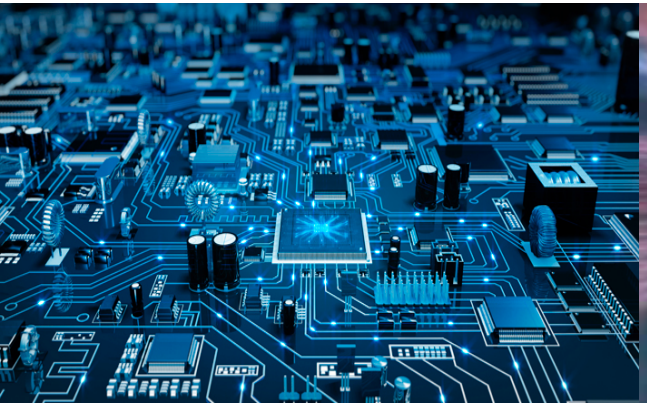
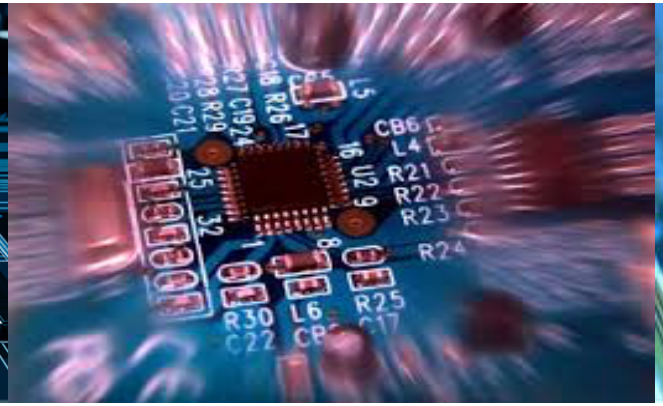
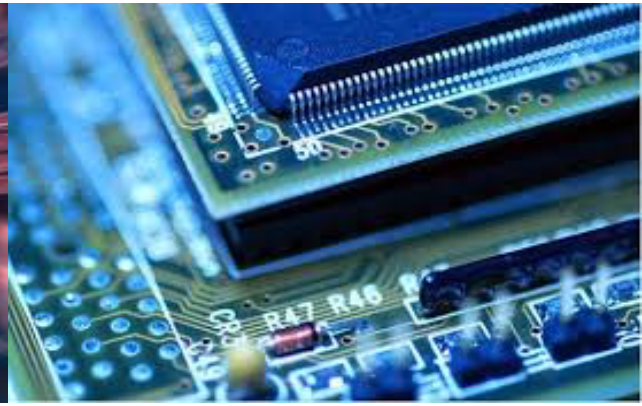
Building Trustworthy System with Untrusted Components:

Countermeasures against Hardware Trojans Collusion

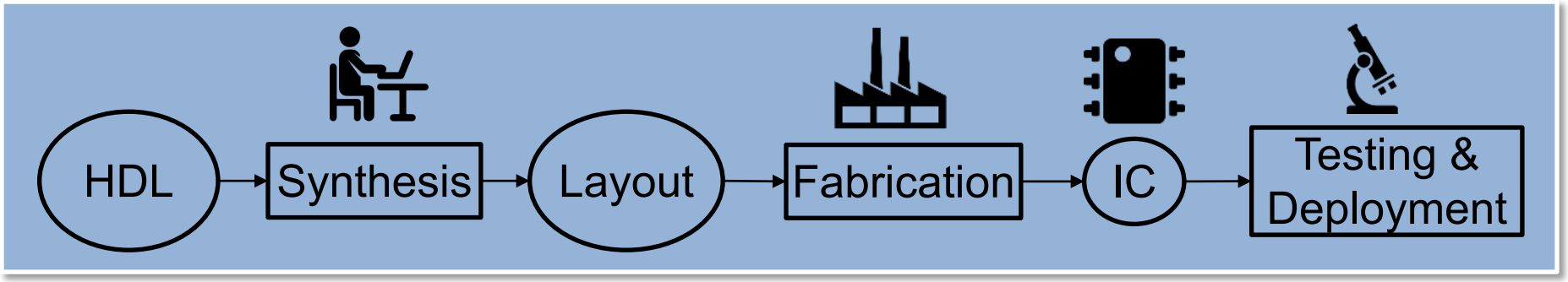
Chengmo Yang

5/13/20

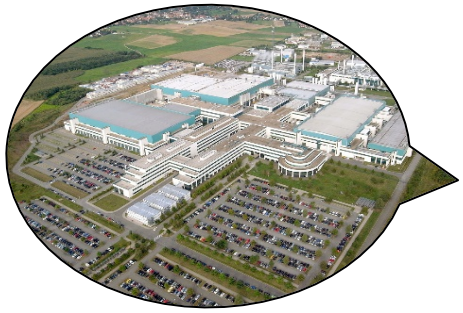
Outline

- Introduction
  - Hardware Trojan collusion problem
  - Design-for-trust & runtime monitoring
- Case studies
  - Trojan collusion in MPSoC
  - Trojan collusion in CPS
- Summary

Traditional IC firm

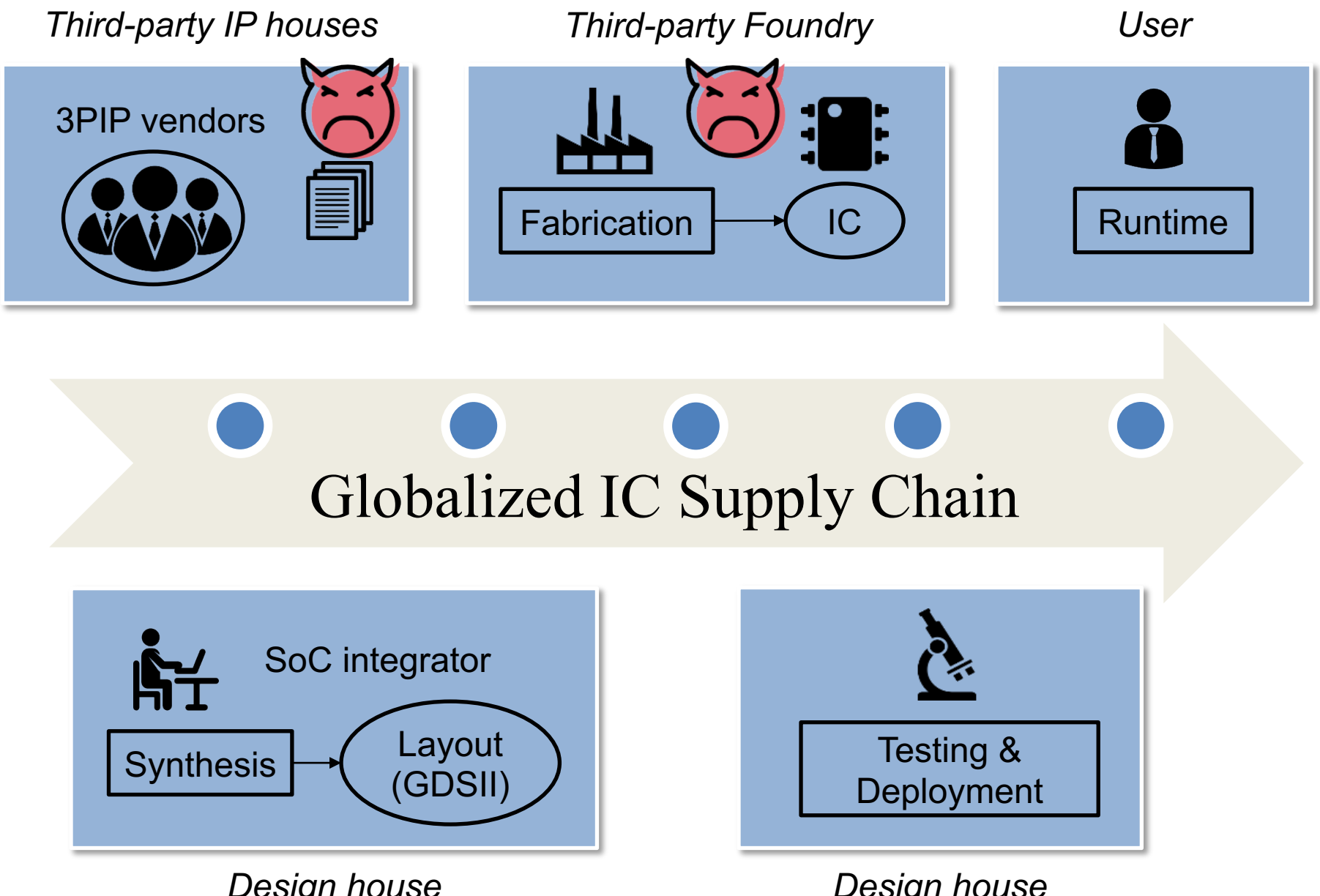


IC Design Flow (Old fashioned)



Two new trends:

- Time-to-market becomes more critical
- Fab becomes super expensive! (GlobalFoundries Fab1 > $7billion)


Third-party IP houses
3PIP vendors
Third-party Foundry
Fabrication
IC
User
Runtime
Globalized IC Supply Chain
SoC integrator
Synthesis
Layout (GDSII)
Testing & Deployment
Design house
Design house

Third-party IP houses

3PIP vendors

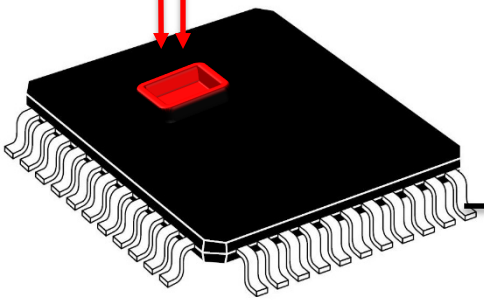
Third-party Foundry

Fabrication

IC

User

Runtime

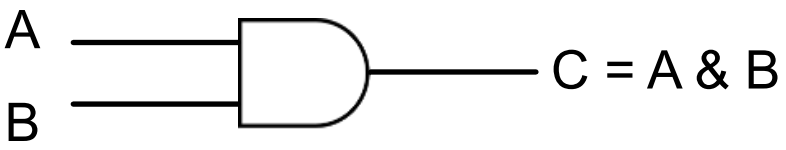


Hardware Trojans:

Unwanted malicious IC modification

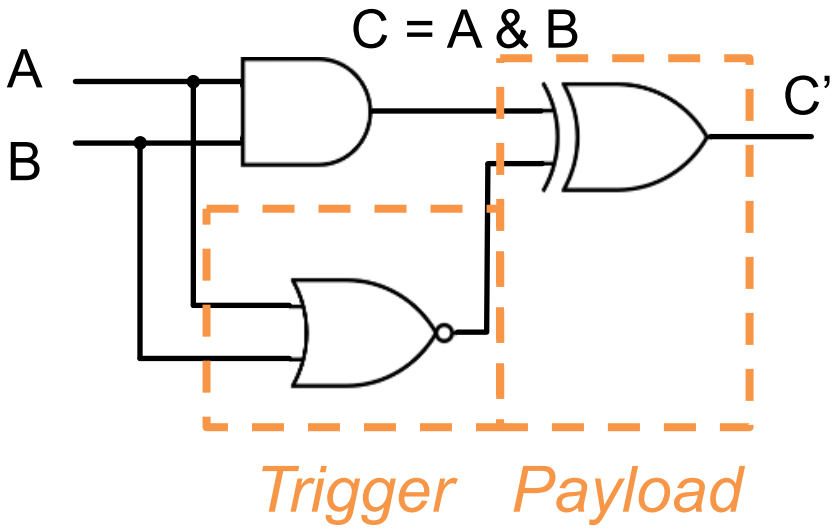
A simple hardware Trojan example

Original circuit

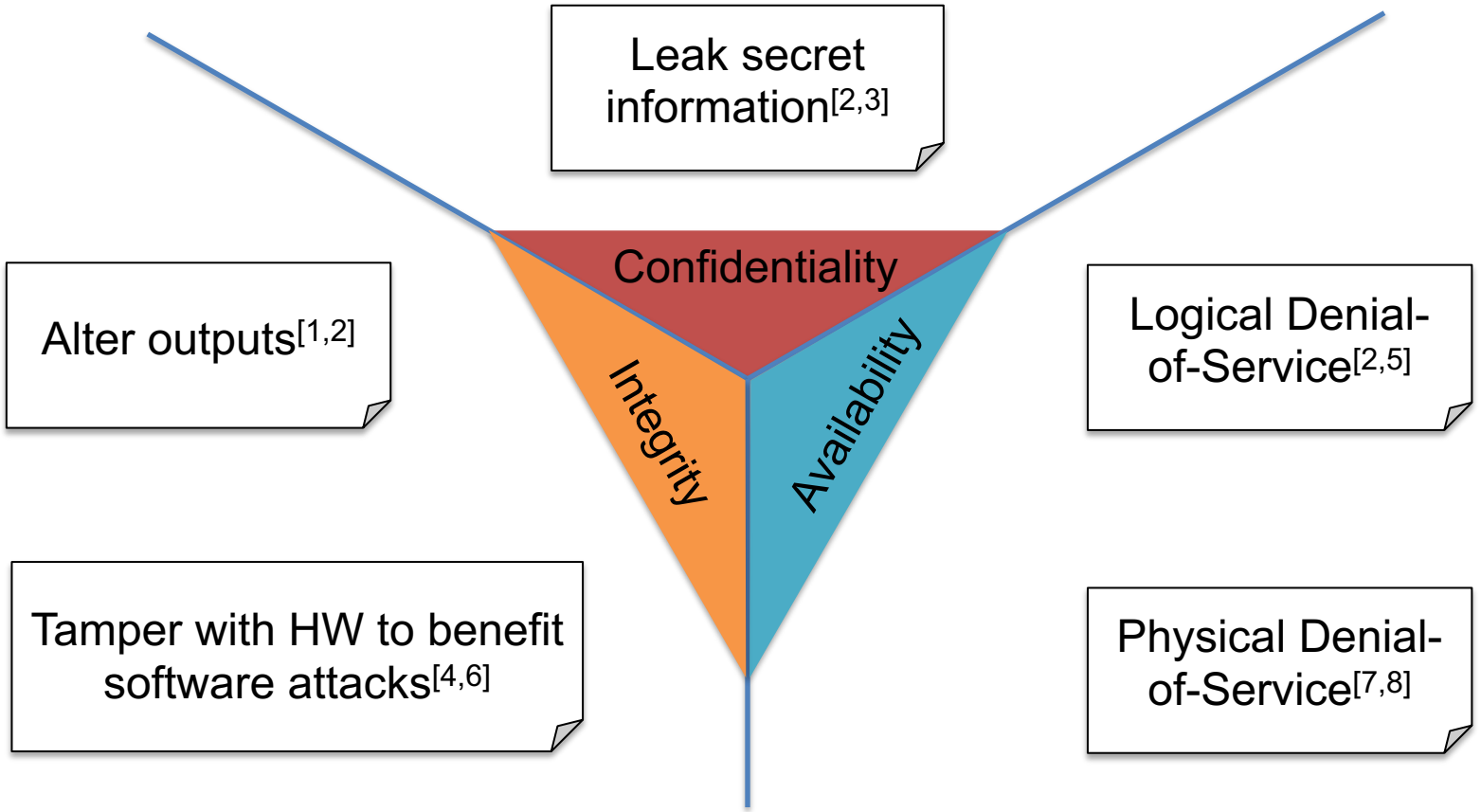


A	B	C	C’
0	0	0	1
0	1	0	0
1	0	0	0
1	1	1	1

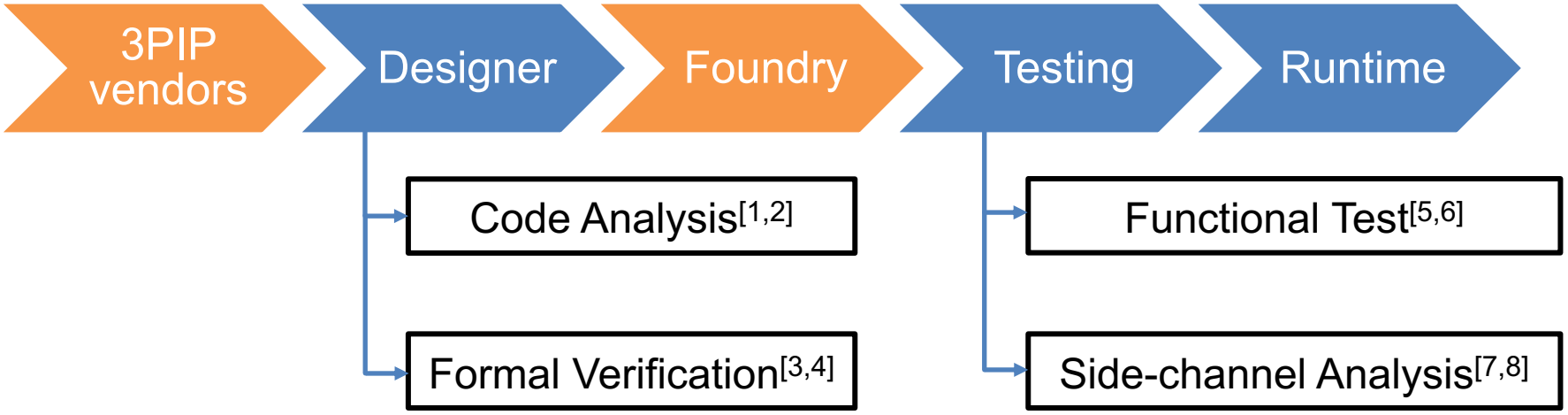
Circuit with hardware Trojan[1]



[1] R. S. Chakraborty, S. Narasimhan and S. Bhunia, "Hardware Trojan: Threats and emerging solutions," *HLDVT’ 09*



[1] Y. Jin and Y. Makris, "Hardware Trojan detection using path delay fingerprint," HOST' 08
[2] Y. Jin, et al., "Experiences in Hardware Trojan design and implementation," HOST' 09
[3] J. Clark, et al., "Hardware Trojan Horse Device Based on Unintended USB Channels," NSS 09
[4] R. Karri, et al., "Trustworthy Hardware: Identifying and Classifying Hardware Trojans," in Computer, 2010 Oct
[5] J. Rajendran, et al., "Towards a comprehensive and systematic classification of hardware Trojans," ISCAS' 10
[6] C. Sturton, et al., "Defeating UCI: Building Stealthy and Malicious Hardware," S&P' 11
[7] R. S. Chakraborty, et al., "Hardware Trojan: Threats and emerging solutions," HLDVT' 09
[8] Y. Shiyanovskii, et al., "Process reliability based trojans through NBTI and HCI effects," AHS' 10



[1] H. Salmani and M. Tehranipoor, "Analyzing circuit vulnerability to hardware Trojan insertion at the behavioral level," DFTs' 13

[2] A. Waksman, et al, "FANCI: Identification of Stealthy Malicious Logic Using Boolean Functional Analysis," CCS' 13

[3] E. Love, et al, "Proof-Carrying Hardware Intellectual Property: A Pathway to Trusted Module Acquisition," IEEE Transaction on Information Forensics and Security, 2012

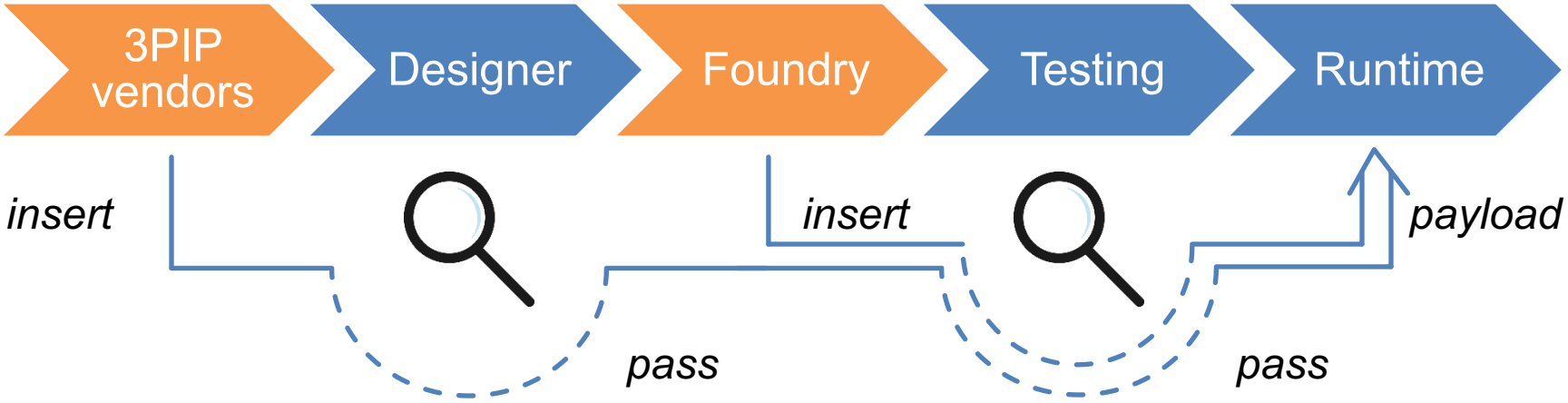
[4] X. Zhang and M. Tehranipoor, "Case study: Detecting hardware Trojans in third-party digital IP cores," HOST' 11

[5] S. Jha and S. K. Jha, "Randomization Based Probabilistic Approach to Detect Trojan Circuits," HASE' 08

[6] R. S. Chakraborty, et al., "MERO:A Statistical Approach for Hardware Trojan Detection," CHES' 09

[7] R. Rad, et al., "A Sensitivity Analysis of Power Signal Methods for Detecting Hardware Trojans Under Real Process and Environmental Conditions," in IEEE Transaction on VLSI, 2010

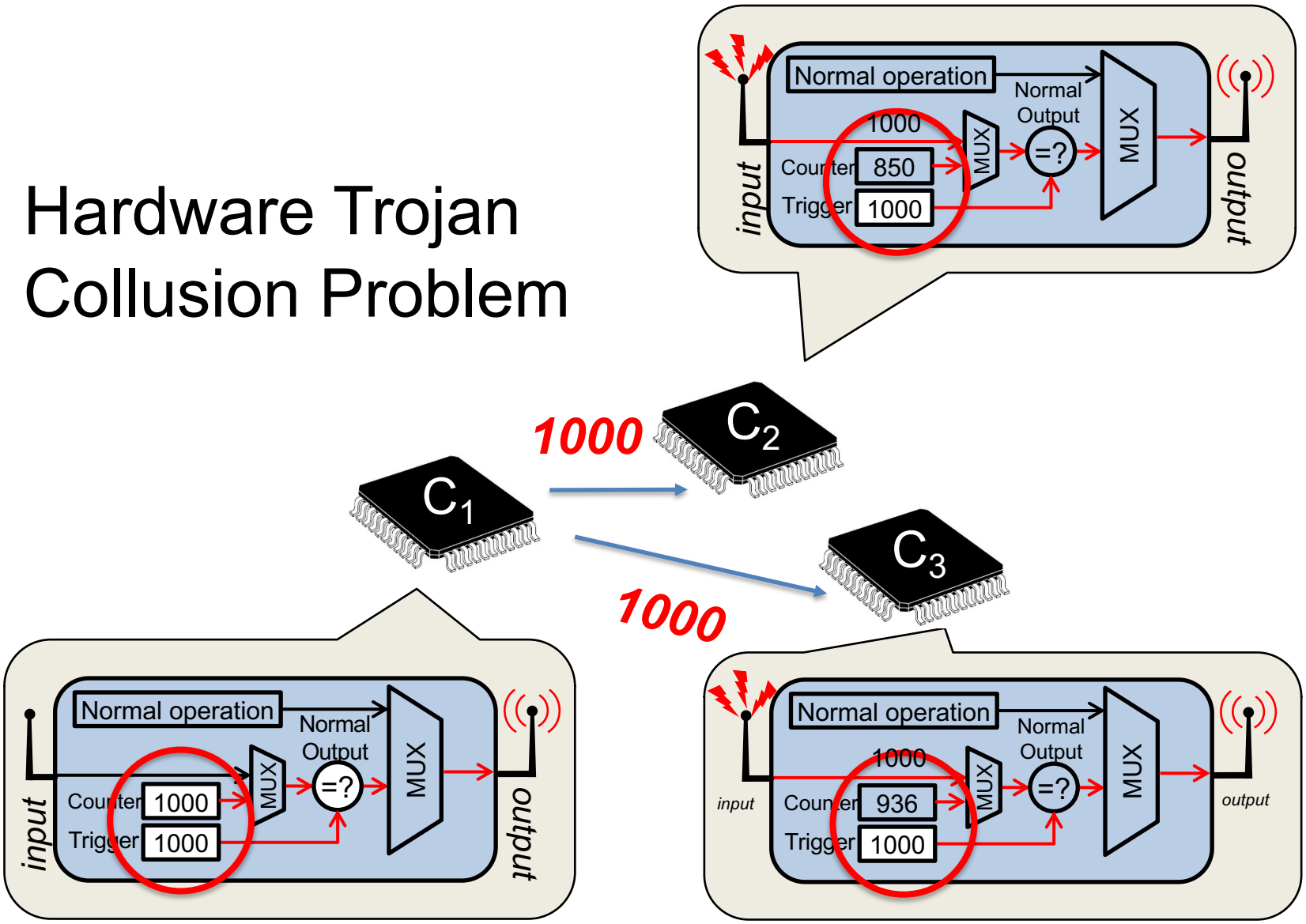
[8] S. Narasimhan, et al. "Multiple-parameter side-channel analysis: A noninvasive hardware Trojan detection approach," HOST '10



How to make Trojan hard to discover?

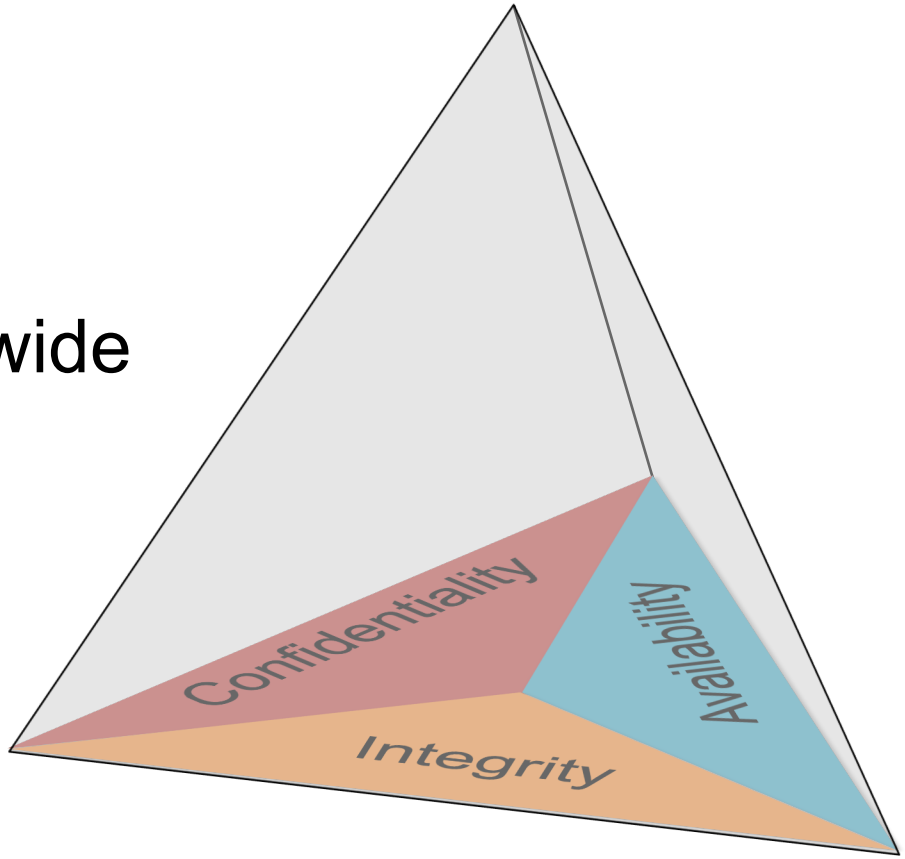
- Small in size
- Located at rarely used areas
- Conditionally triggered

Hardware Trojan Collusion Problem



Collusion (between Trojans)

- An orthogonal dimension
- Cause system-wide damage



Single Trojan Payloads

Problem:

- Defend against hardware Trojan Collusion

Goals:

- Prevent and detect Trojan collusion at runtime
- Build trustworthy system with untrusted components

Idea:

- Integrate runtime monitoring and design-for-trust.

