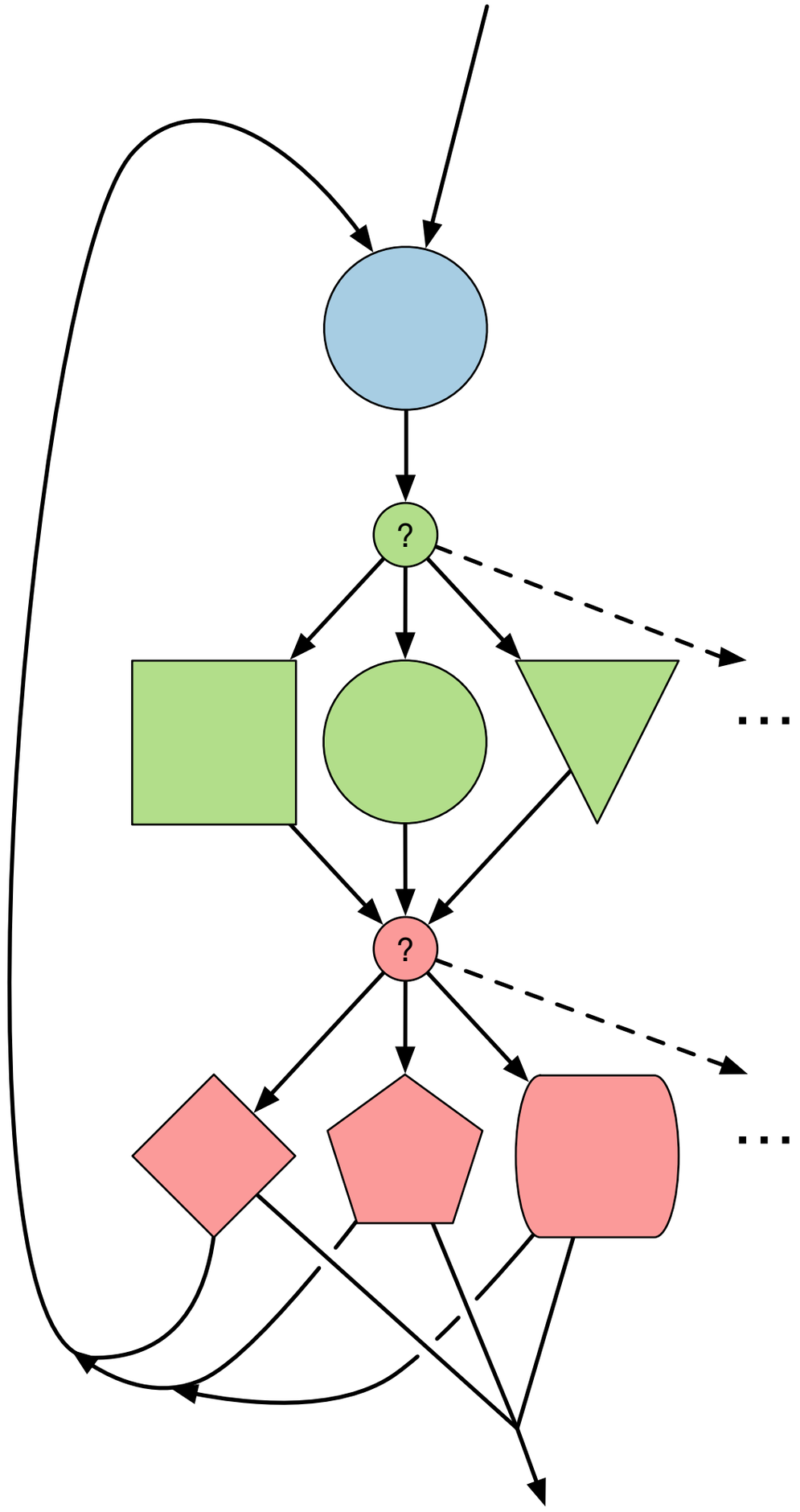
Diversity

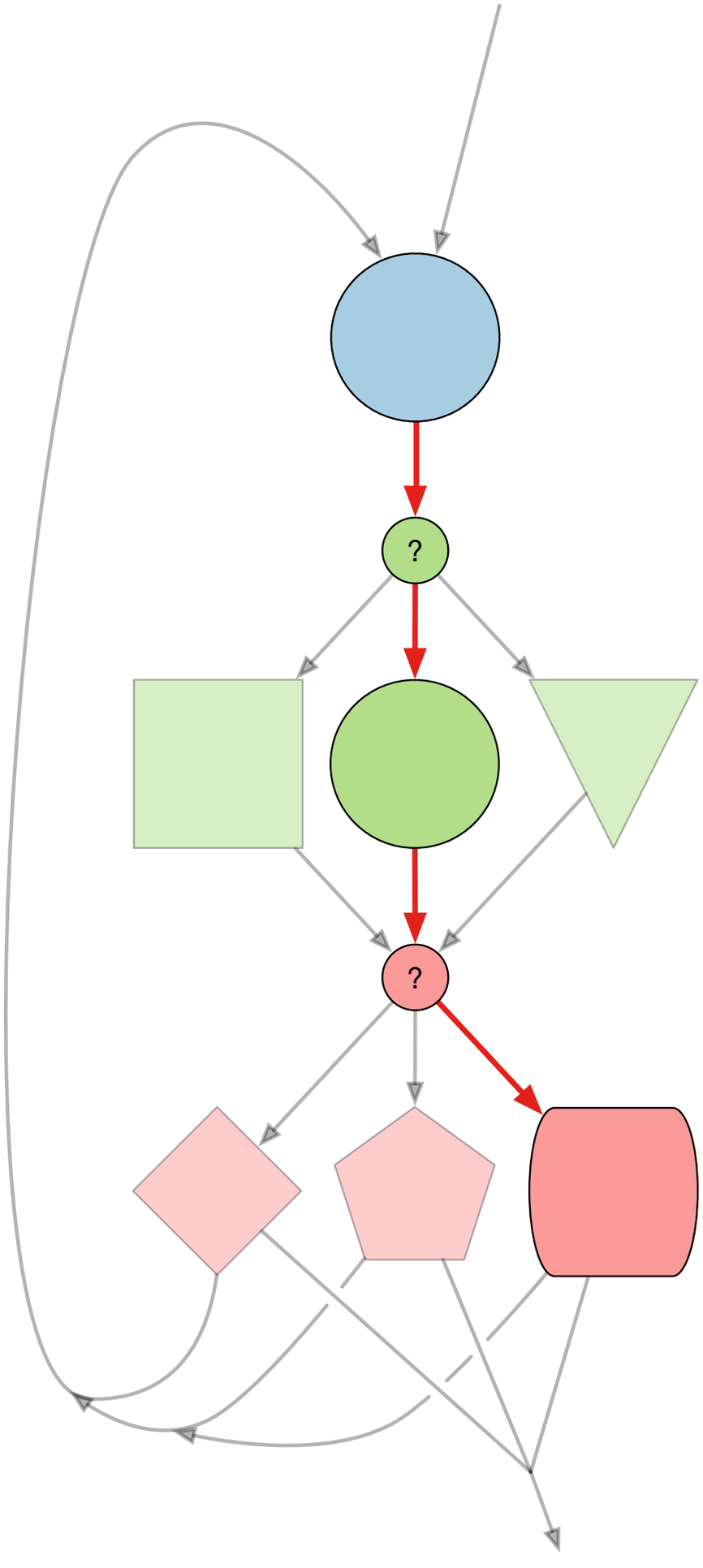


Control-Flow Diversity



Control-Flow Diversity

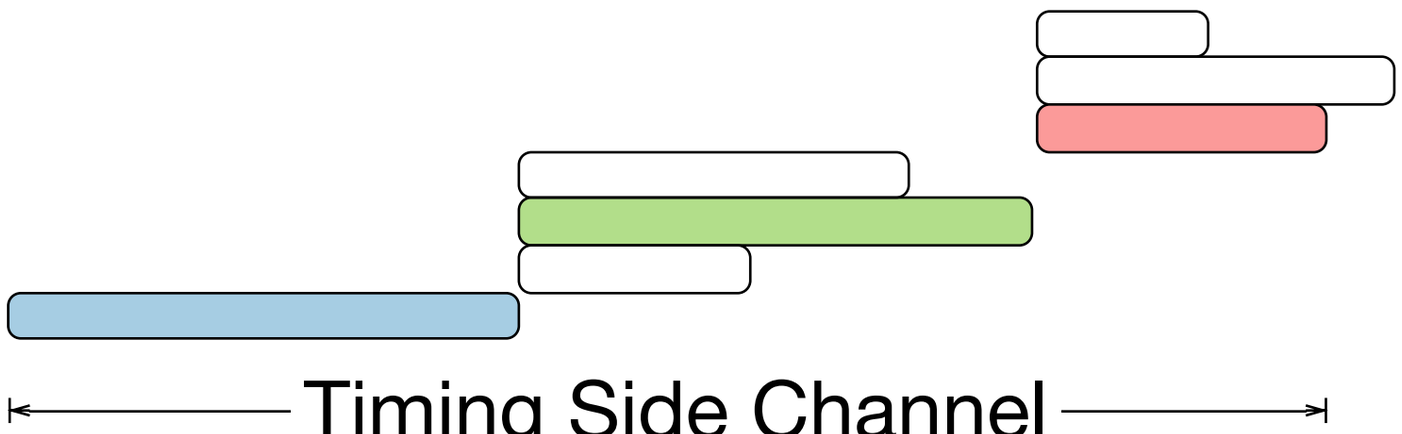




Side-Channel Variation

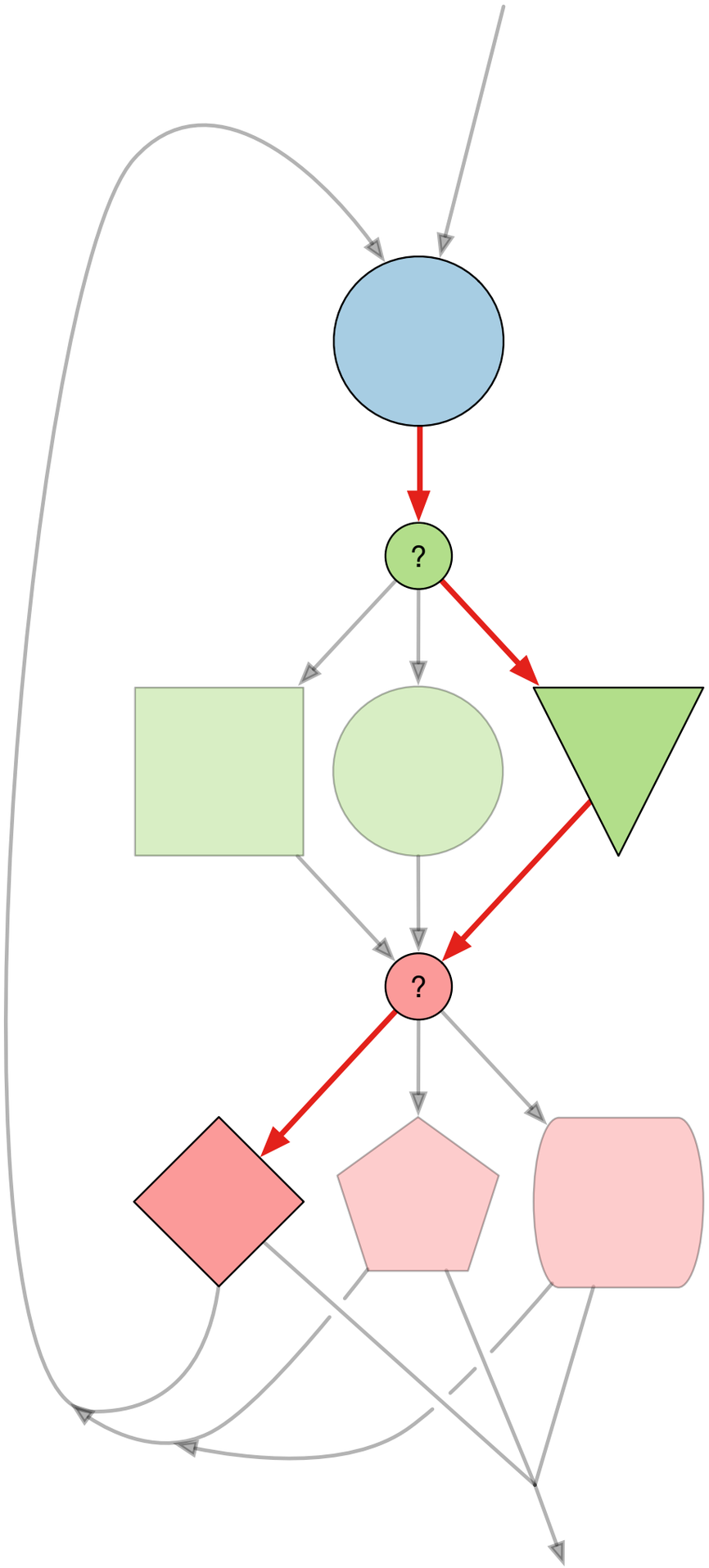
APPROACH

Each loop iteration results in different side-channel observations, even with the same input.