Offline stage

Design for Trust (DfT)

- Define rules to follow at runtime
- Help distinguish legal/illegal behaviors

Online stage

Runtime monitoring

- "Last line of defense" against hardware Trojan
- Aims to detect Trojan when it is ***active***

Runtime monitoring only:

monitor

runtime monitor

DfT+ runtime monitoring:

security rules

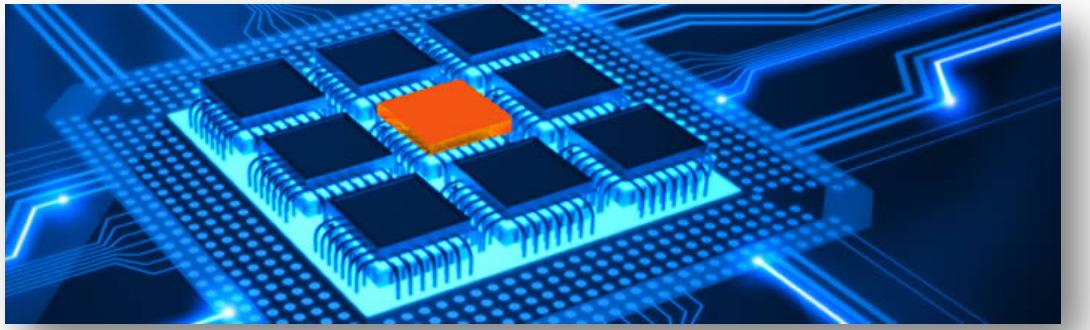
monitor

runtime monitor

Two case studies

Hardware Trojan collusion in:

Multiprocessor System-on-Chips (MPSoCs)

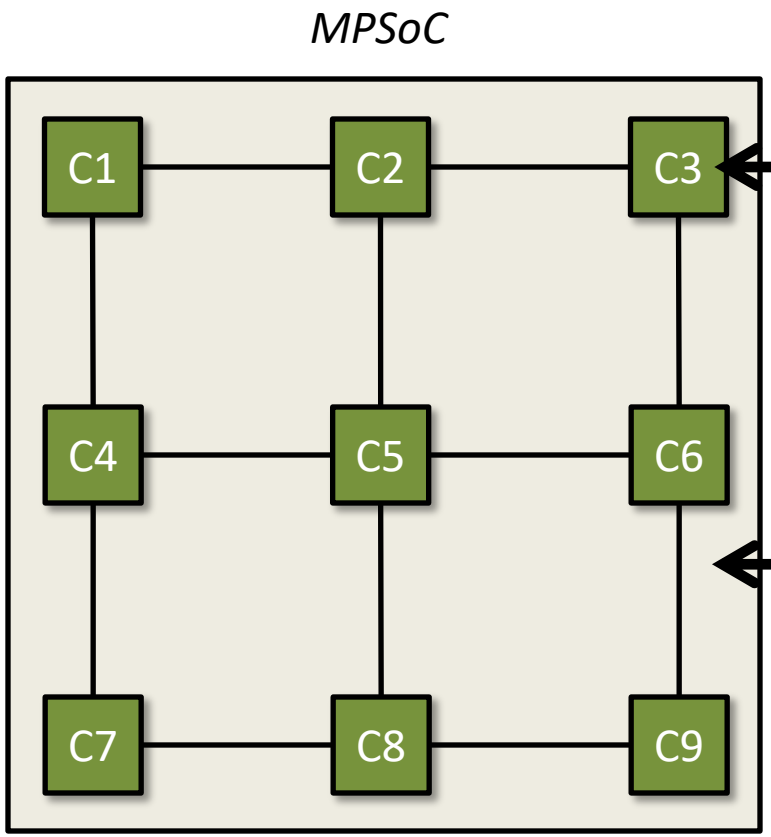


Cyber-Physical Systems (CPS)

Outline

- Introduction
  - Hardware Trojan collusion problem
  - Design-for-trust & runtime monitoring
- Case studies
  - Trojan collusion in MPSoC
  - Trojan collusion in CPS
- Summary

Threat model

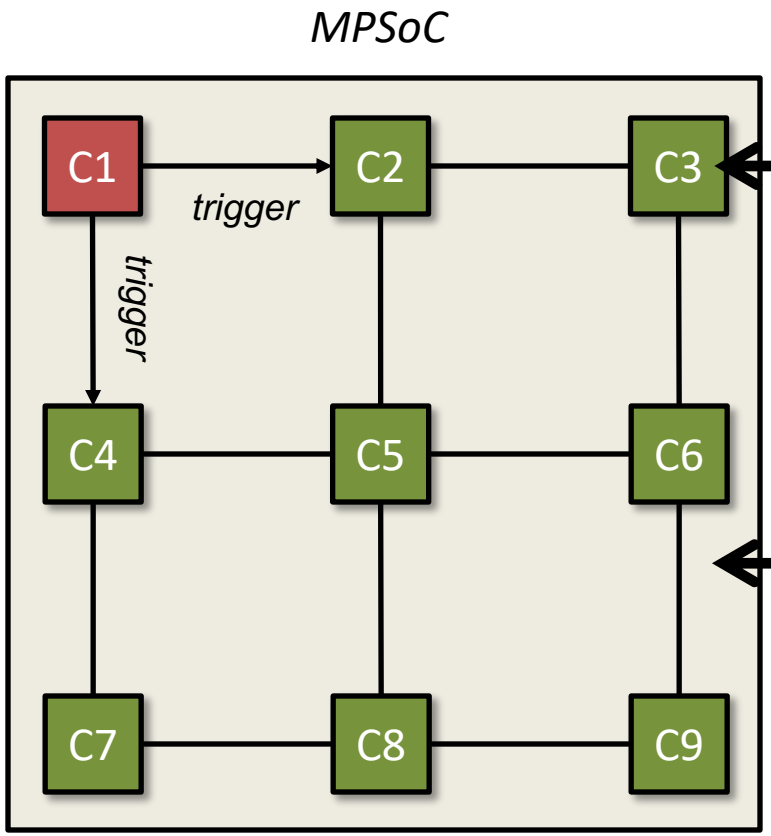


Trojan in each 3PIP core may be self-triggered

Trojans distributed on different 3PIP cores may trigger each other to **collude**

C — Core with hibernating Trojan

Threat model



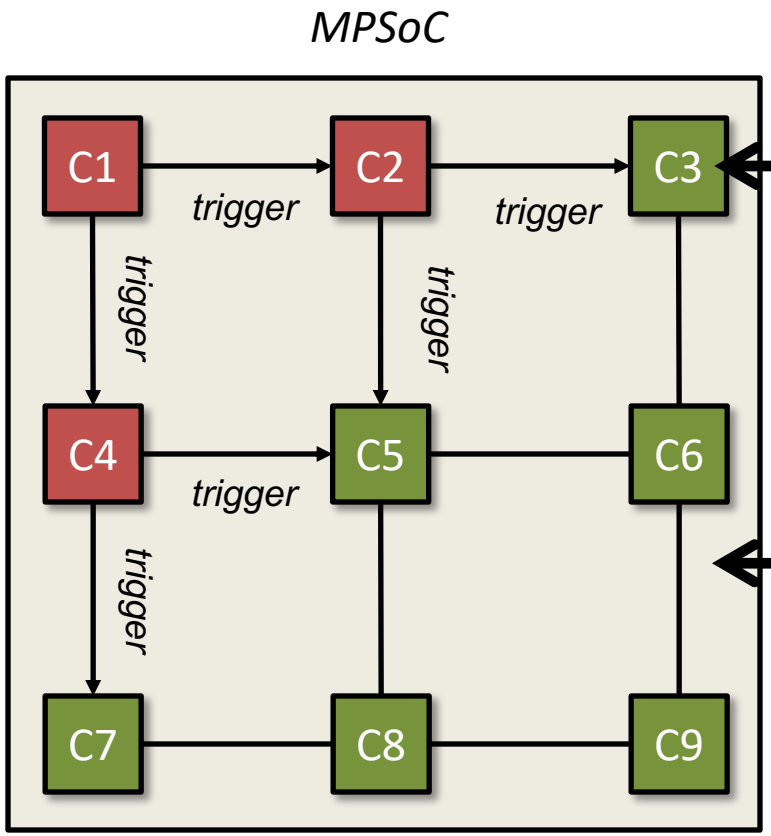
Trojan in each 3PIP core may be self-triggered

Trojans distributed on different 3PIP cores may trigger each other to **collude**

C — Core with hibernating Trojan

C — Core with activated Trojan

Threat model



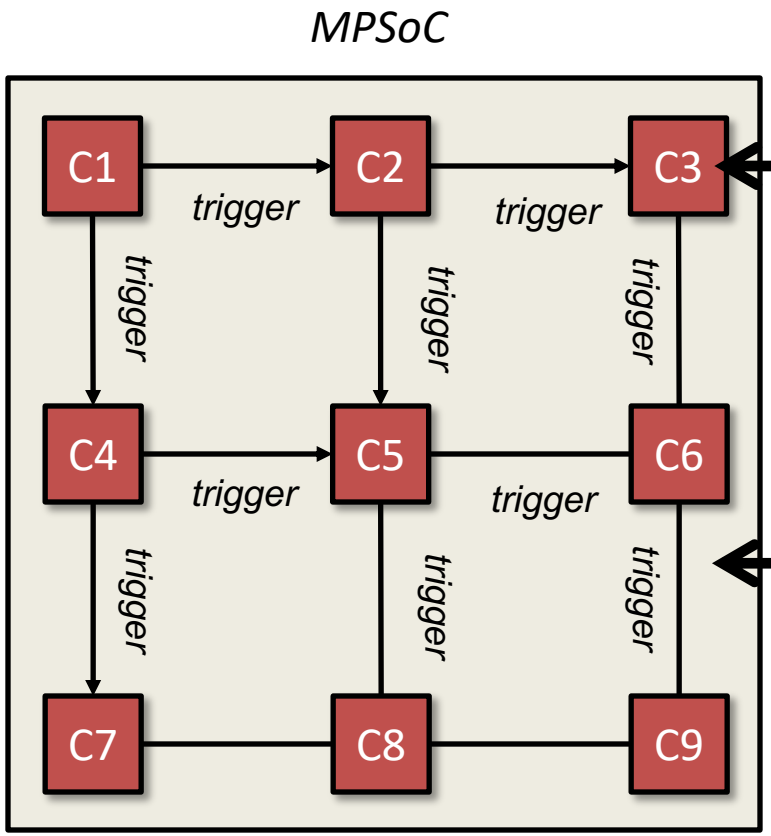
Trojan in each 3PIP core may be self-triggered

Trojans distributed on different 3PIP cores may trigger each other to **collude**

C — Core with hibernating Trojan

C — Core with activated Trojan

Threat model



Trojan in each 3PIP core may be self-triggered

Trojans distributed on different 3PIP cores may trigger each other to **collude**

C — Core with hibernating Trojan

C — Core with activated Trojan

Countermeasure

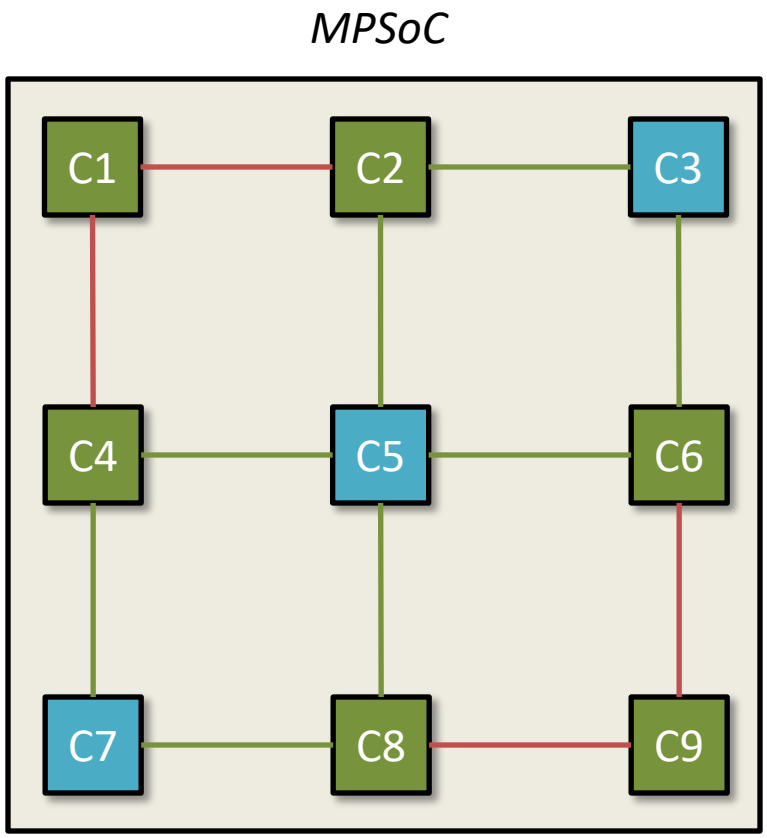
Design-for-Trust

- Vendor diversity
- Security-driven task scheduling

Runtime monitoring

- Check message source and destination

Countermeasure



Vendor diversity:

- Use cores from different 3PIP vendors

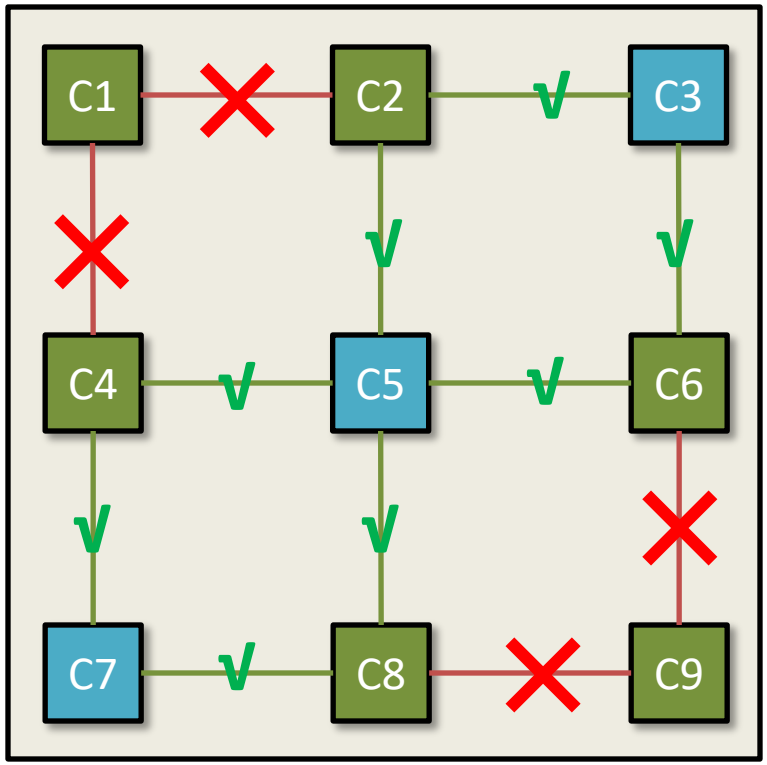
Assumption – cores from different vendors do not have the same Trojan (trigger pattern) (i.e. do not collude)

Core from vendor A

Core from vendor B

Countermeasure

MPSoC



Vendor diversity:

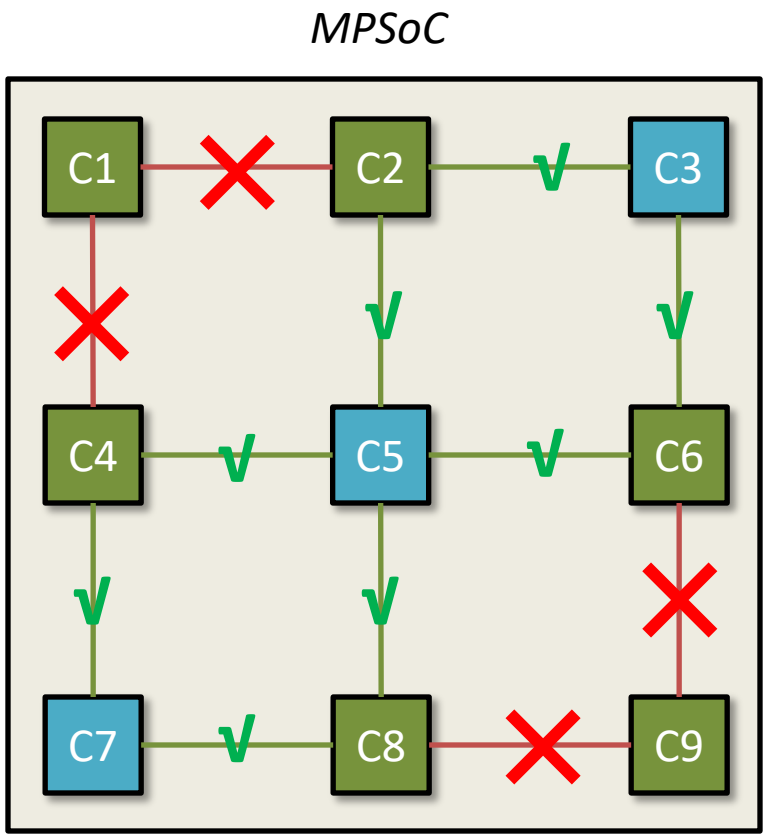
- Use cores from different 3PIP vendors
- Channels between different vendors are safe
- Channels within same vendor are unsafe