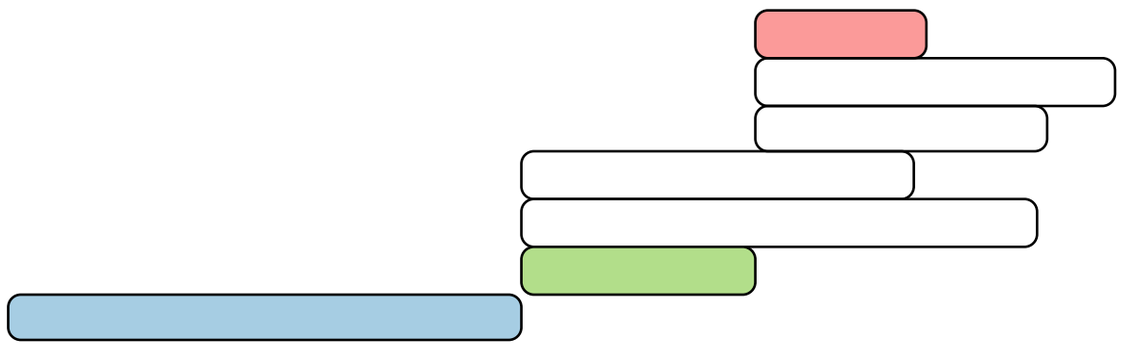


Side-Channel Variation

APPROACH

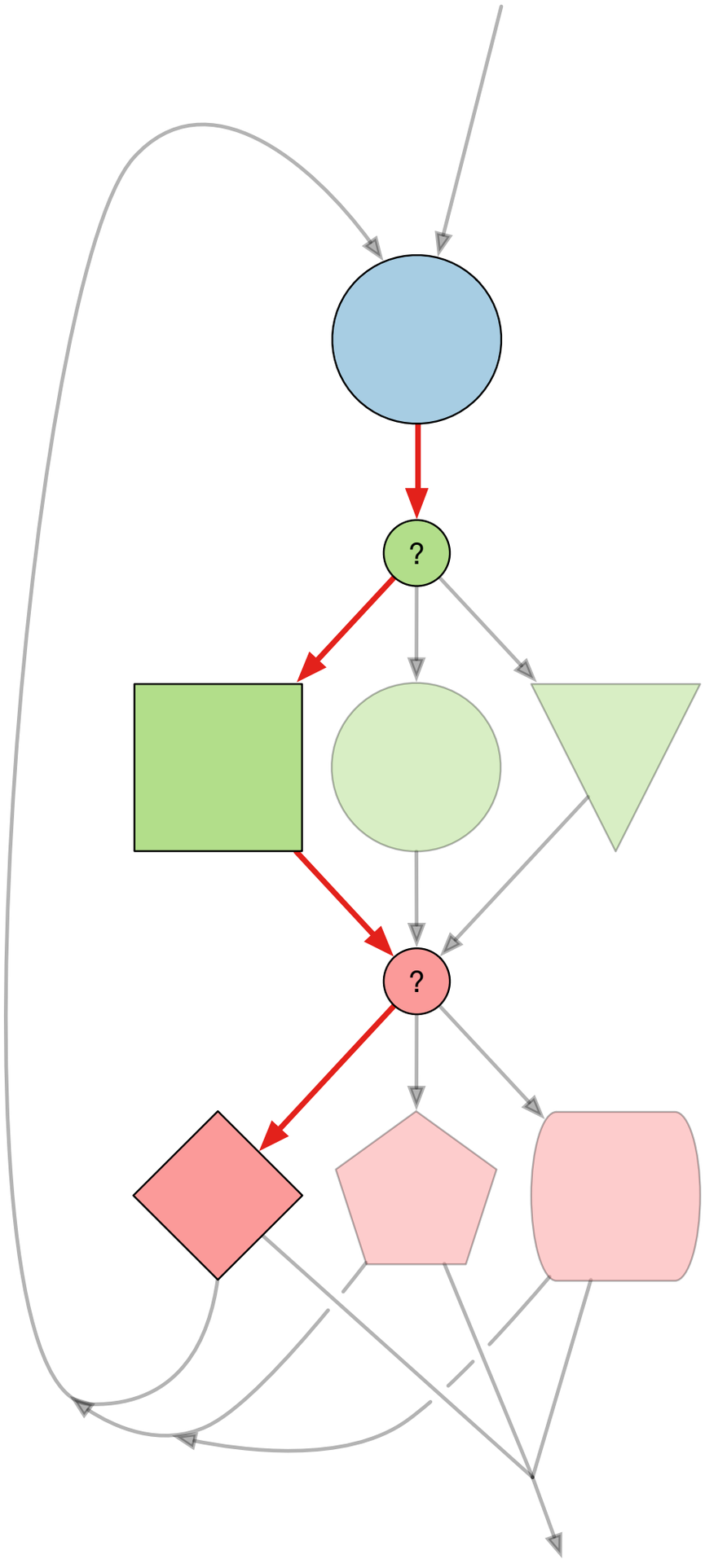


Each loop iteration results in different side-channel observations, even with the same input.

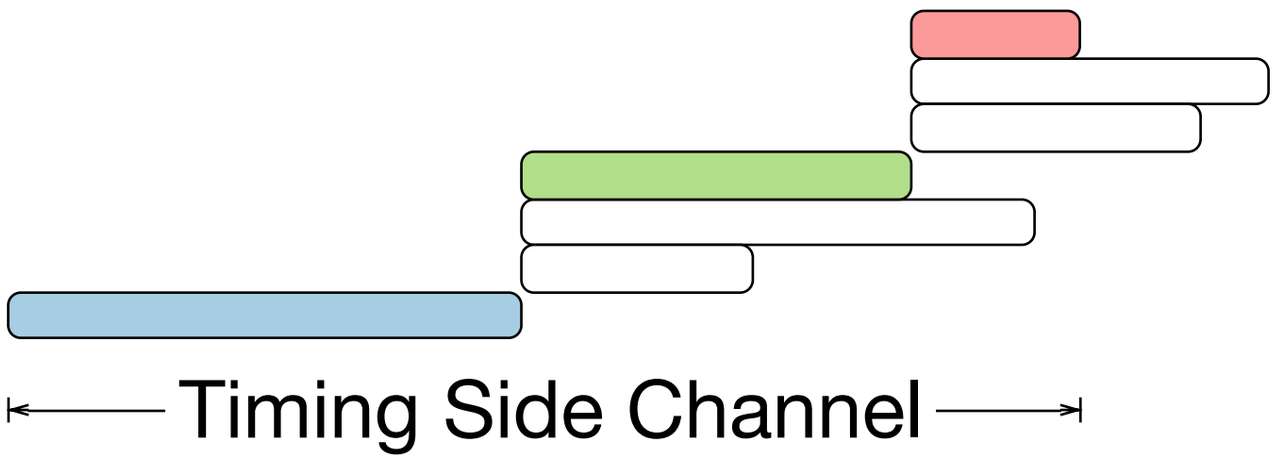


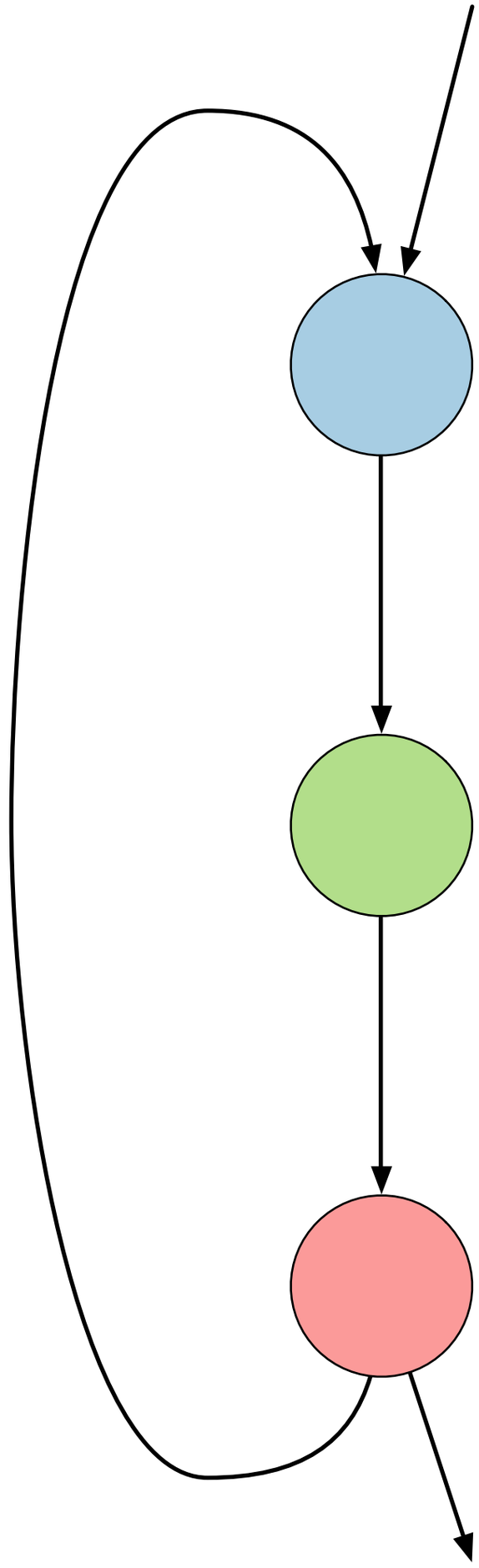
Side-Channel Variation

APPROACH

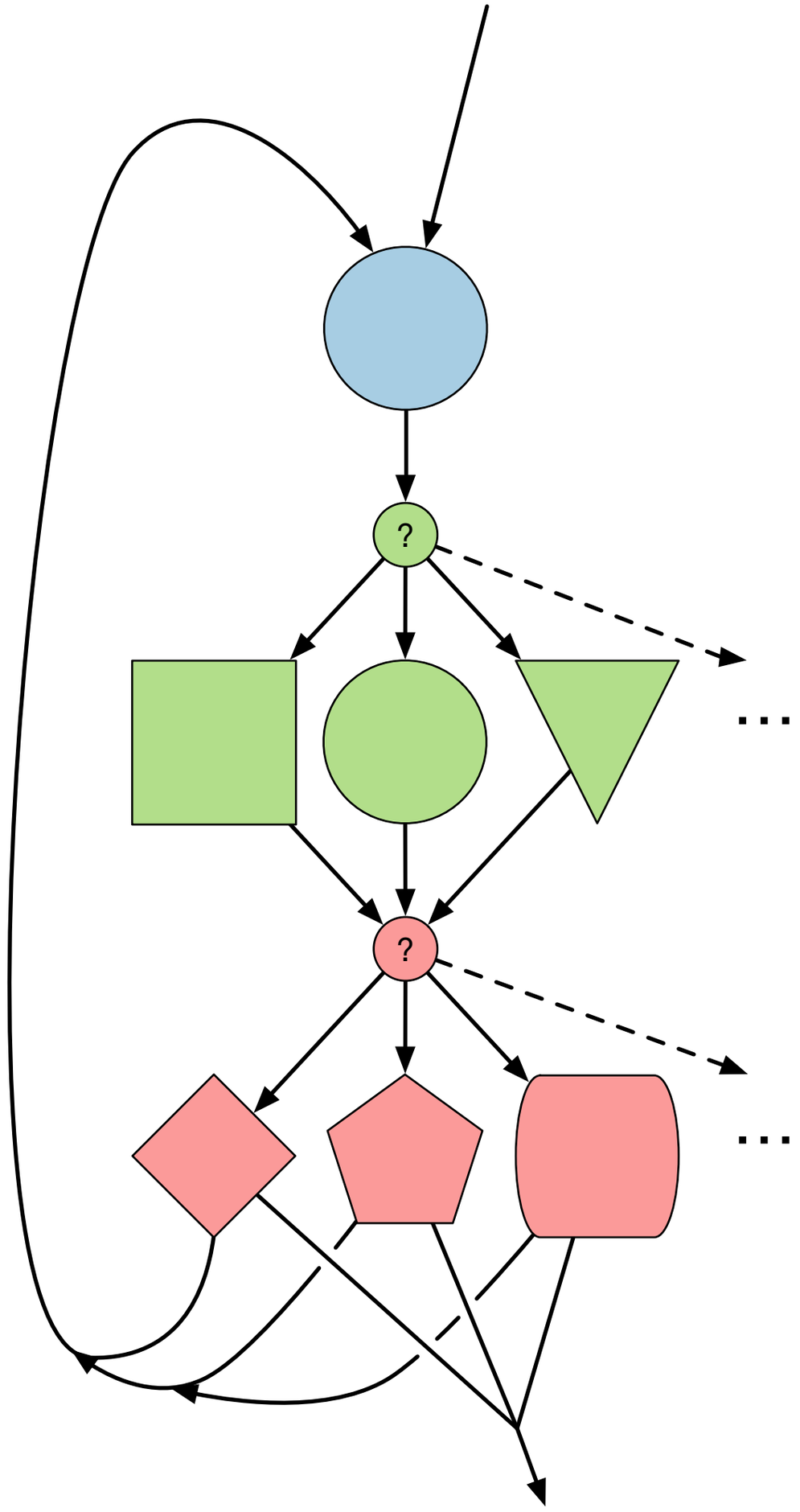


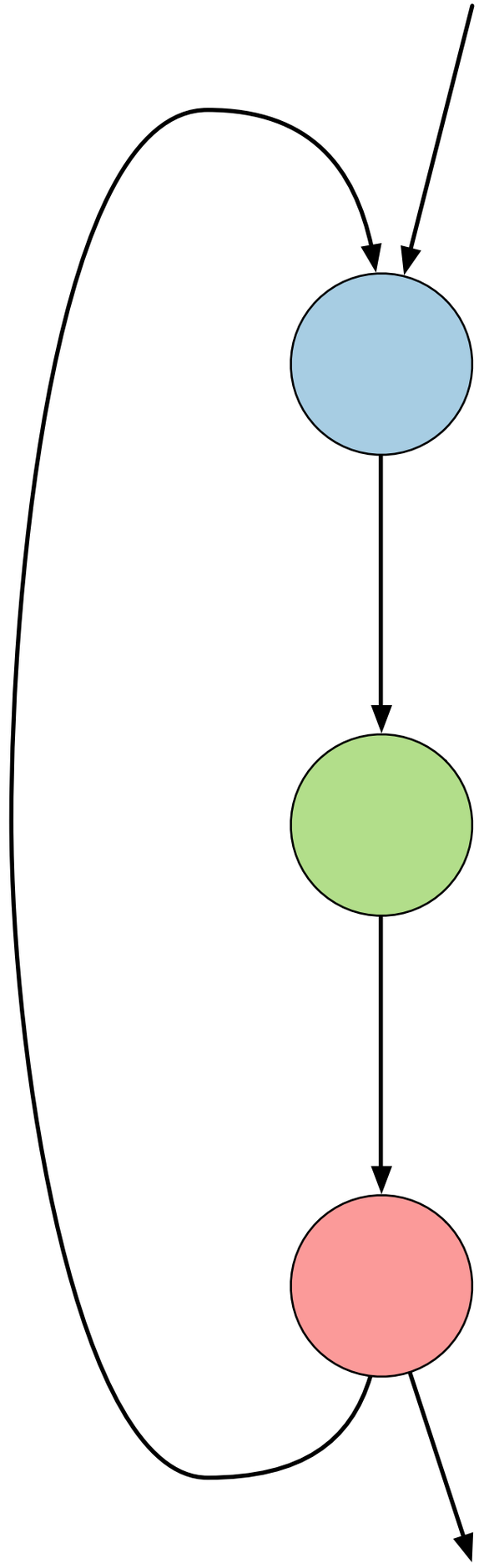
Each loop iteration results in different side-channel observations, even with the same input.