Assumption – cores from different vendors do not have the same Trojan (trigger pattern) (i.e. do not collude)

C Core from vendor A

C Core from vendor B

Countermeasure



Security-driven task scheduling:

- NOT map inter-task communications to unsafe channels

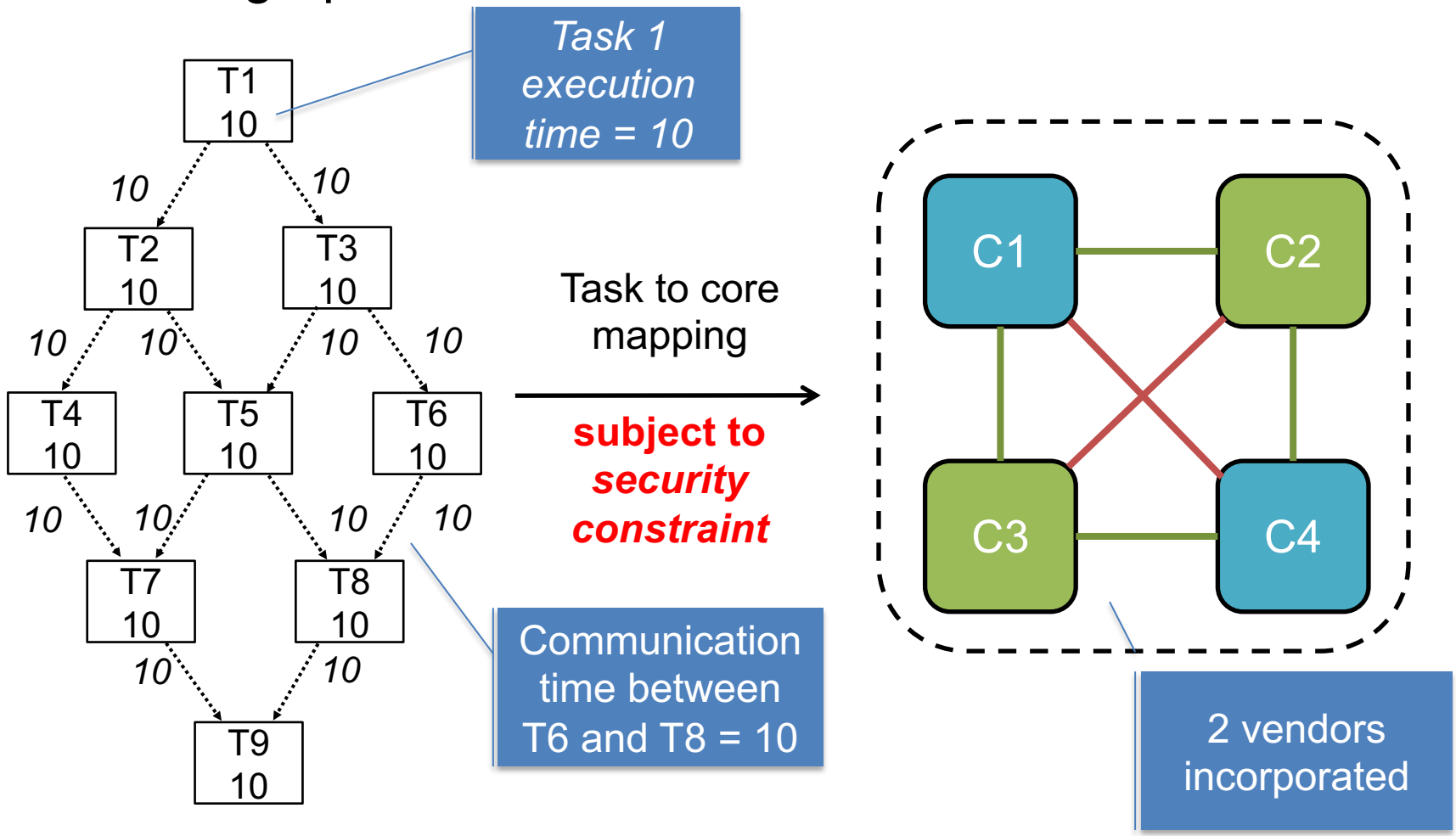
Runtime monitoring:

- Any communication on unsafe channel triggers alarm

C Core from vendor A

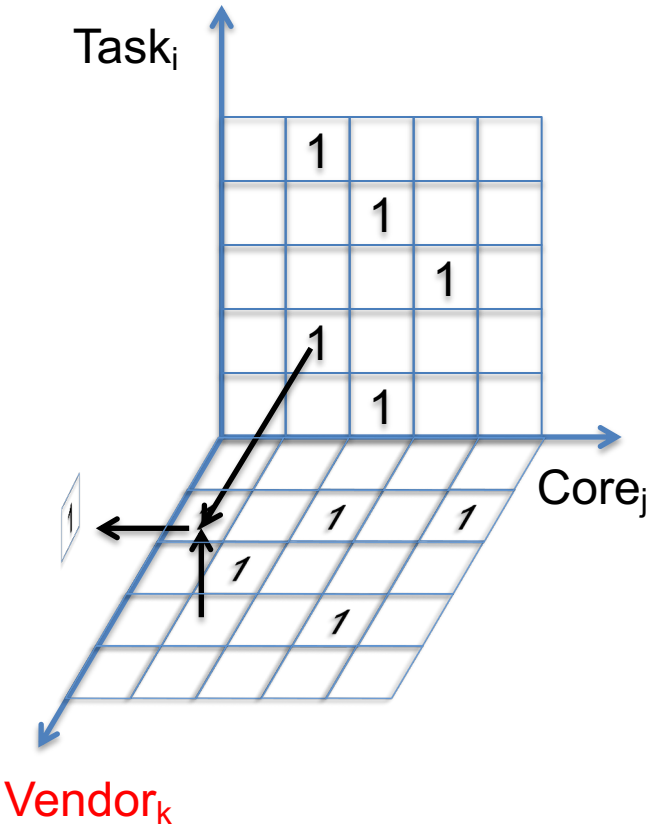
C Core from vendor B

Security-driven task scheduling example



The problem can formulated with *Integer Linear Programming*

ILP Formulation - binding



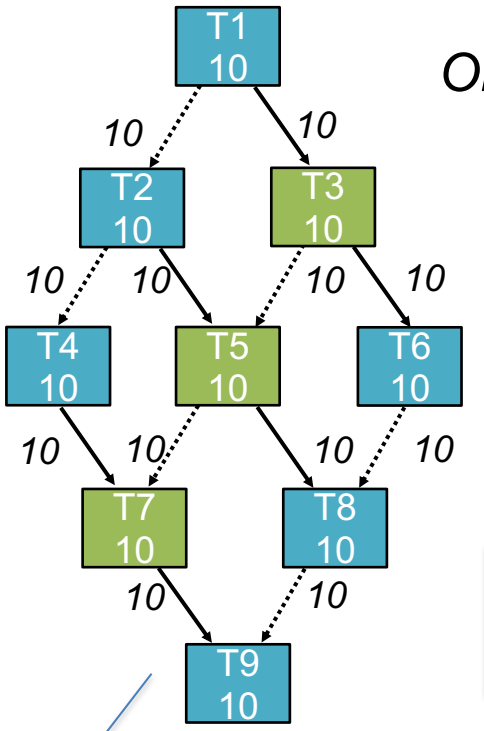
$$T2C_{i,j} = \begin{cases} 1 & \text{Task}_i \text{ on Core}_j \\ 0 & \text{Otherwise} \end{cases} \qquad \sum_{j=1}^{m} T2C_{i,j} = 1$$

$$C2V_{j,k} = \begin{cases} 1 & \text{Core}_j \text{ on Vendor}_k \\ 0 & \text{Otherwise} \end{cases} \qquad \sum_{k=1}^{p} C2V_{j,k} = 1$$

$$T2V_{i,k} = \sum_{j=1}^{m} T2C_{i,j} * C2V_{j,k}$$

ILP Formulation - security constraint

Task graph



Finishing time = 70

$$Order_{i,j} = \begin{cases} 0 & Task_i \text{ and } Task_j \text{ are dependent \& not on the same core} \\ 1 & Otherwise \end{cases}$$

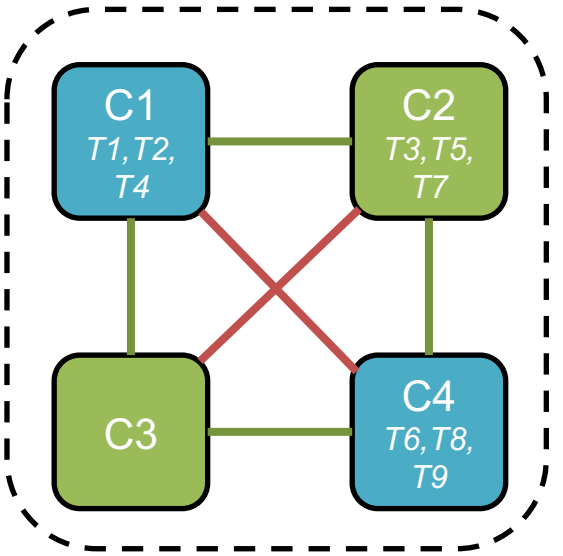
$$T2V_{i,k} + T2V_{j,k} \leq 1 + Order_{i,j}$$

security constraint

i, j: tasks i and j
k: vendor k

No communication is mapped to unsafe channel

MPSoC



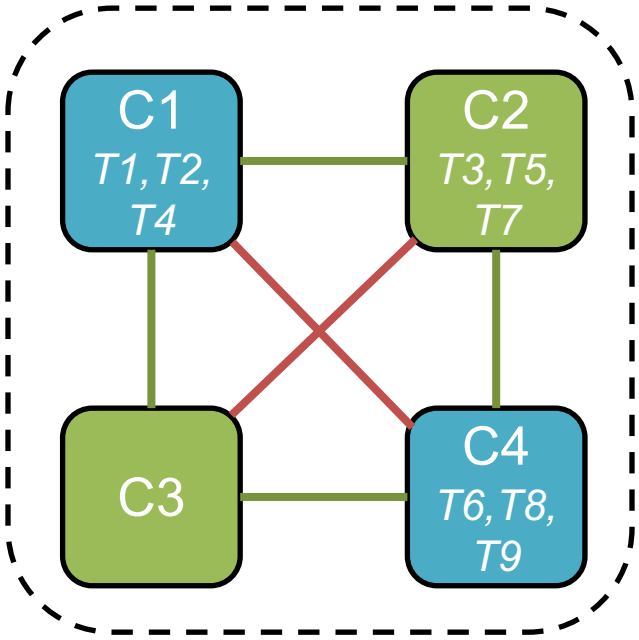
Effectiveness

Runtime monitor rule:

```
If (source.vendor=destination.vendor){
        trigger alarm;
}
```

Comm. type	Channel	Consequence
Task comm.	Safe	Allowed
Task comm.	Unsafe	Precluded
Trojan comm.	Safe	Muted
Trojan comm.	Unsafe	Detected

MPSoC

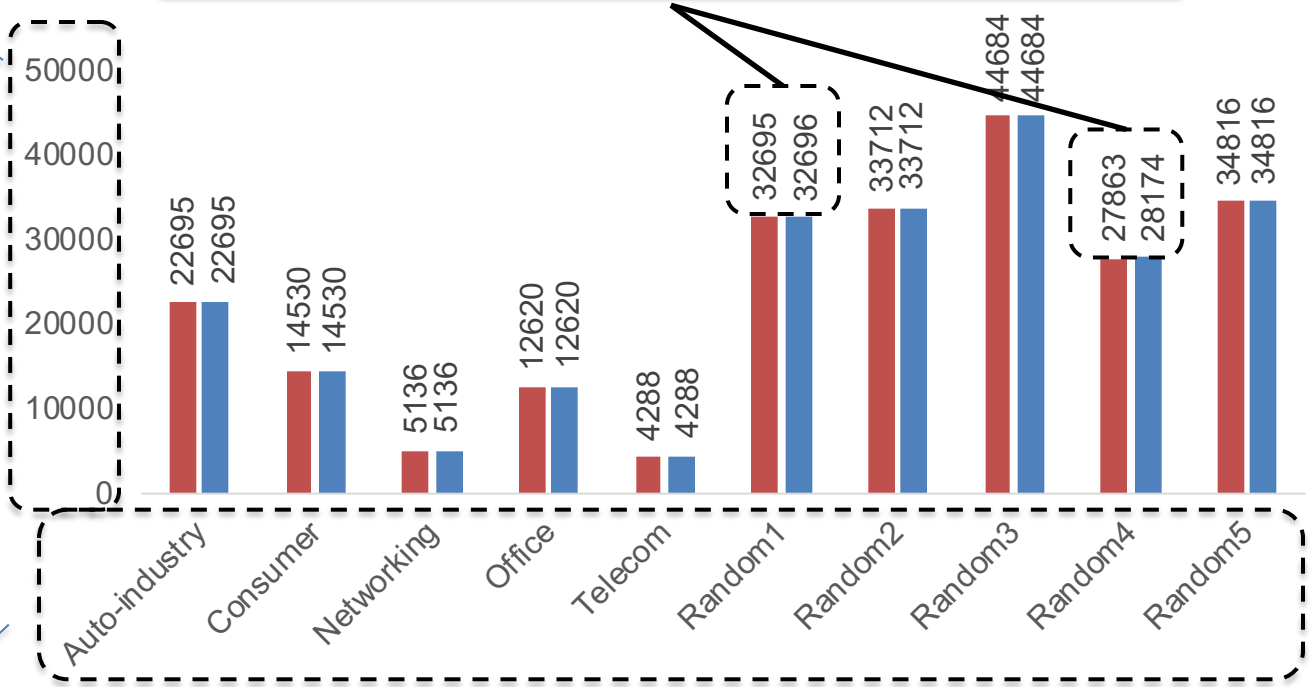


Efficiency

Compare scheduling lengths

Proposed countermeasure introduces negligible performance impact

Core count ≤ 8
Vendor count ≤ 3



5 standard + 5 random benchmarks

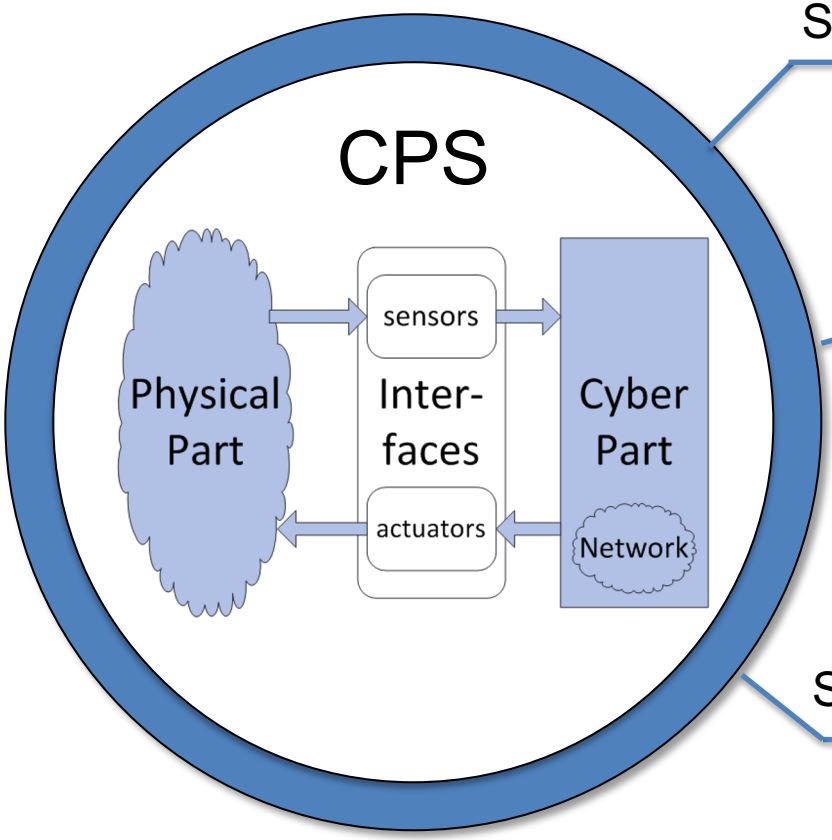
W/O security constraint

With security constraint

Outline

- Introduction
  - Hardware Trojan collusion problem
  - Design-for-trust & runtime monitoring
- Case studies
  - Trojan collusion in MPSoC
  - Trojan collusion in CPS
- Summary

Cyber-Physical Systems

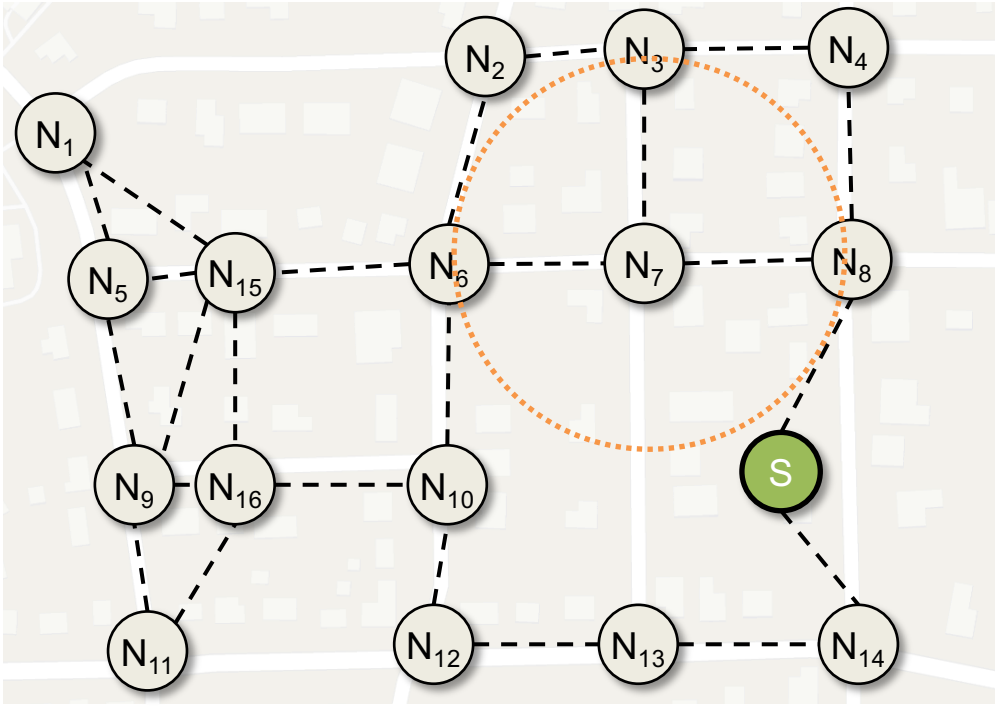


Smart grid

Smart transportation

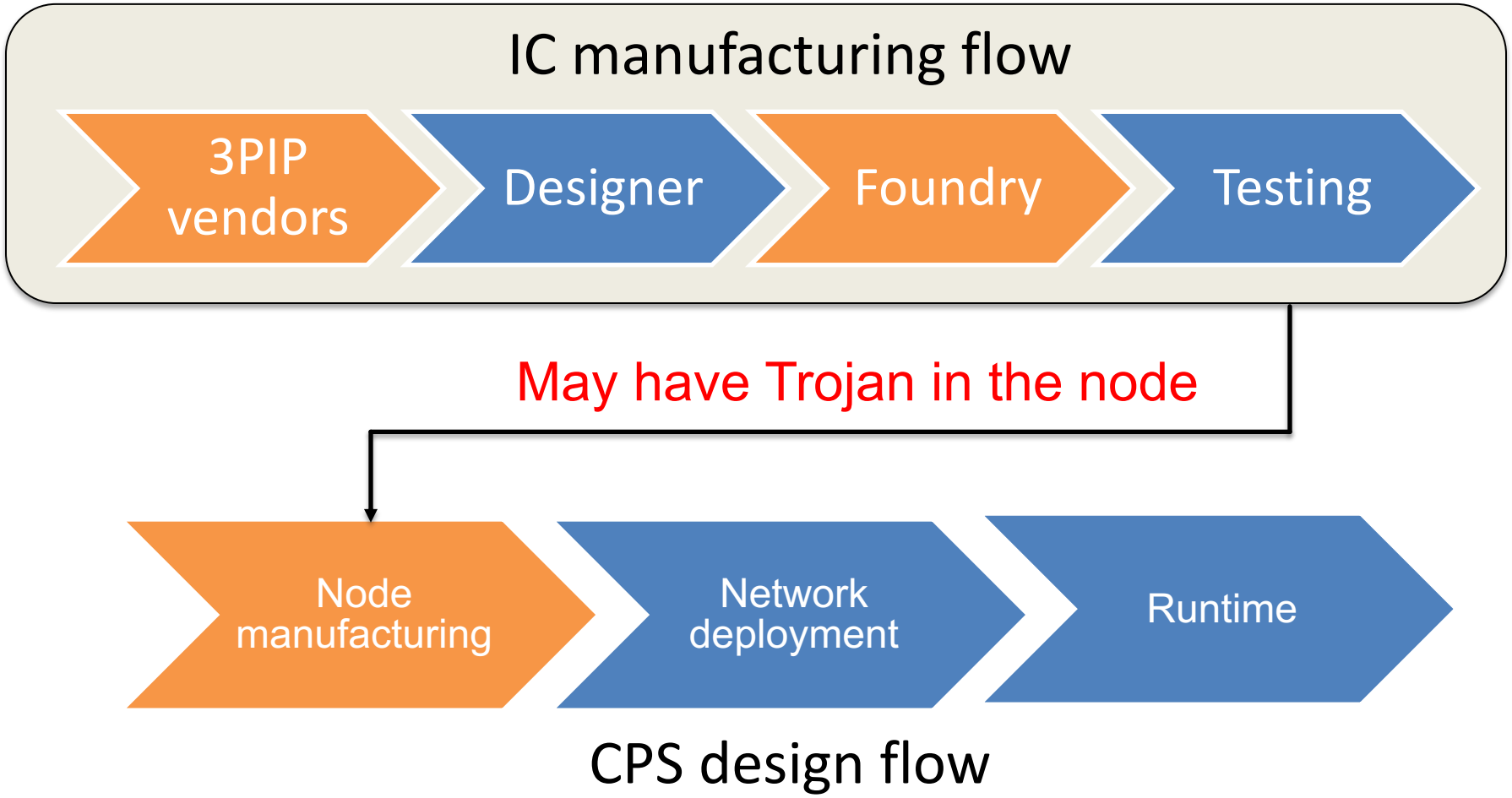
Smart home

Target CPS Example – Smart Traffic Lights

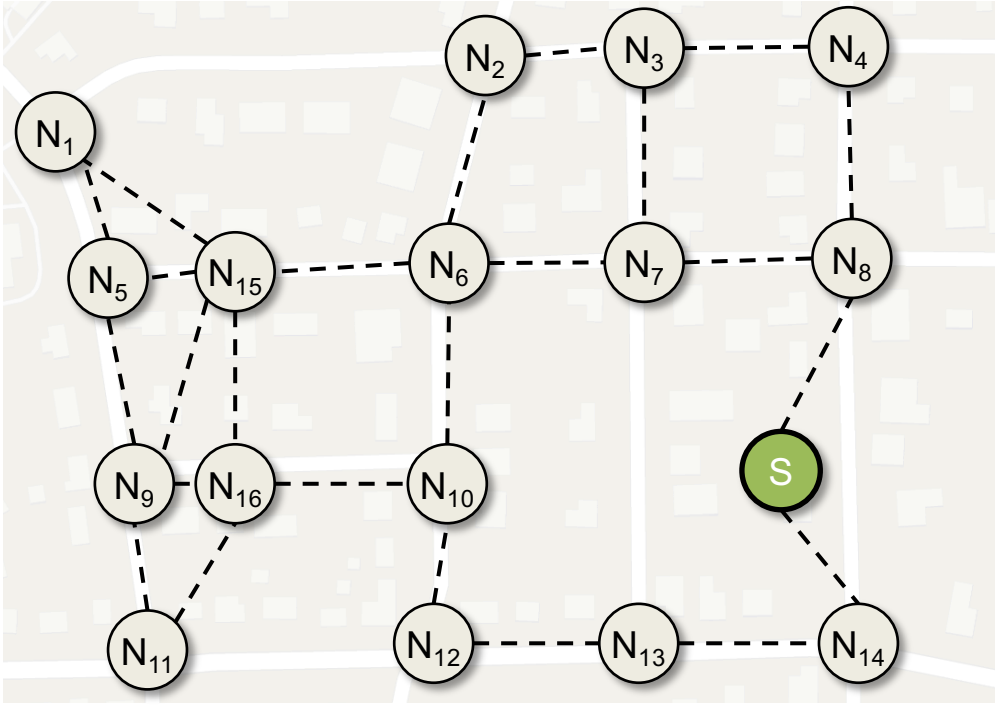


- Stationary nodes
- Stationary server(s)
- Wireless communication with a maximum transmission range
- Nodes within transmission range = neighbors
- Multi-hop communication between nodes and server

Threat Model



Threat Model



- Untrusted nodes (may have hardware Trojan)
- Trojans may collude with each other by embedding trigger in messages
- Trustworthy server(s)

Countermeasure

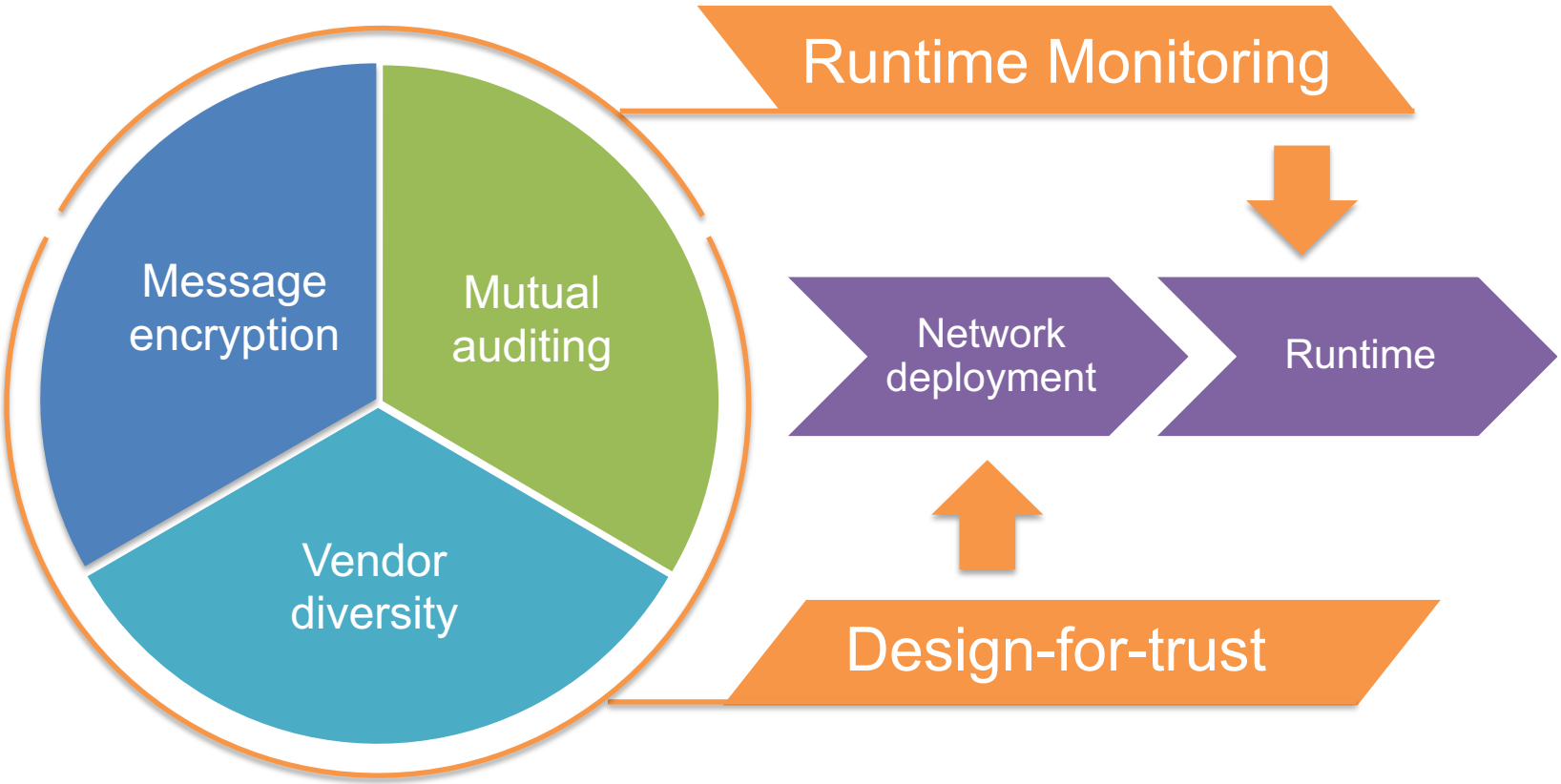
Goal: mute and/or detect Trojan collusion attempts

Same scheme for MPSoC applicable to CPS?