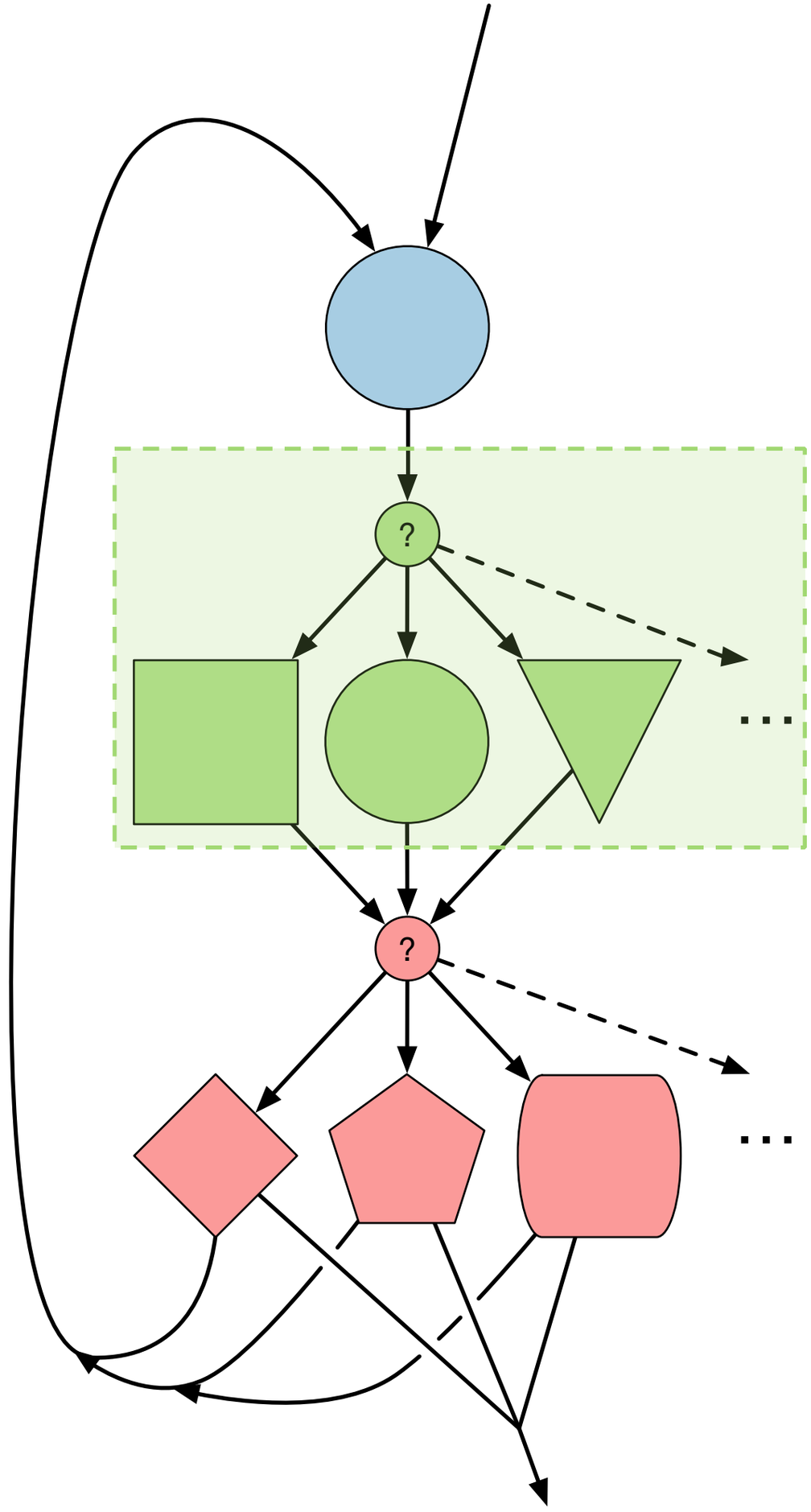


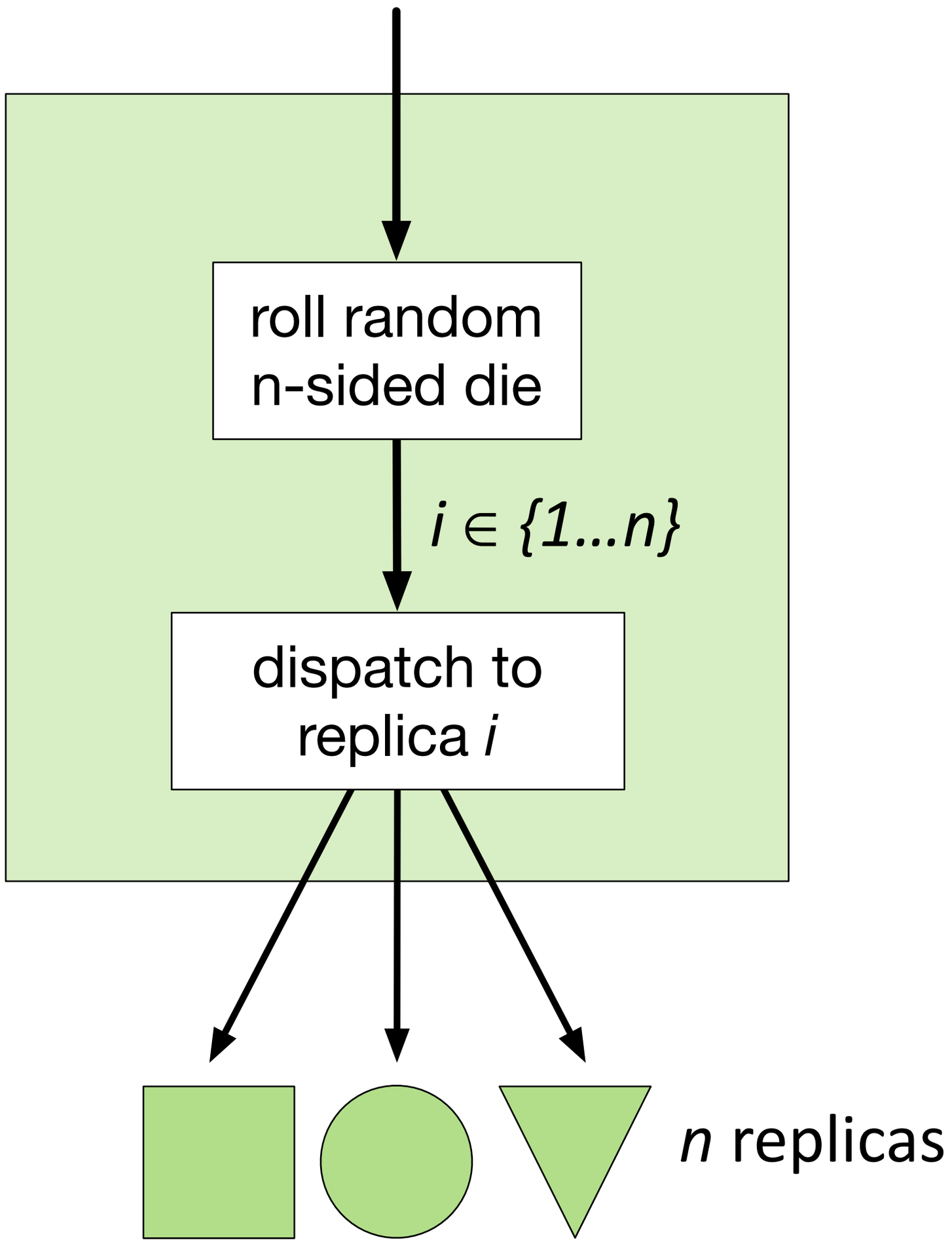
Control-Flow Diversity





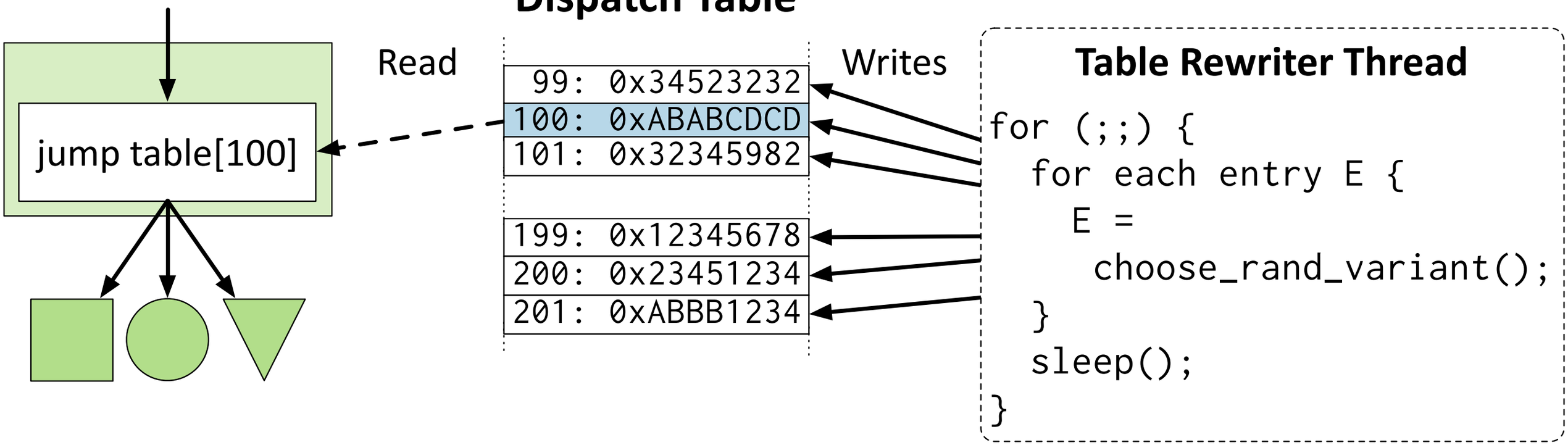
Control-Flow Diversity




roll random
n-sided die
i ∈ {1...n}
dispatch to
replica i
n replicas

Optimized Asynchronous Update

IMPLEMENTATION



AES Cache Side-Channel Attack

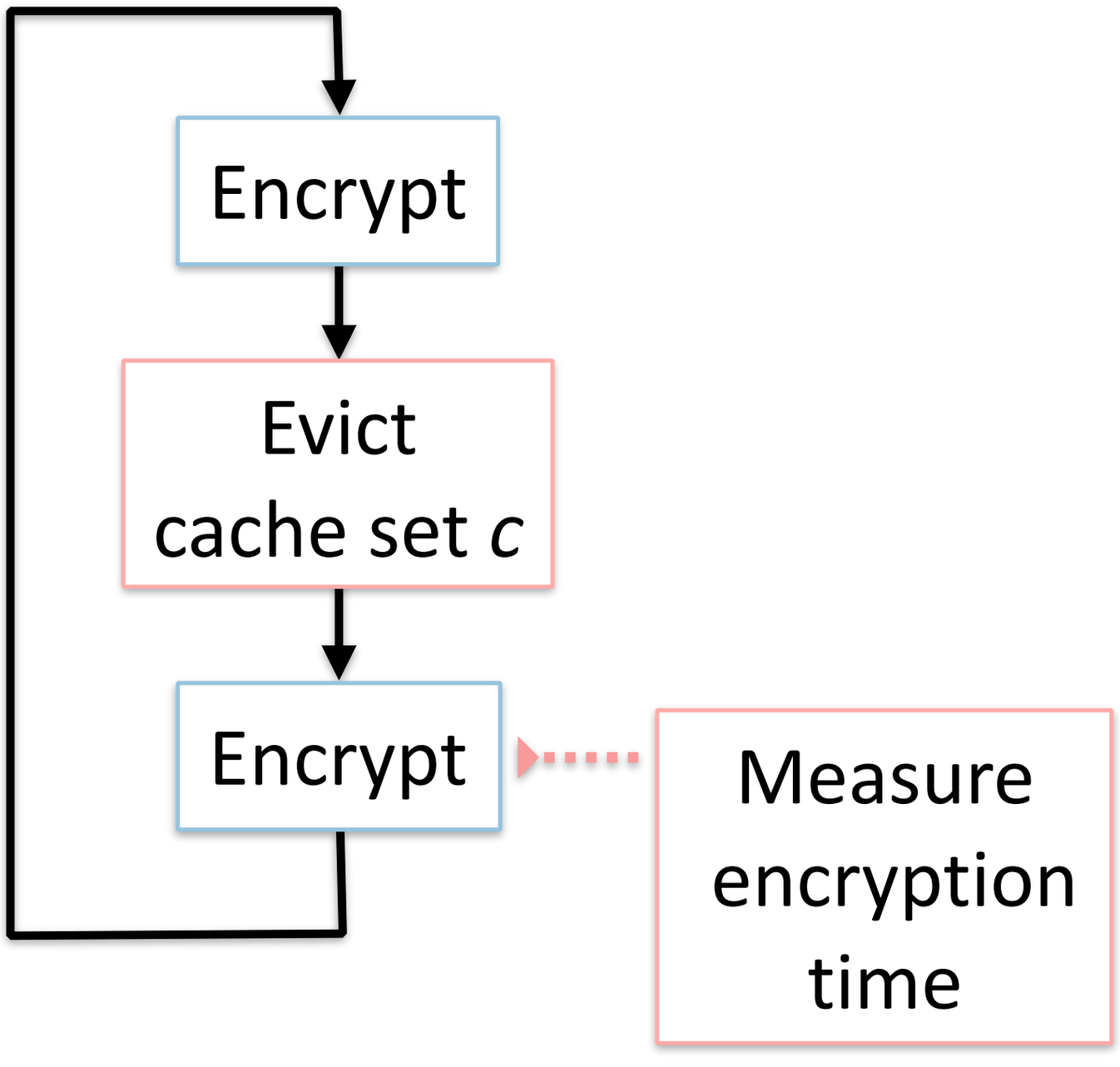
EVALUATION

- Practical attack on the libgcrypt AES implementation
  - Targets L2 caching of AES S-box table lookups
- Modern hardware
  - Intel Core 2 Quad Q9300, 2.5Ghz
- Two types of cache side channels [1]:
  - EVICT+TIME: Overall timing
  - PRIME+PROBE: Cache usage

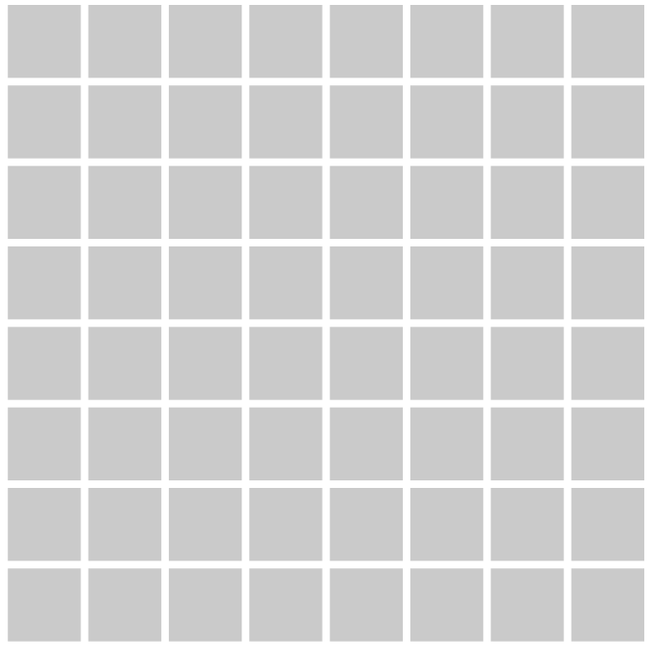
[1] E. Tromer, D. A. Osvik, and A. Shamir, “Efficient cache attacks on AES, and countermeasures,” Journal of Cryptology, vol. 23, no. 1, pp. 37–71, Jan. 2010.

EVICT+TIME Attack

EVALUATION

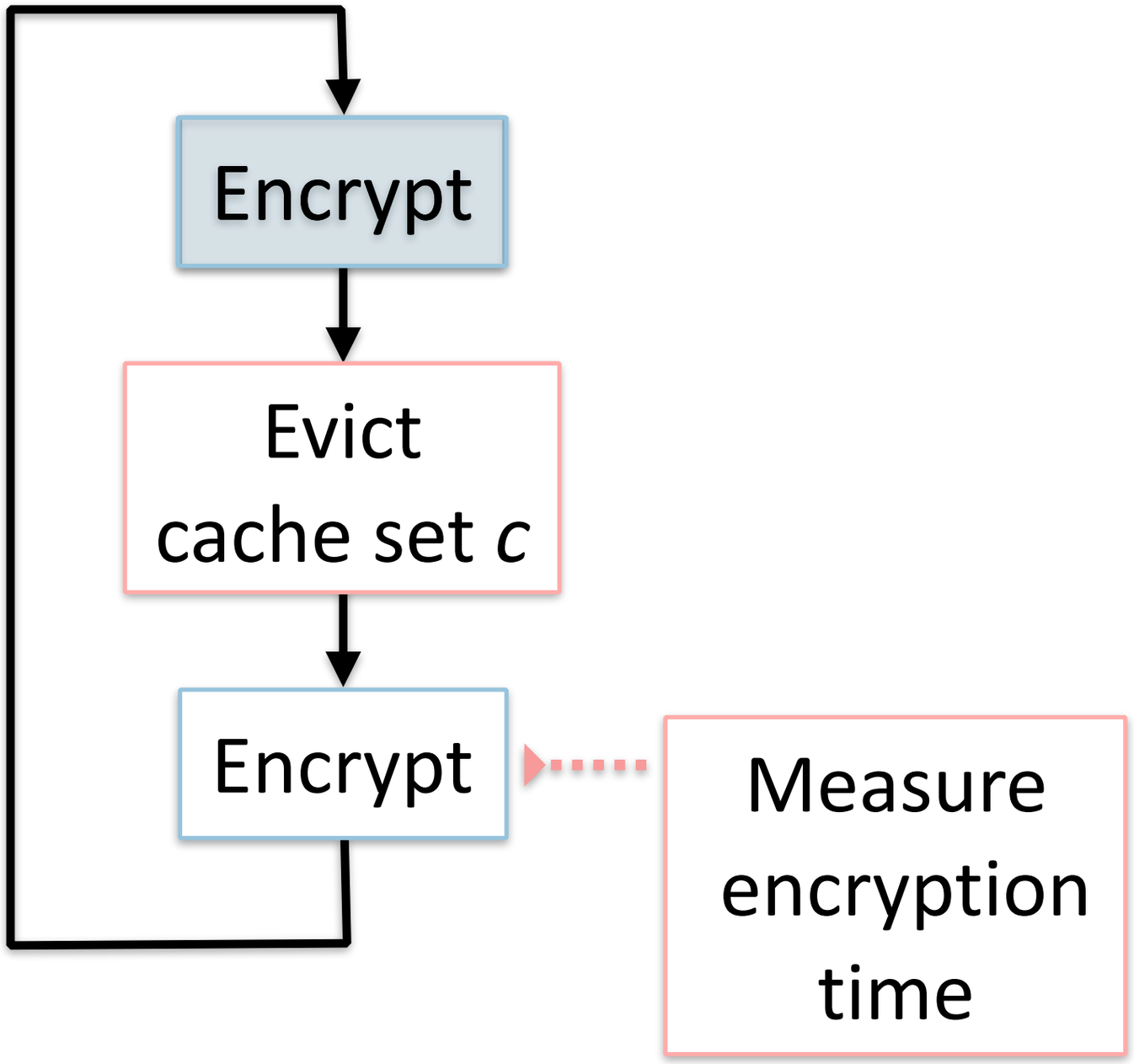


Cache Contents

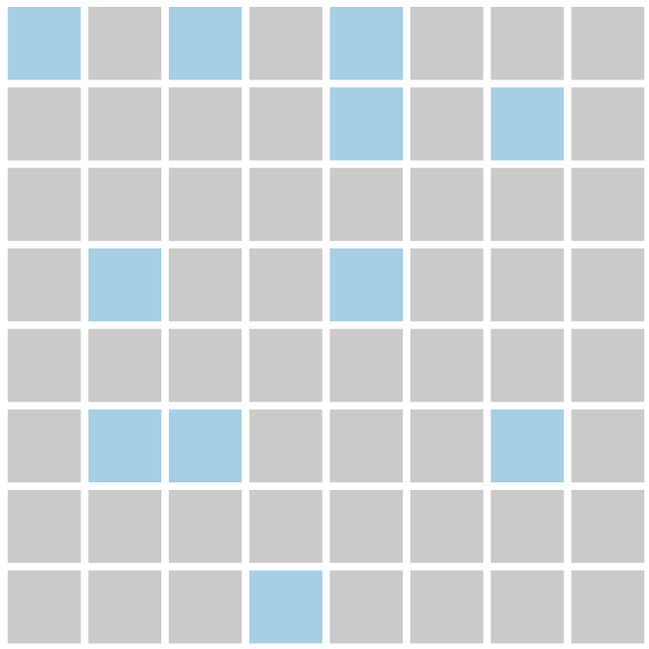


EVICT+TIME Attack

EVALUATION

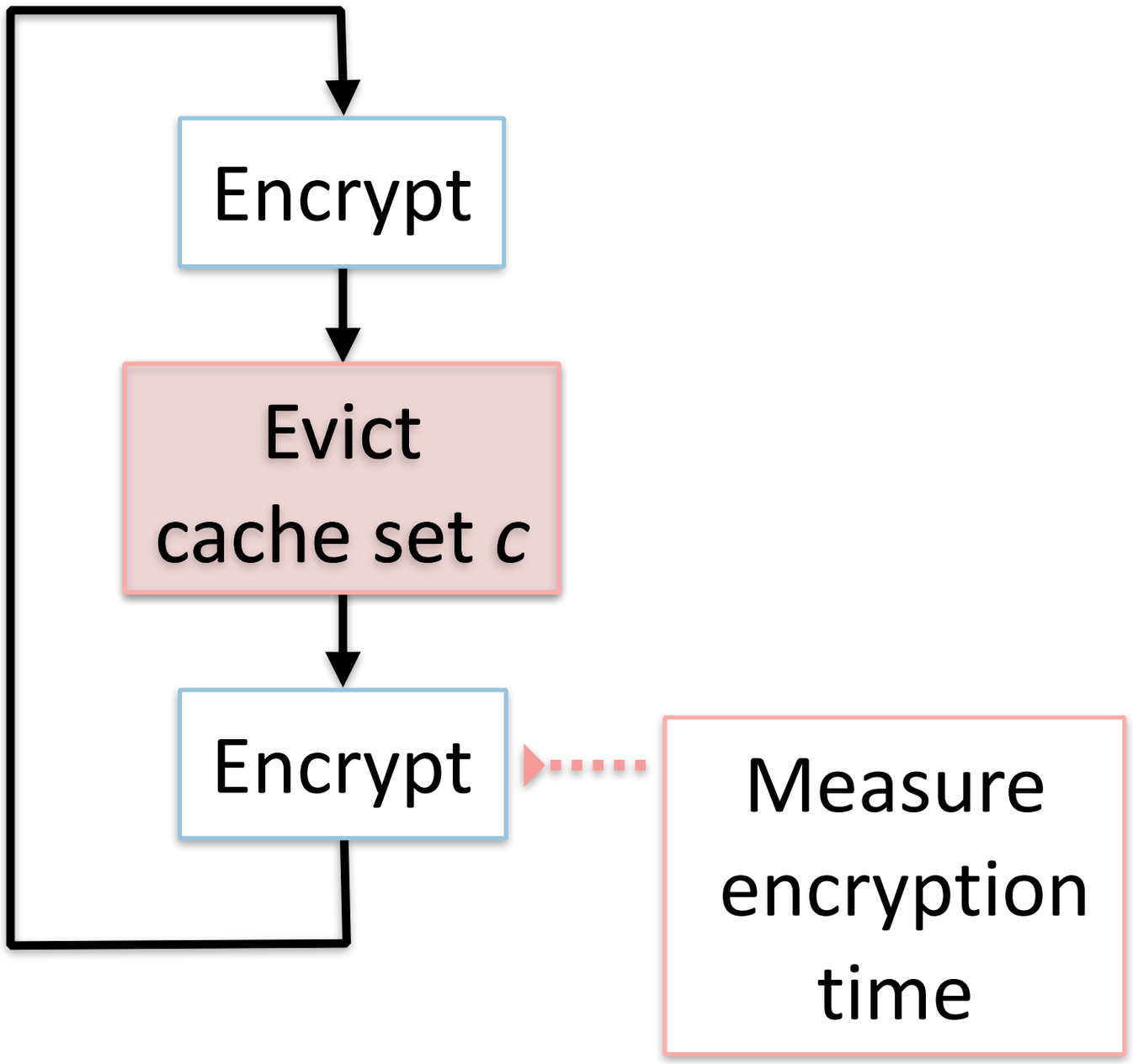


Cache Contents

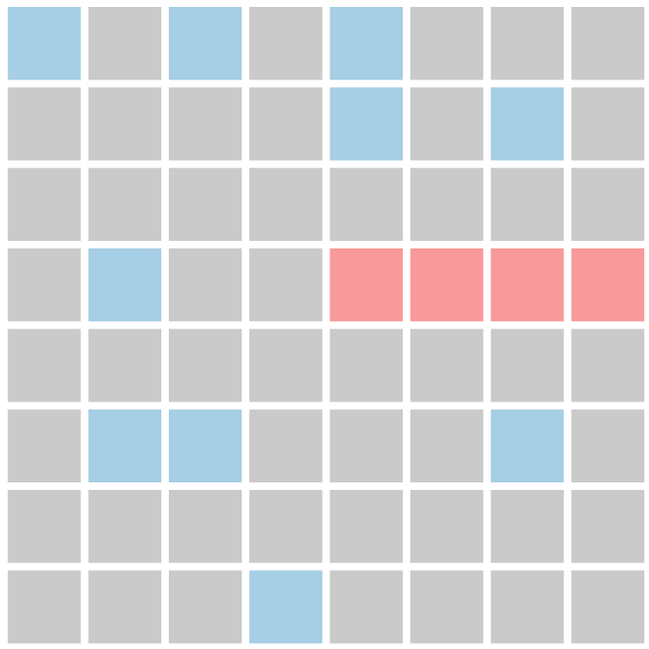


EVICT+TIME Attack

EVALUATION

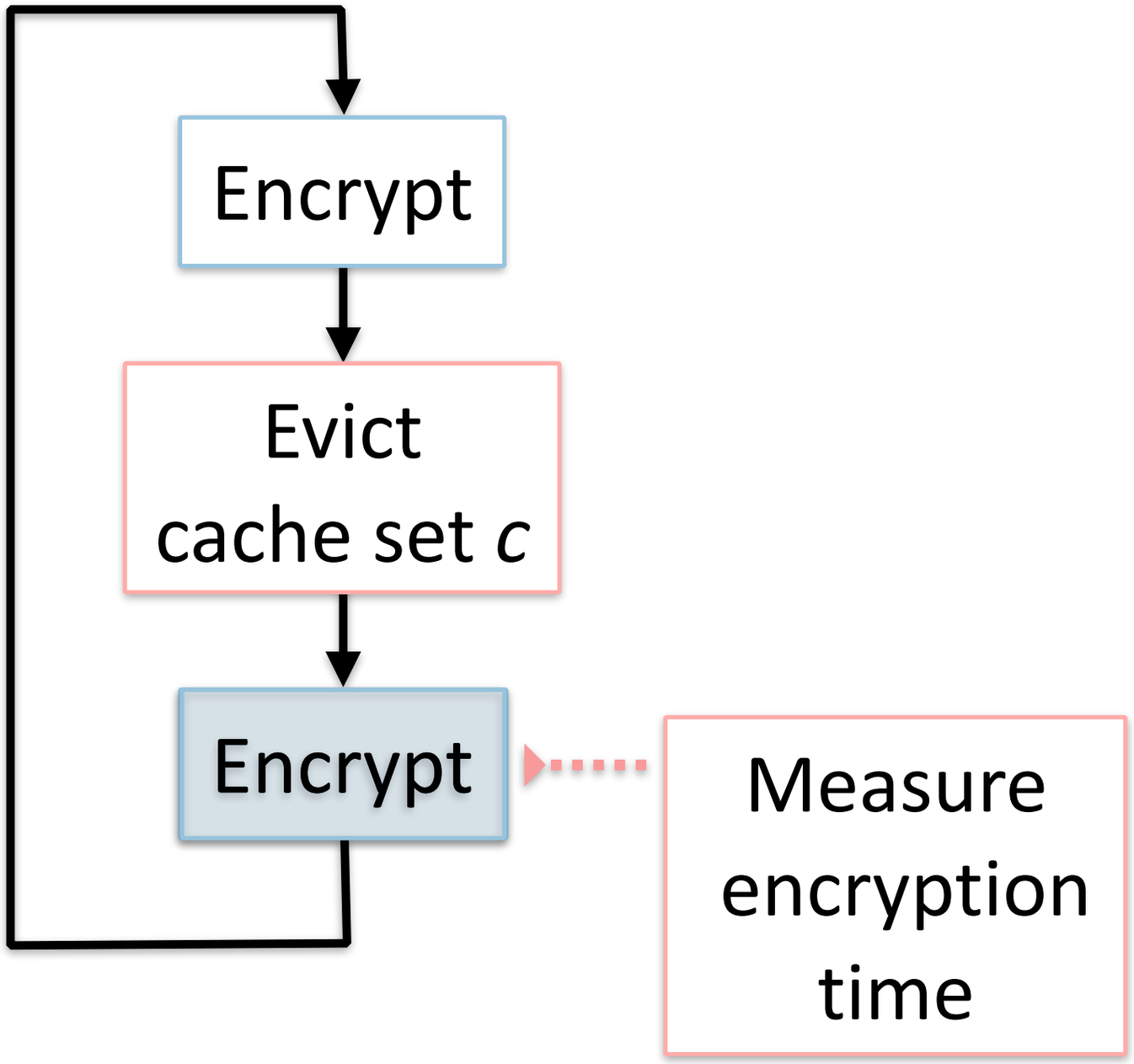


Cache Contents

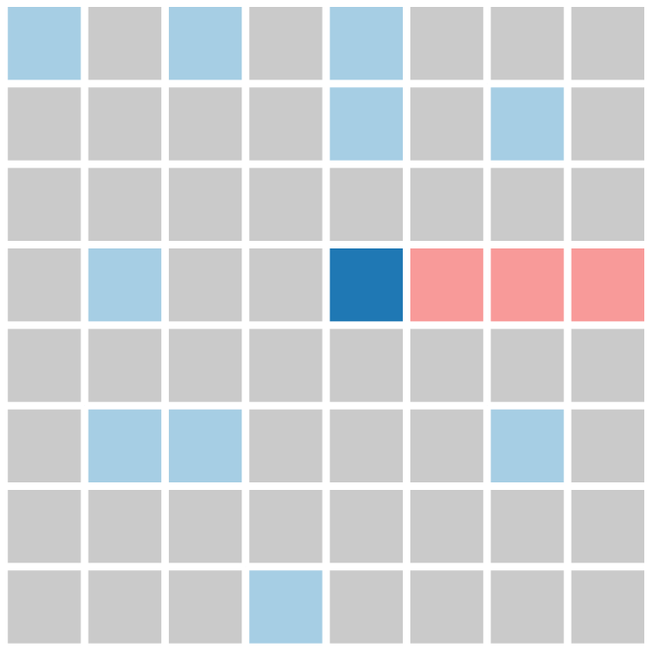


EVICT+TIME Attack

EVALUATION

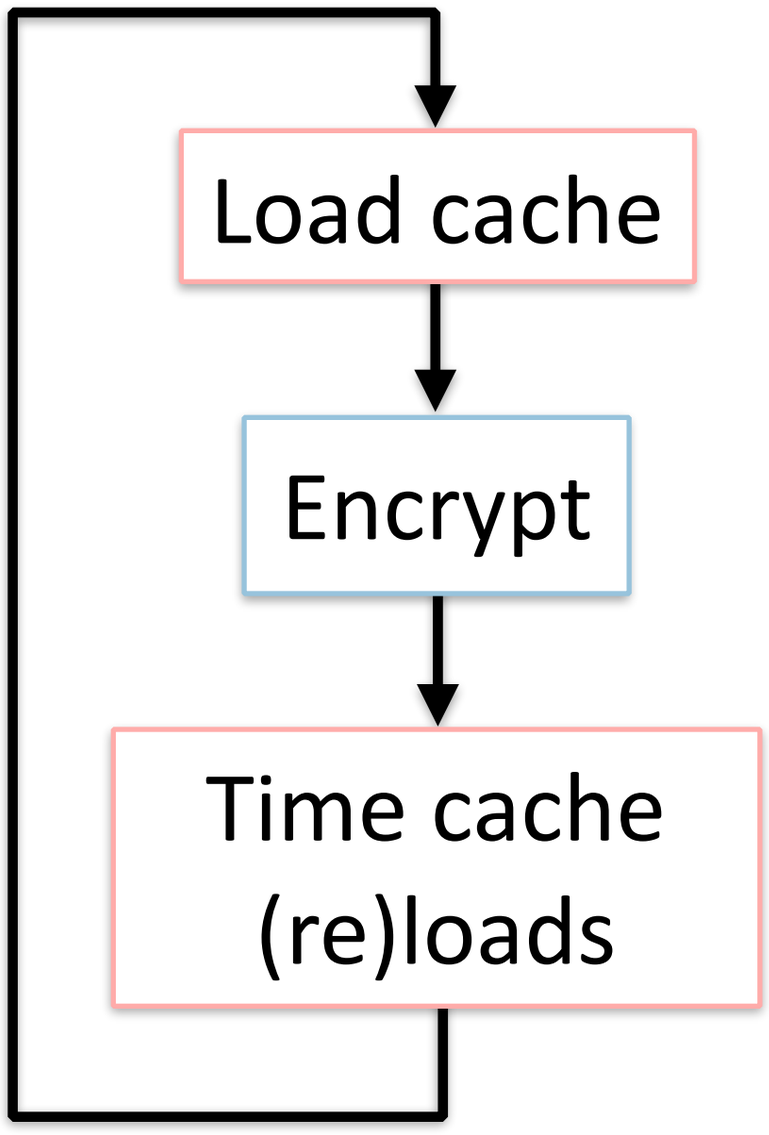


Cache Contents