	MPSoC	CPS	Consequence
Differences in communication method	Direct (single-hop)	Multi-hop	No centralized monitor
	Wired	Wireless	Message can be broadcast
	Determined communication	Ad-hoc communication	Unable to perform static task scheduling

A different countermeasure is needed for CPS

Three essential components of countermeasure



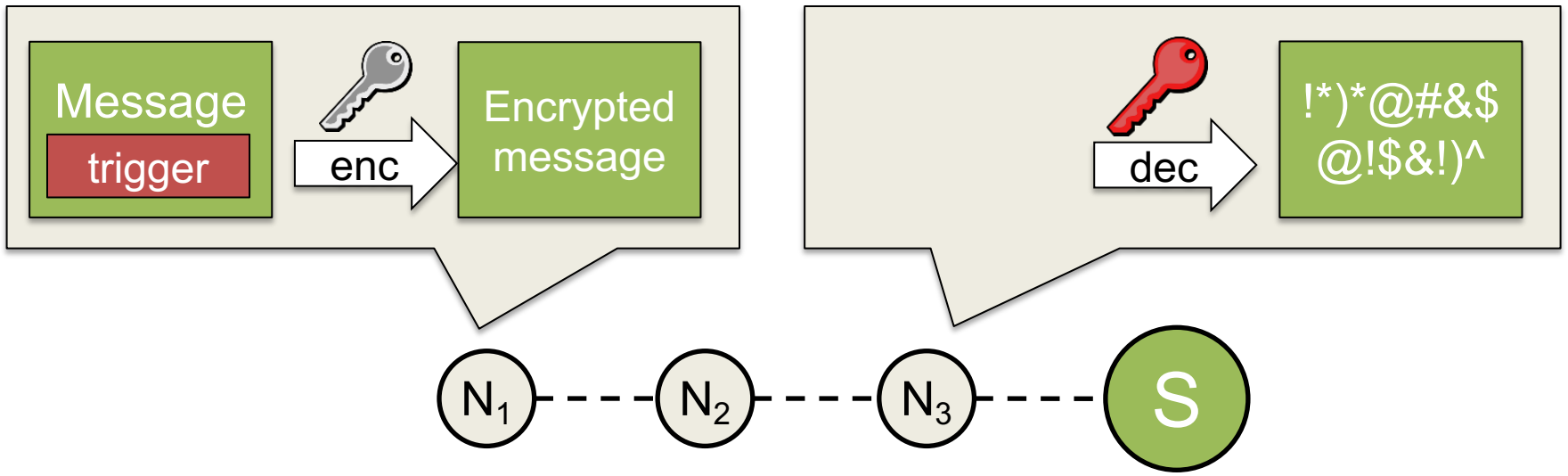
Message encryption

Why?

- Mute Trojan trigger

How:

- At network deployment: each node is assigned a unique *communication key*
- At runtime: message generator encrypts message with its key



Node mutual auditing

Why?

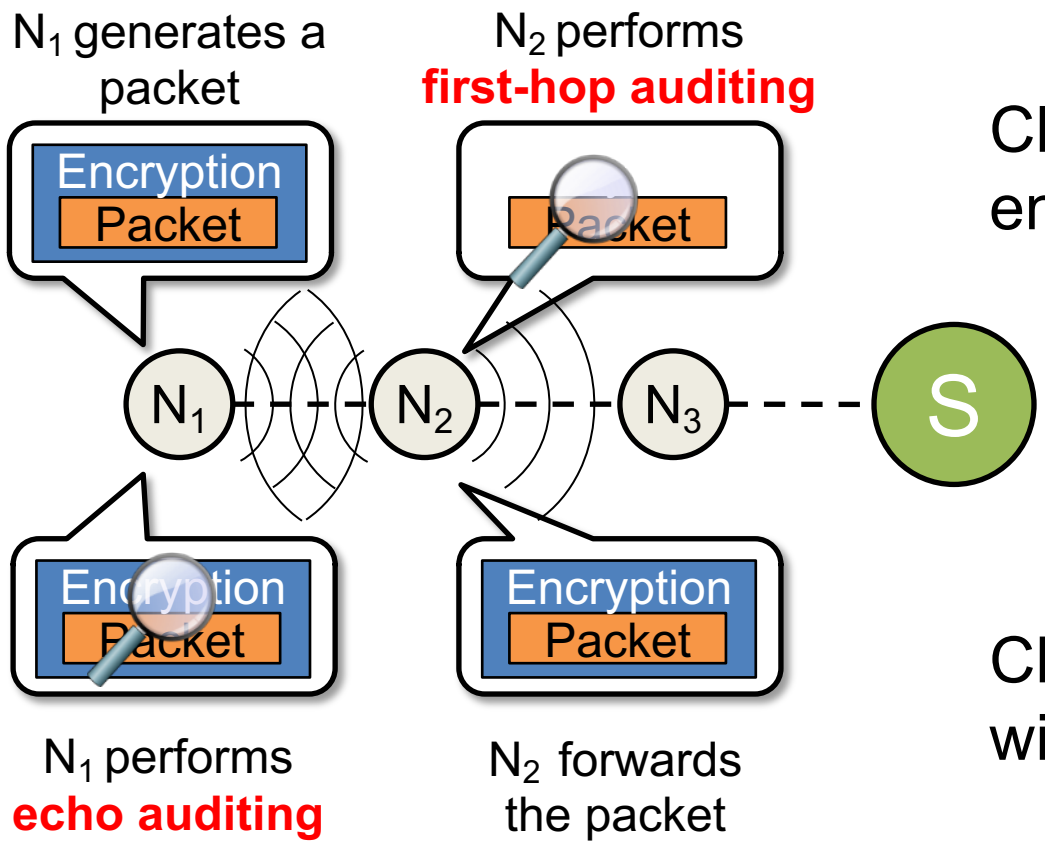
- Check if messages are correctly encrypted

How:

- At network deployment: each node is also assigned the communication keys of its neighbors
- At runtime: each node audits its neighbors
- Mutual auditing = first hop auditing + echo auditing