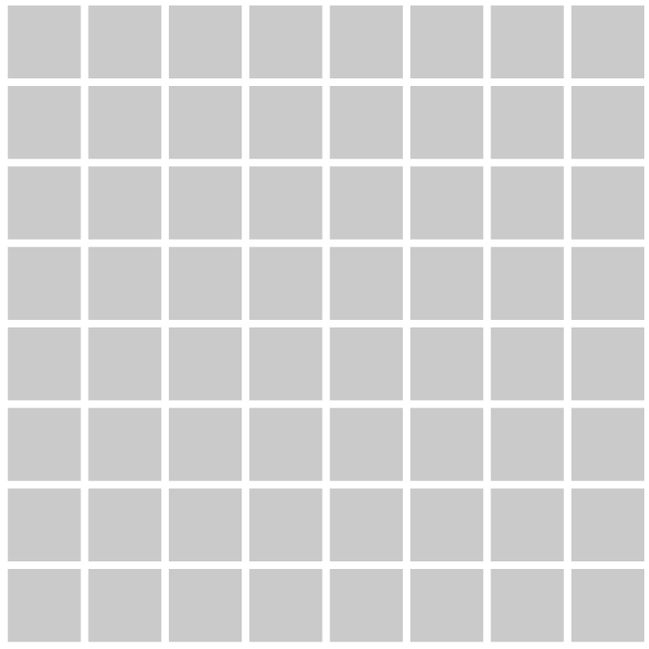


PRIME+PROBE Attack

EVALUATION

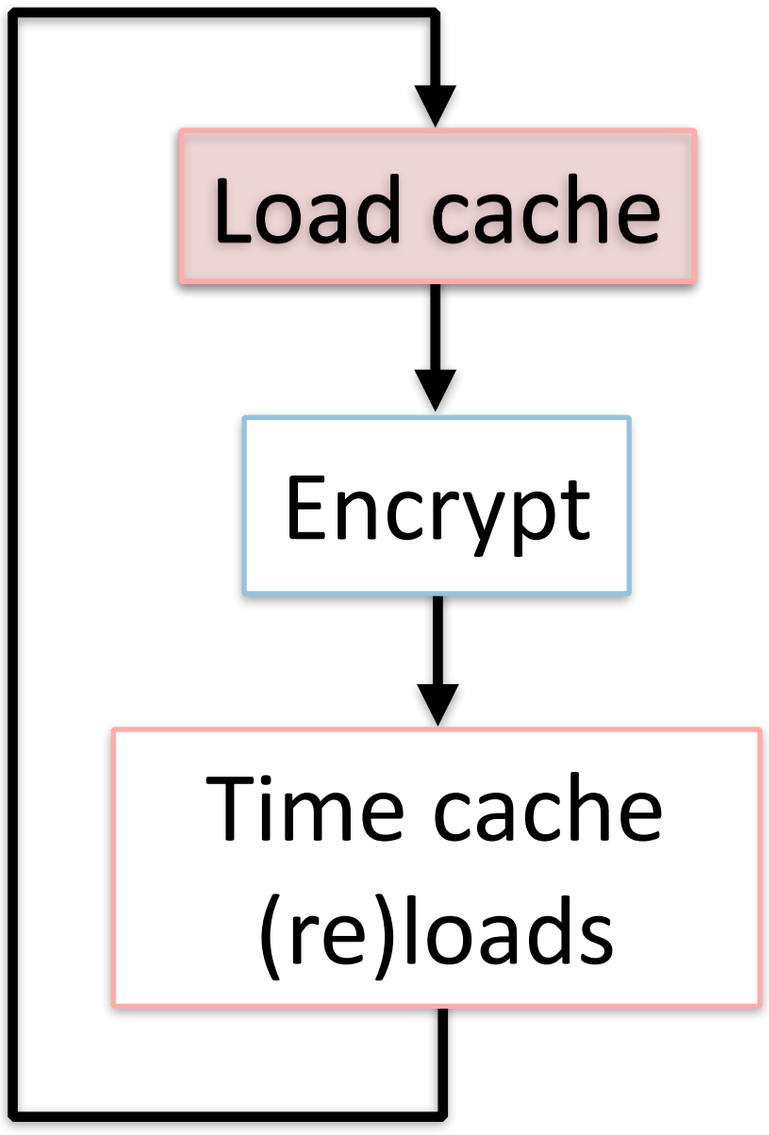


Cache Contents

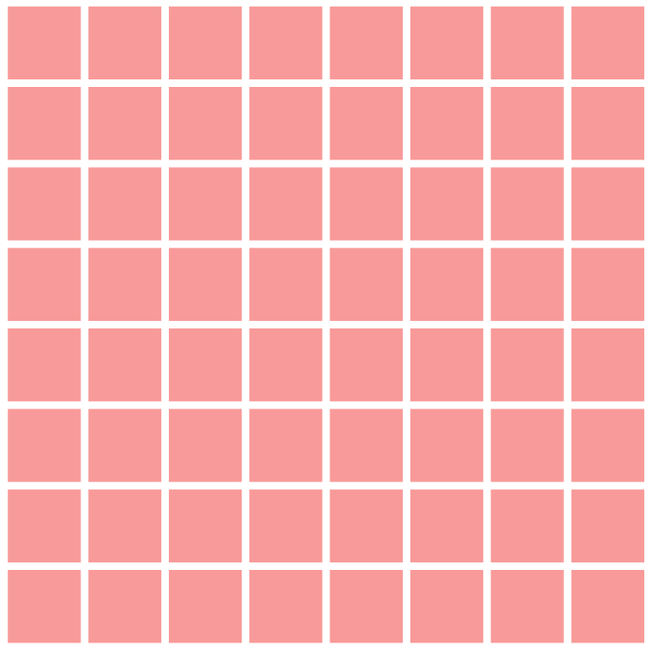


PRIME+PROBE Attack

EVALUATION

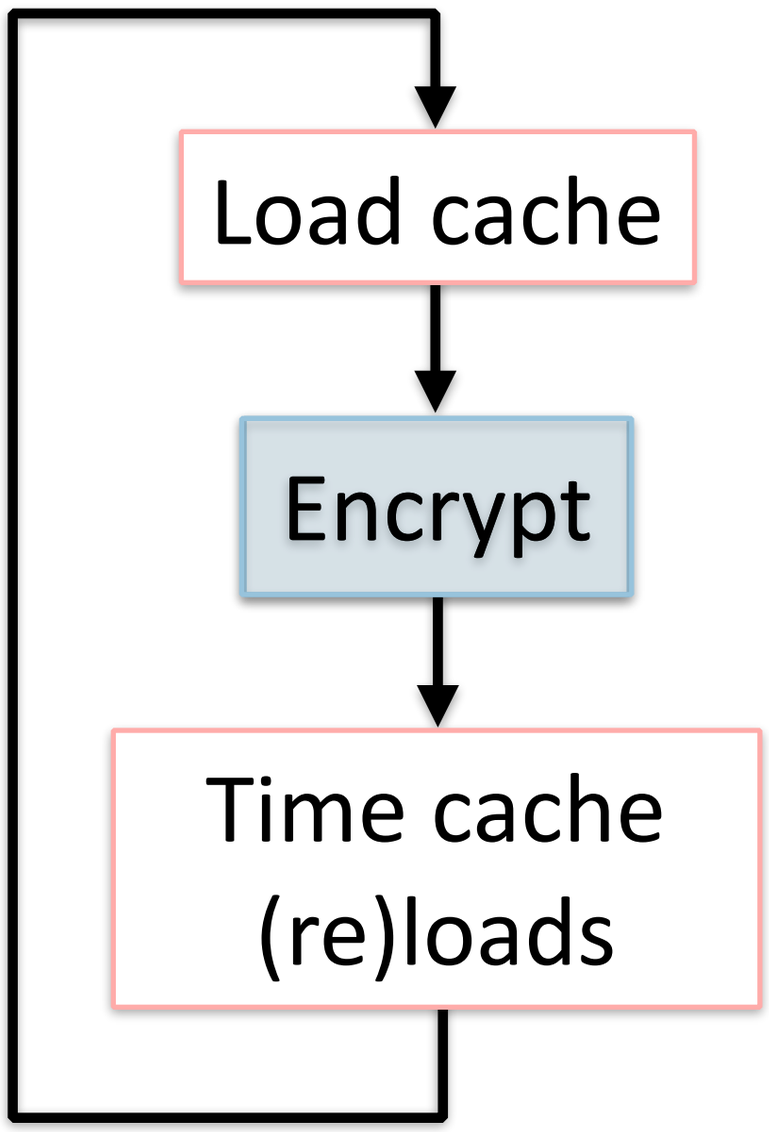


Cache Contents

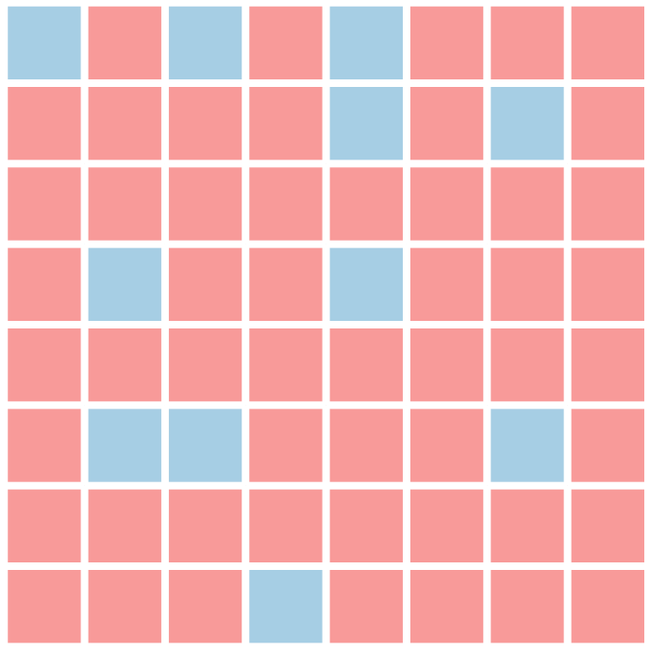


PRIME+PROBE Attack

EVALUATION

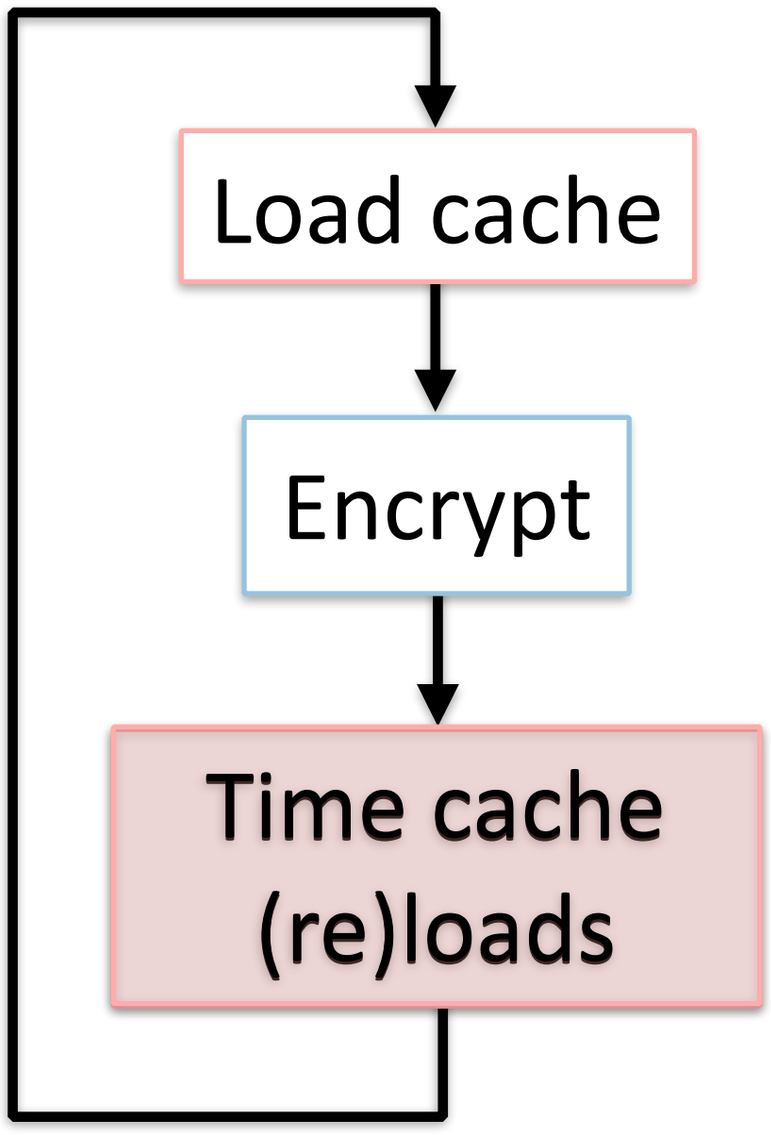


Cache Contents

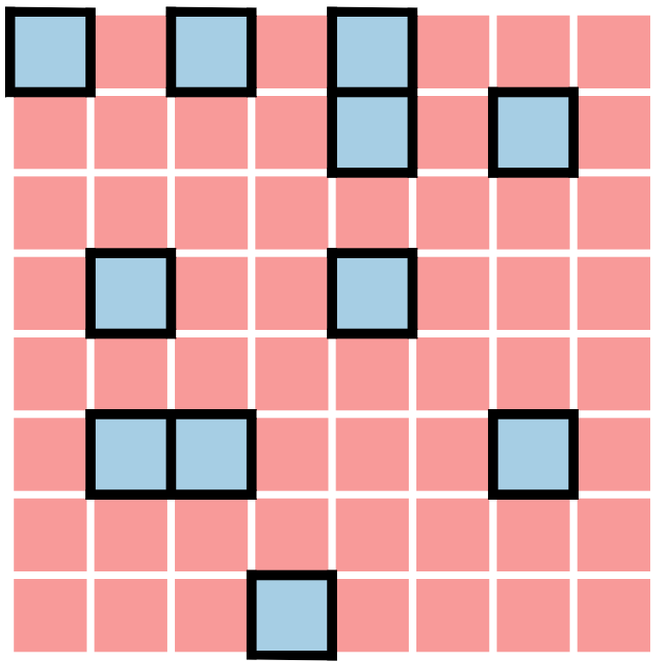


PRIME+PROBE Attack

EVALUATION



Cache Contents



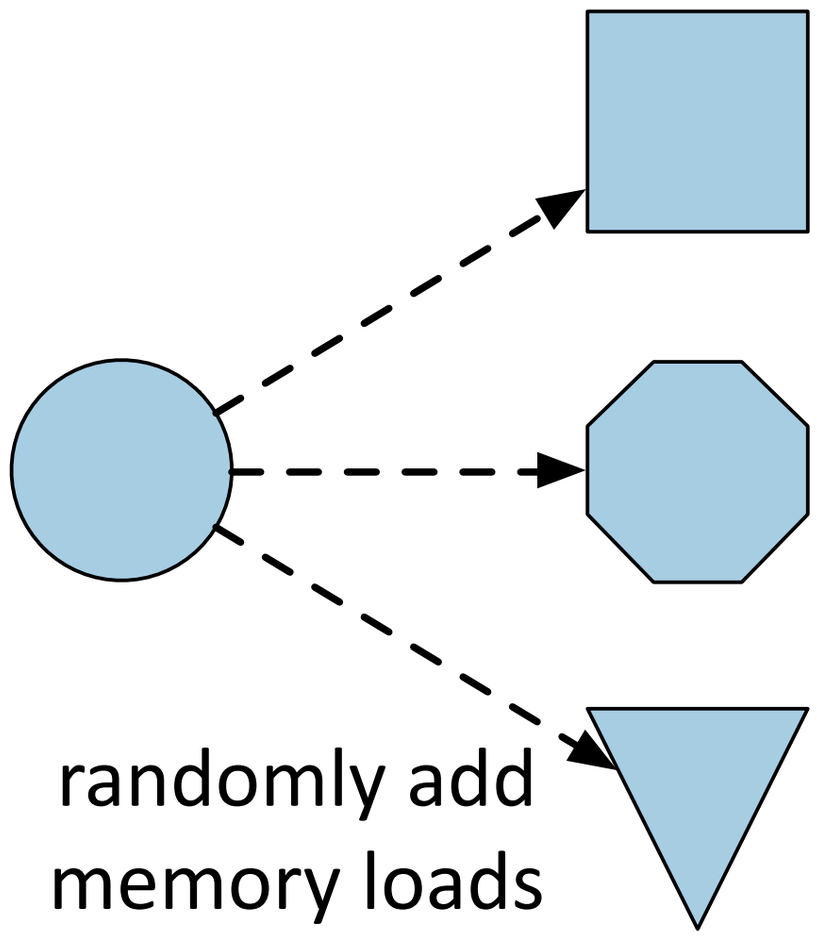
slow load

fast load

Cache Noise Diversity

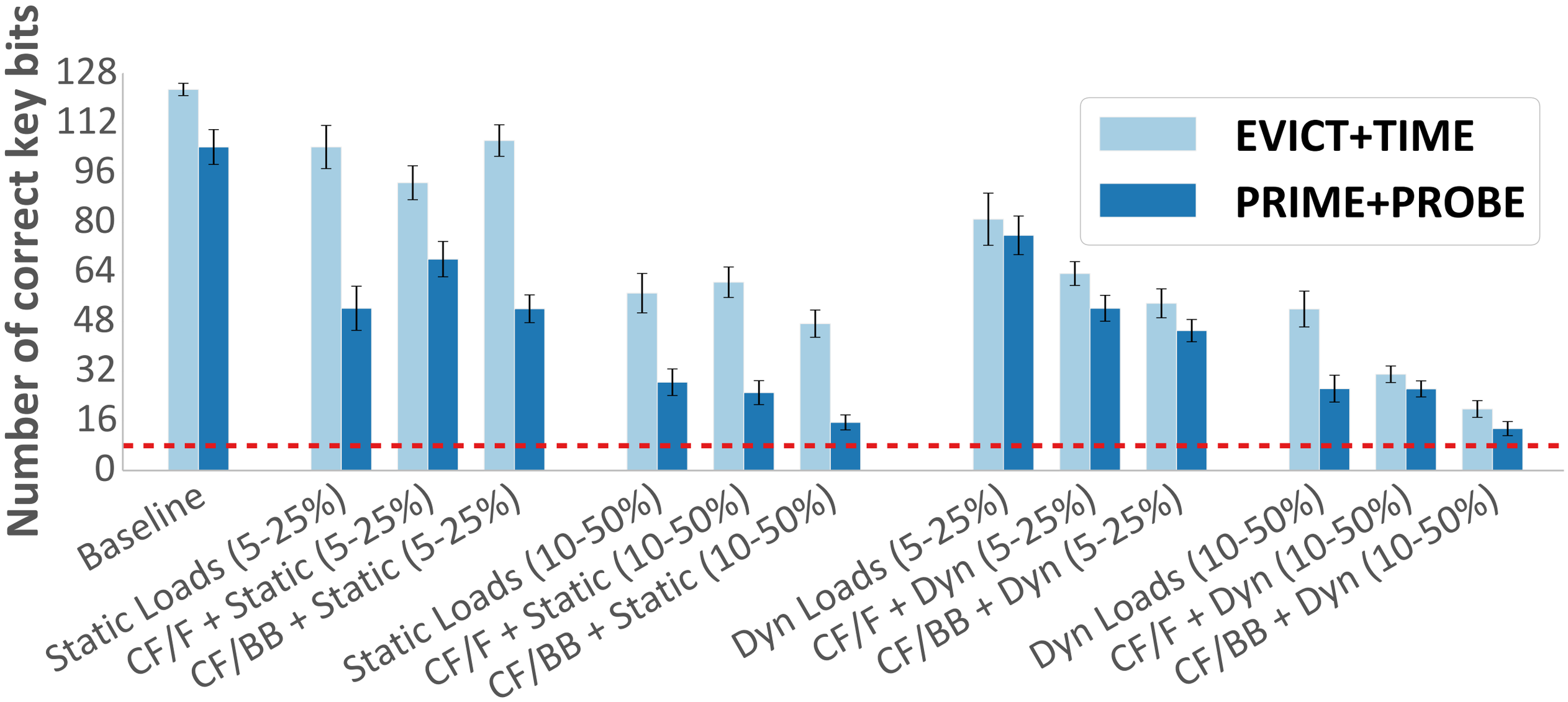
EVALUATION

- Attacks observe cache usage
- We alter cache behavior by randomly adding memory loads
- Tested two memory load variants: static & dynamic
  - both overwrite AES S-box cache lines



Security

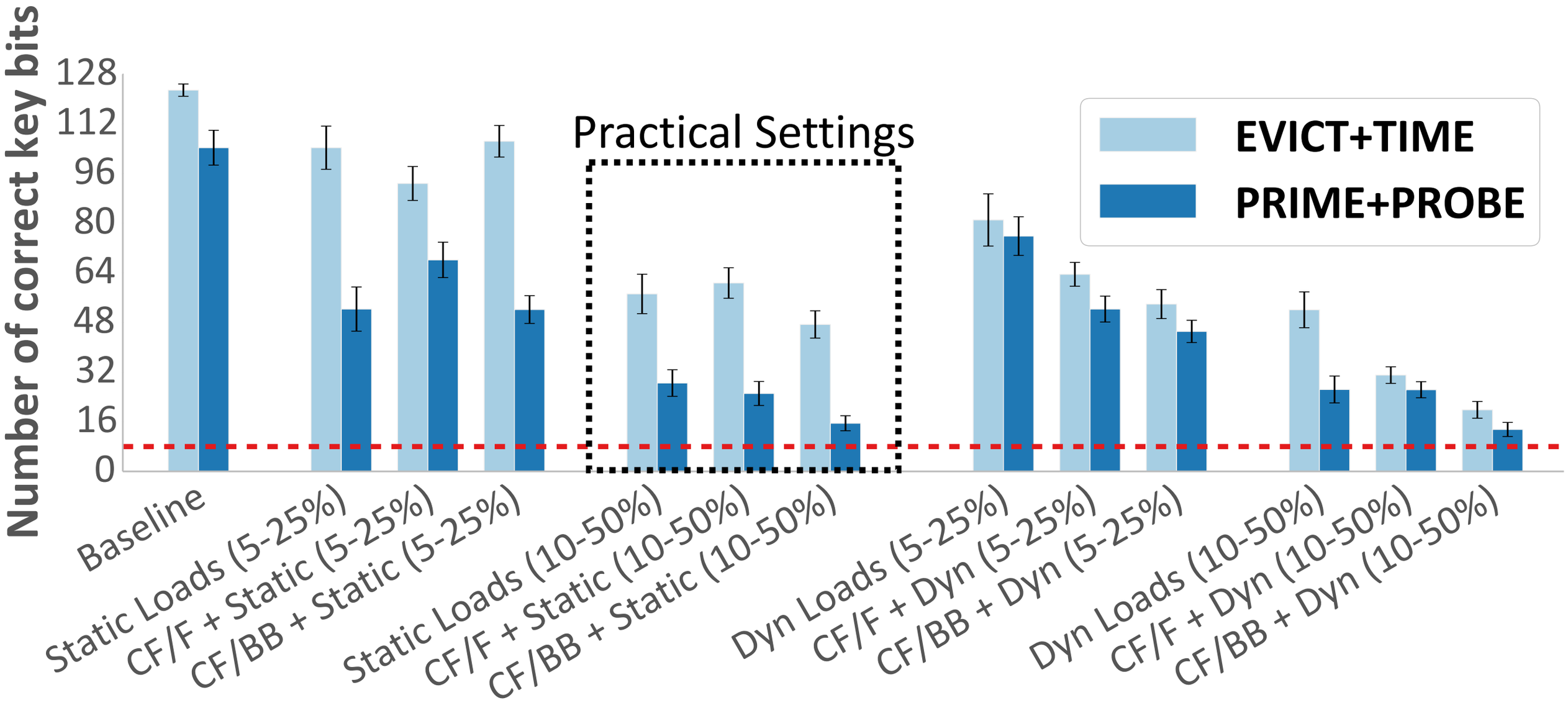
EVALUATION



Dashed red line indicates the expected success of an attacker with no side-channel information.