Node mutual auditing

Mutual auditing = first hop auditing + echo auditing

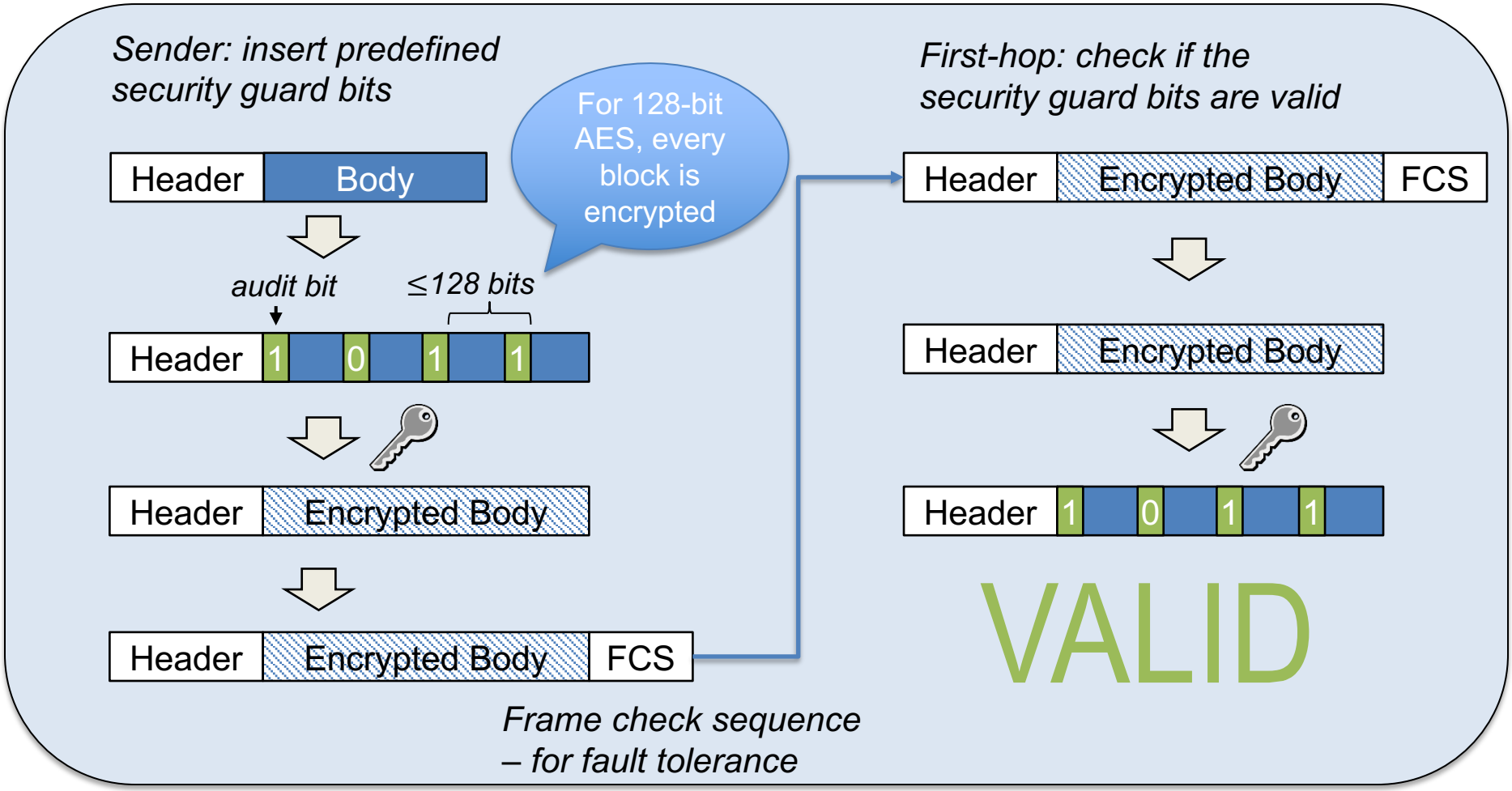


Check if N_1 correctly encrypts message

Check if N_2 tampers with the packet

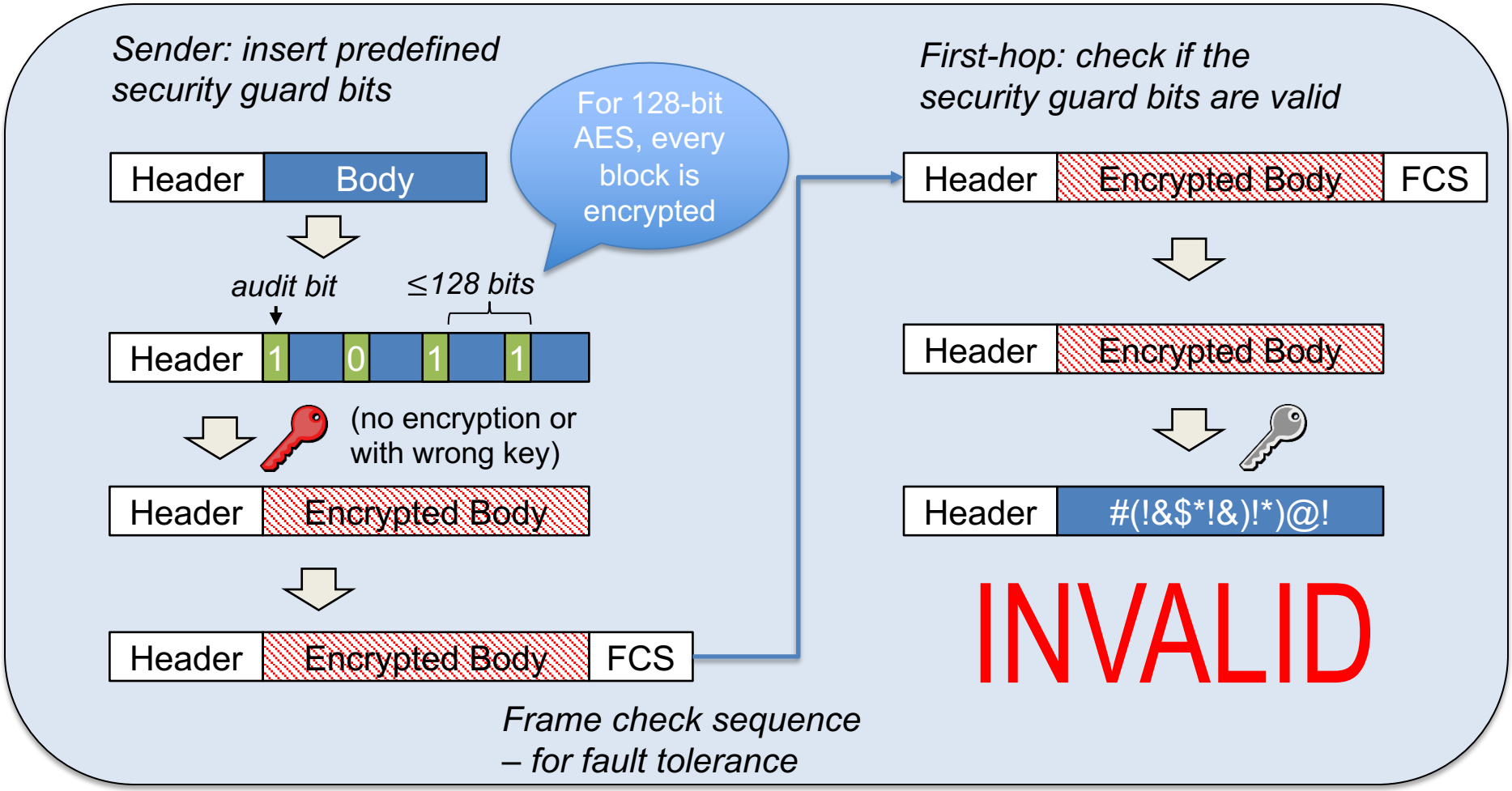
Node mutual auditing

First-hop auditing: sender node audits by first-hop node



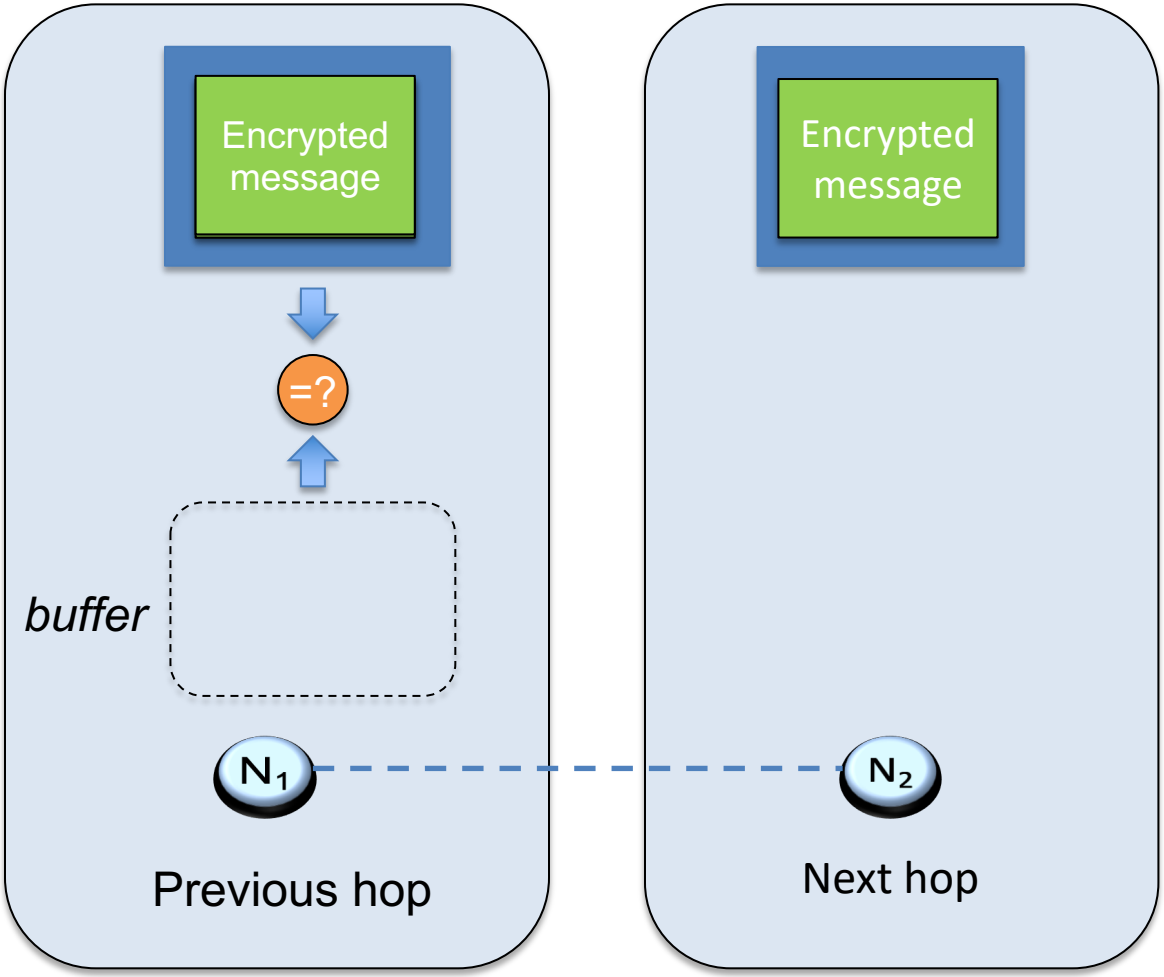
Node mutual auditing

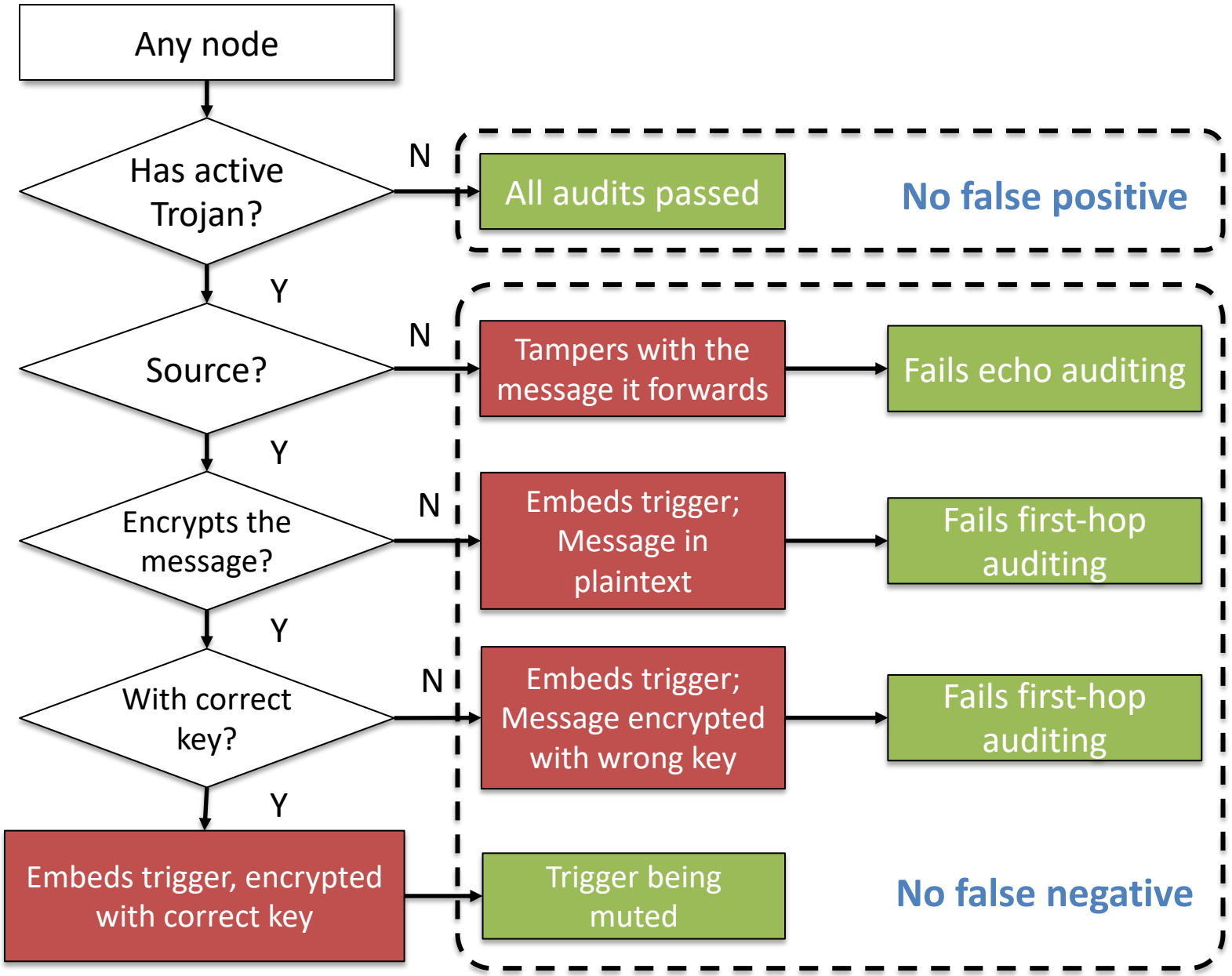
First-hop auditing: sender node audits by first-hop node



Node mutual auditing

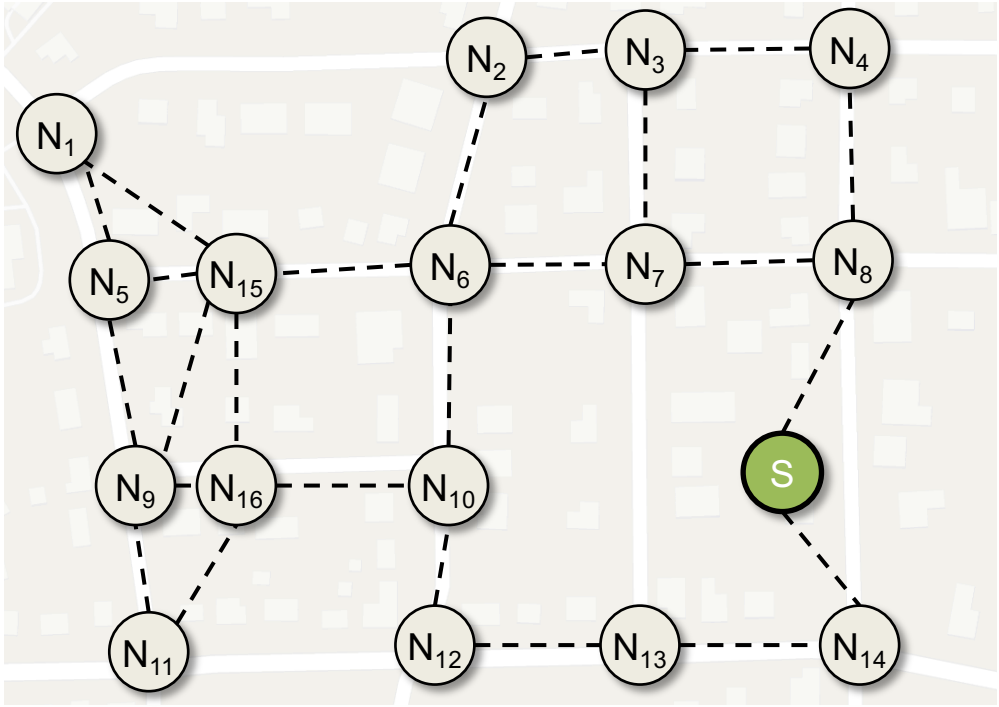
Echo auditing: forwarded message is also audited by the previous hop




Any node
Has active Trojan?
N
All audits passed
No false positive
Y
Source?
N
Tampers with the message it forwards
Fails echo auditing
Y
Encrypts the message?
N
Embeds trigger; Message in plaintext
Fails first-hop auditing
Y
With correct key?
N
Embeds trigger; Message encrypted with wrong key
Fails first-hop auditing
Y
Embeds trigger, encrypted with correct key
Trigger being muted
No false negative

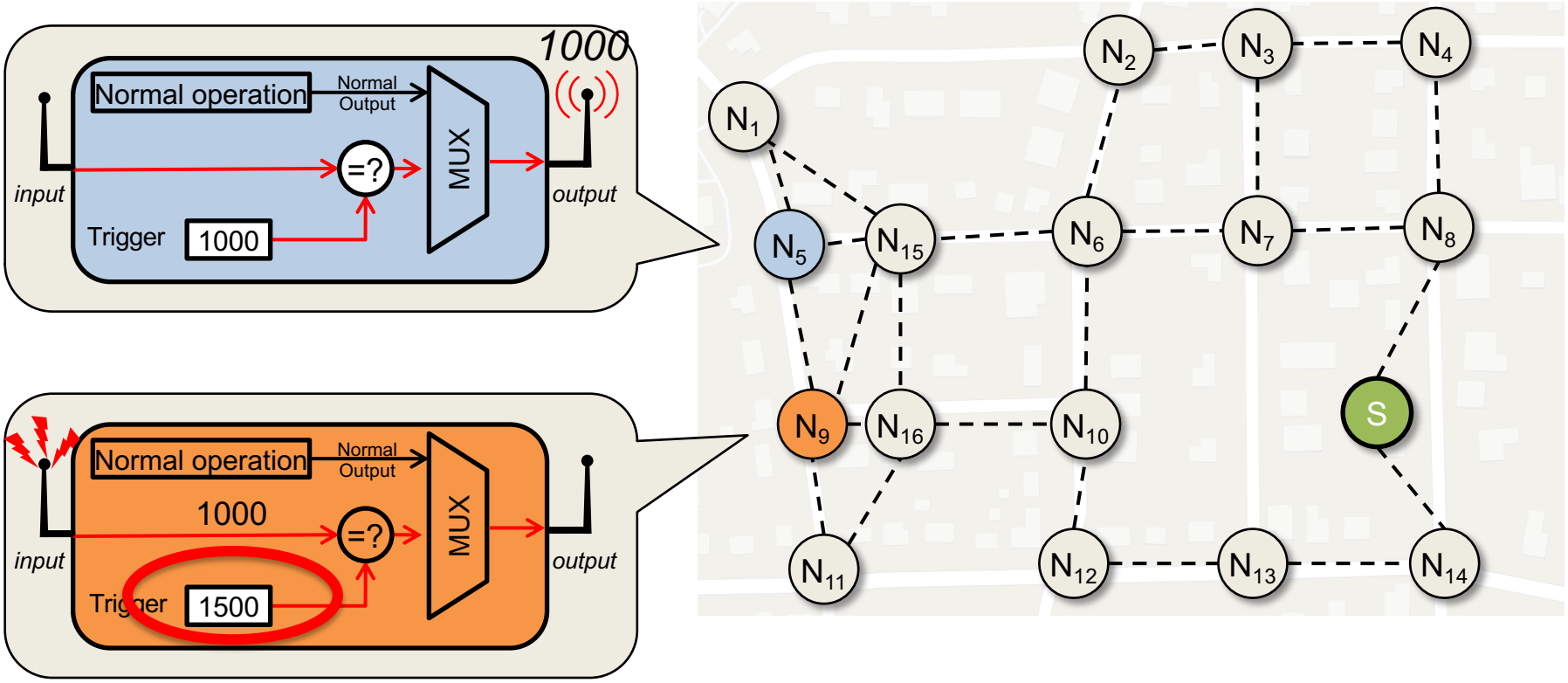
Node vendor diversity

Vendor diversity precludes nodes from collusion



Node vendor diversity

Vendor diversity precludes nodes from collusion



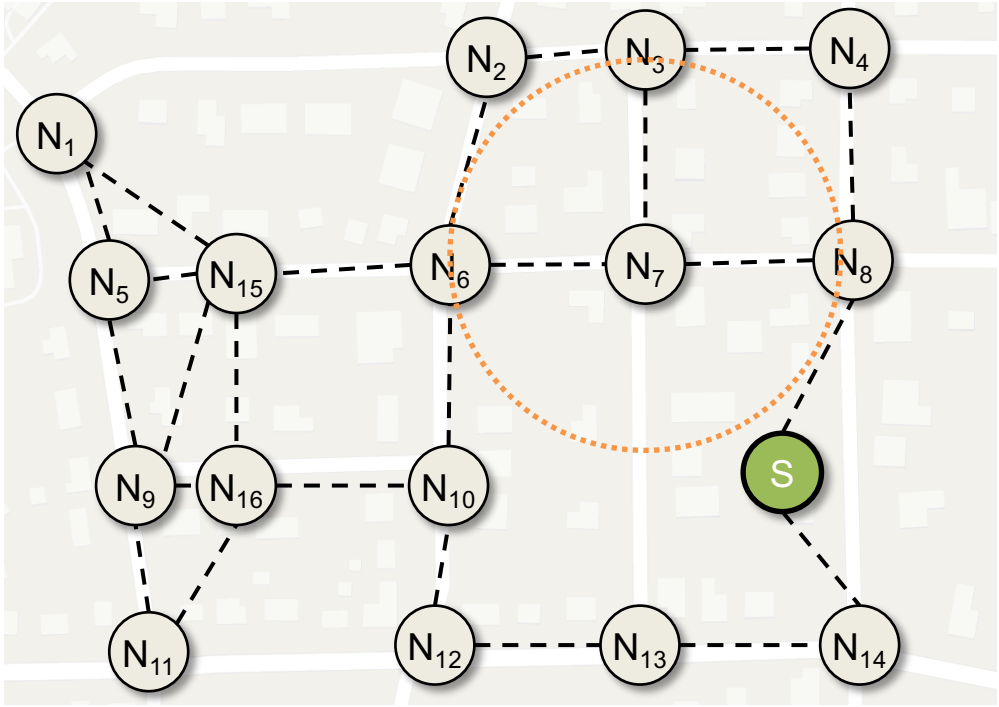
How many vendors?