Security

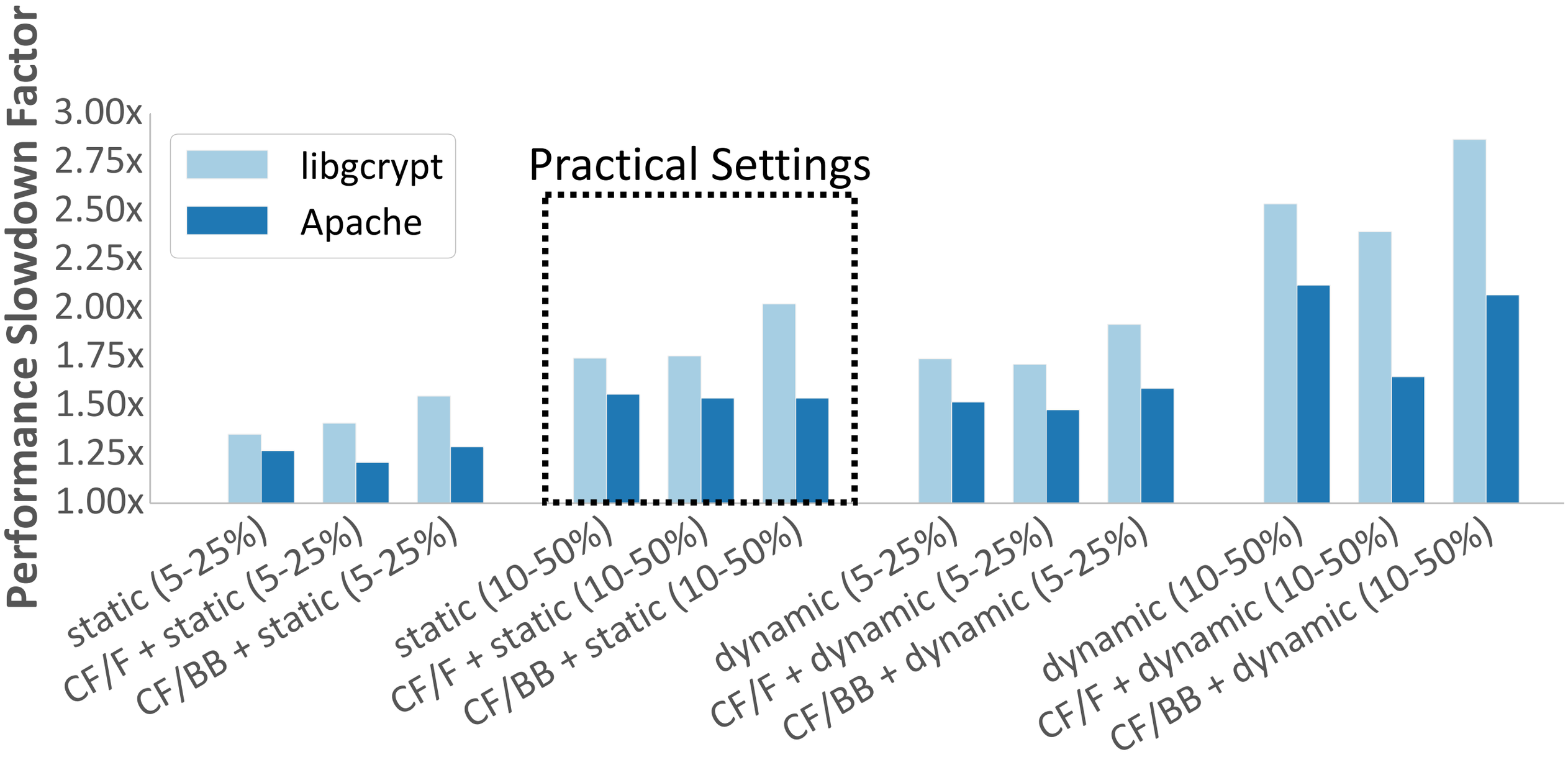
EVALUATION



Dashed red line indicates the expected success of an attacker with no side-channel information.

Performance

EVALUATION



1.5x – 2.0x for practical configurations

Conclusion

- **Generic** technique for **dynamic runtime diversity**
- Dynamic control-flow diversity significantly reduces side-channel leakage
  - reasonable overhead
  - **no developer effort**

Questions?

Stephen Crane
sjcrane@uci.edu