Constraint: Vendor of node ≠ Vendor of any neighbor

Node routing map

Graph coloring

Node vendor selection



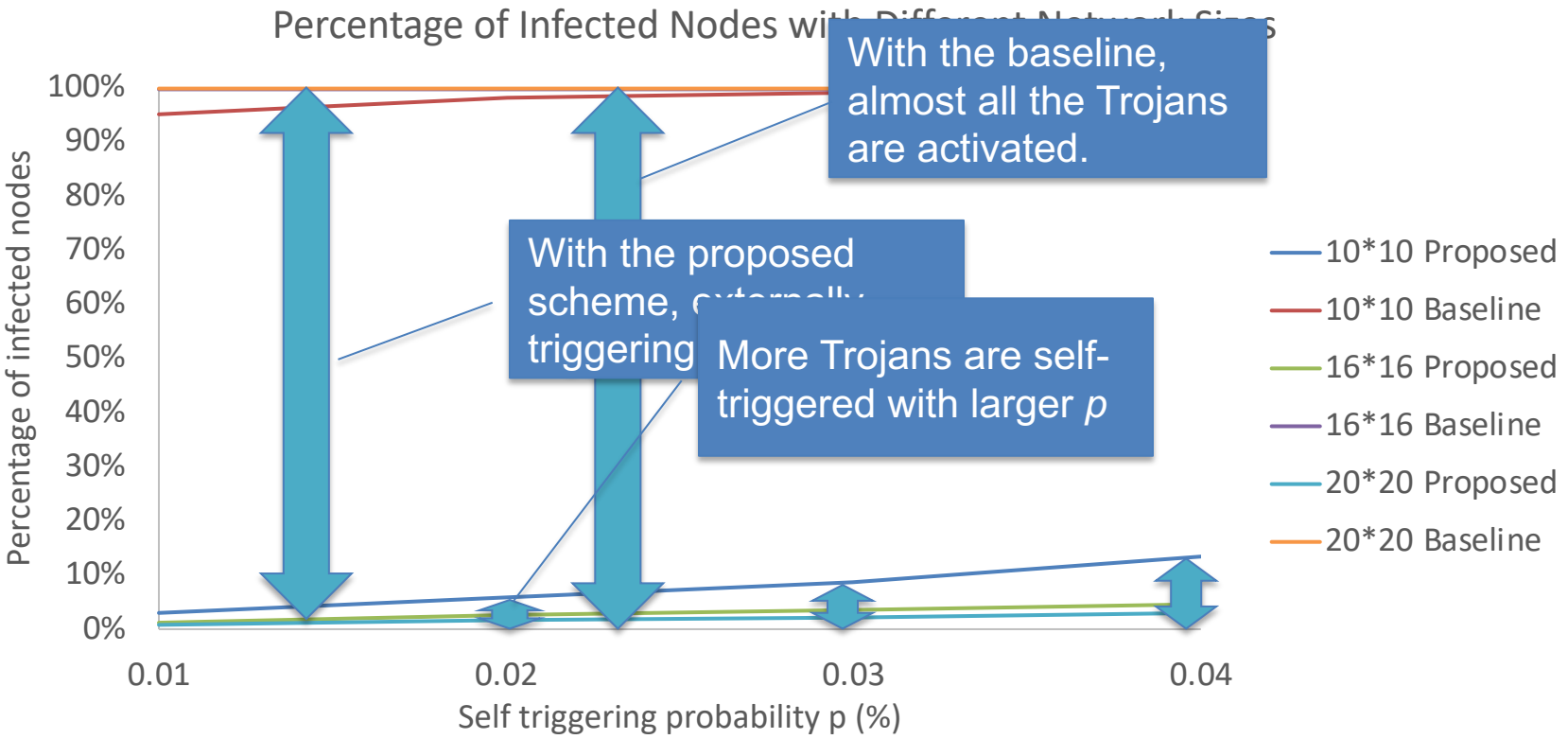
Experiment configurations

Parameters		Values
Simulation tool		NS-3
Network	network structure	2d grid
	network size	ranging from 10 × 10 to 20 × 20
	expected traffic	ranging from 40 to 100 packets/s
Network parameters	packet size	200 B body + 78 B metadata
	packet processing time	1 ms per hop
	cryptography overhead	1 ms per 128 bits
Simulation time		5 minutes

Countermeasure effectiveness study

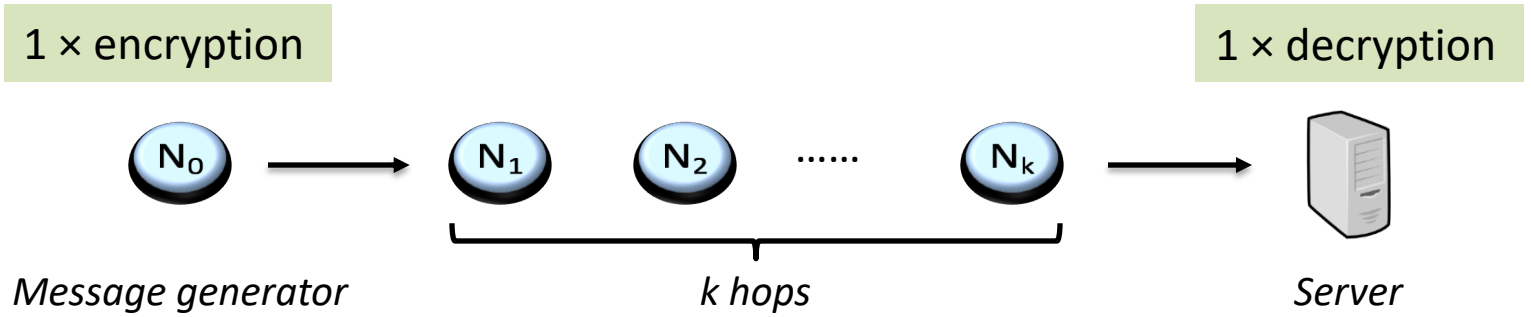
A hibernating Trojan can be either:

- **Self-triggered** with a probability of *p%* per packet generated
- ~~**Externally triggered** by successfully receiving and decoding triggering message sent by activated Trojan from the same vendor~~

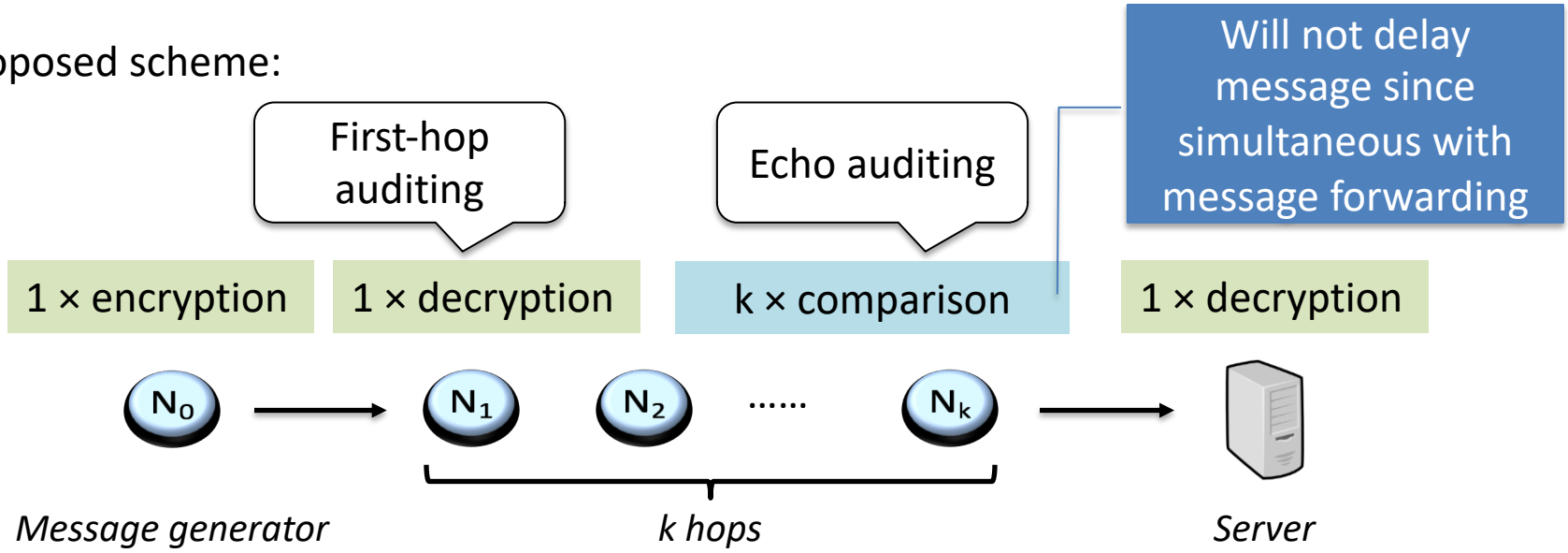


Efficiency analysis

Regular CPS with message encryption:



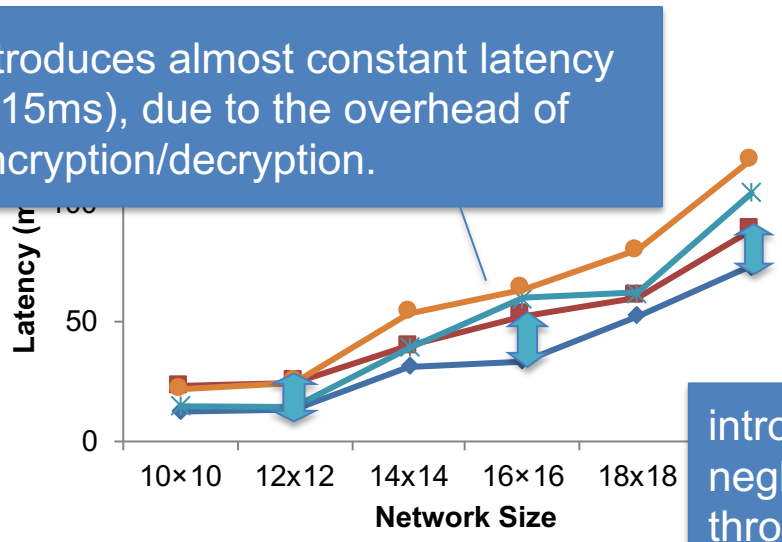
Proposed scheme:

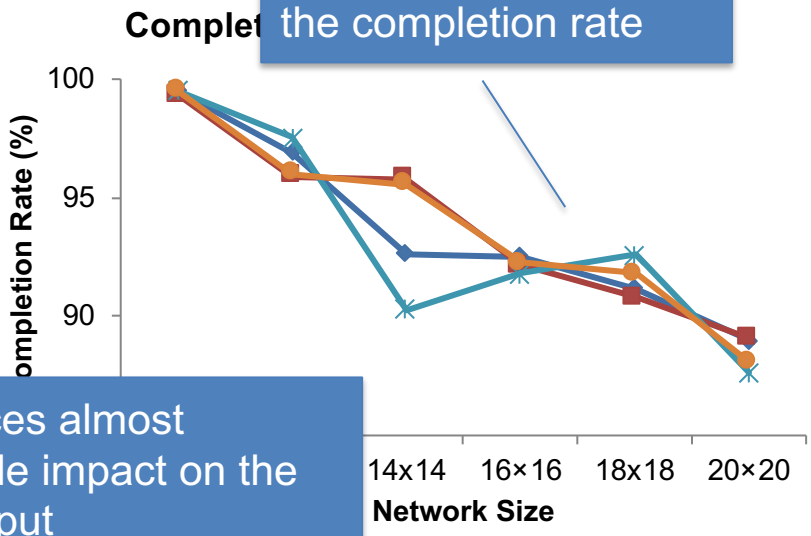


Countermeasure efficier

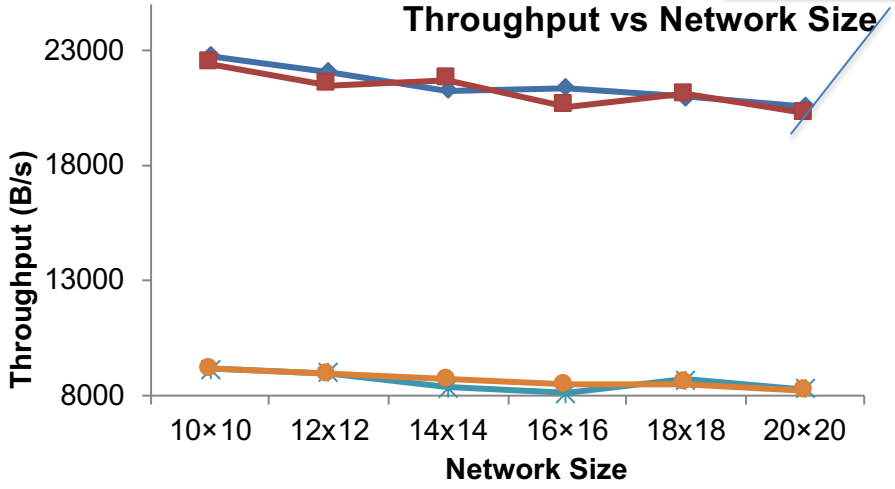
introduces almost negligible impact on the completion rate

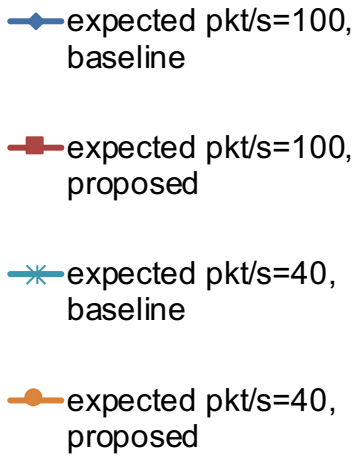
introduces almost constant latency (~15ms), due to the overhead of encryption/decryption.





introduces almost negligible impact on the throughput





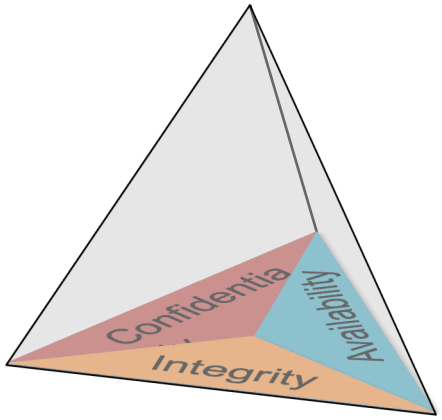
With the countermeasure

Outline

- Introduction
  - Hardware Trojan collusion problem
  - Design-for-trust & runtime monitoring
- Case studies
  - Trojan collusion in MPSoC
  - Trojan collusion in CPS
- Summary

Summary

- Problem: Hardware Trojan collusion



  - In multiple-component system, each component may have Trojan.
  - Active Trojans may spread out trigger to activate hibernating Trojans.

- Goal: build trustworthy system with untrusted components

- Solution: design-for-trust + runtime monitoring

Runtime monitoring	*Monitor particular behaviors to identify illegal ones*
Design-for-trust	*Help distinguish legal/illegal behavior*

THANK YOU

Q & A