NDSS 2017

TenantGuard: Scalable Runtime Verification of Cloud-Wide VM-Level Network Isolation

Y. Wang[1], T. Madi[1], S. Majumdar[1], Y. Jarraya[2],
A. Alimohammadifar[1], M. Pourzandi[2], L. Wang[1] and M. Debbabi[1]
[1] Concordia University, Canada, [2] Ericsson Security Research, Canada

ERICSSON

UNIVERSITÉ Concordia UNIVERSITY

NSERC CRSNG

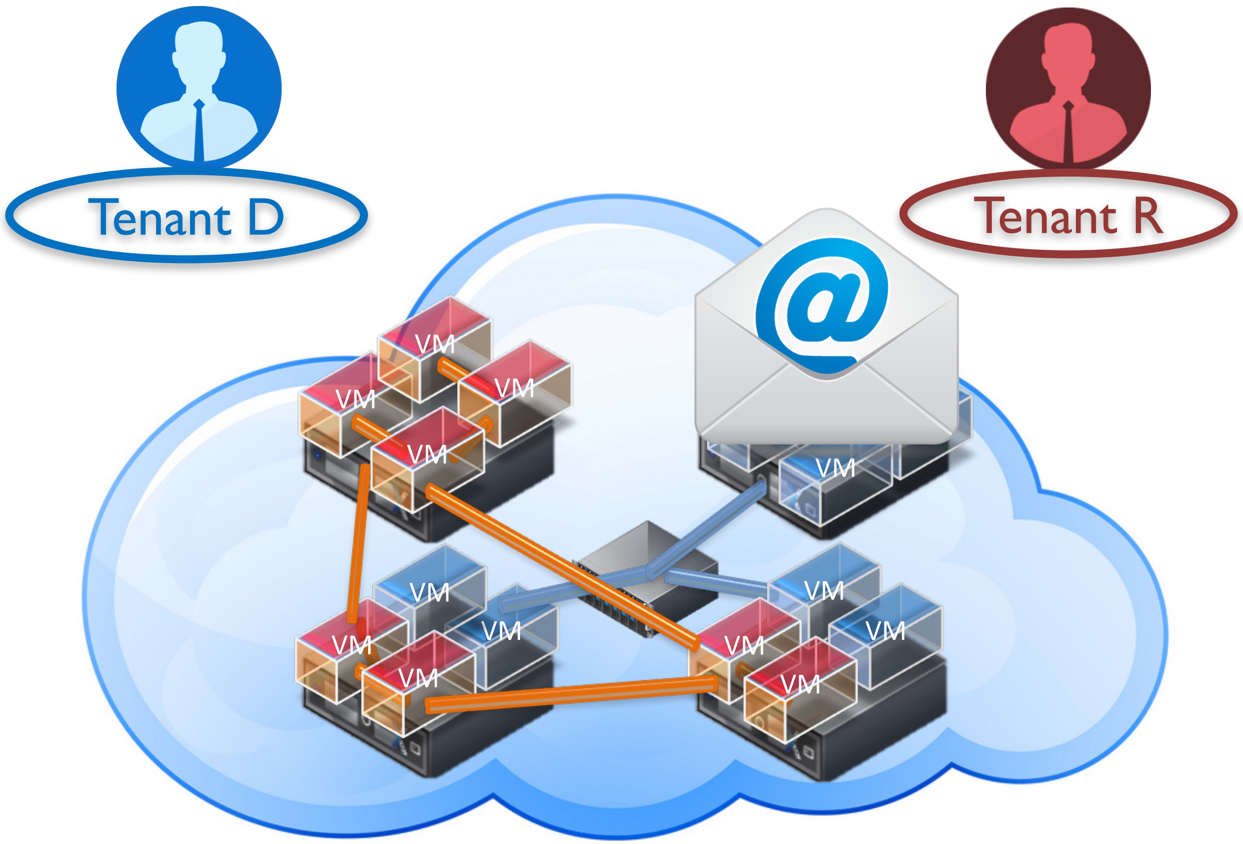
Highlights

TenantGuard, a VM-level network isolation verification system

- Pairwise reachability for over 25K VMs in 13s
- Built on OpenStack, a popular cloud management platform
- Based on a hierarchical model for virtual networks
- Leveraging efficient data structures, incremental verification and parallel computation

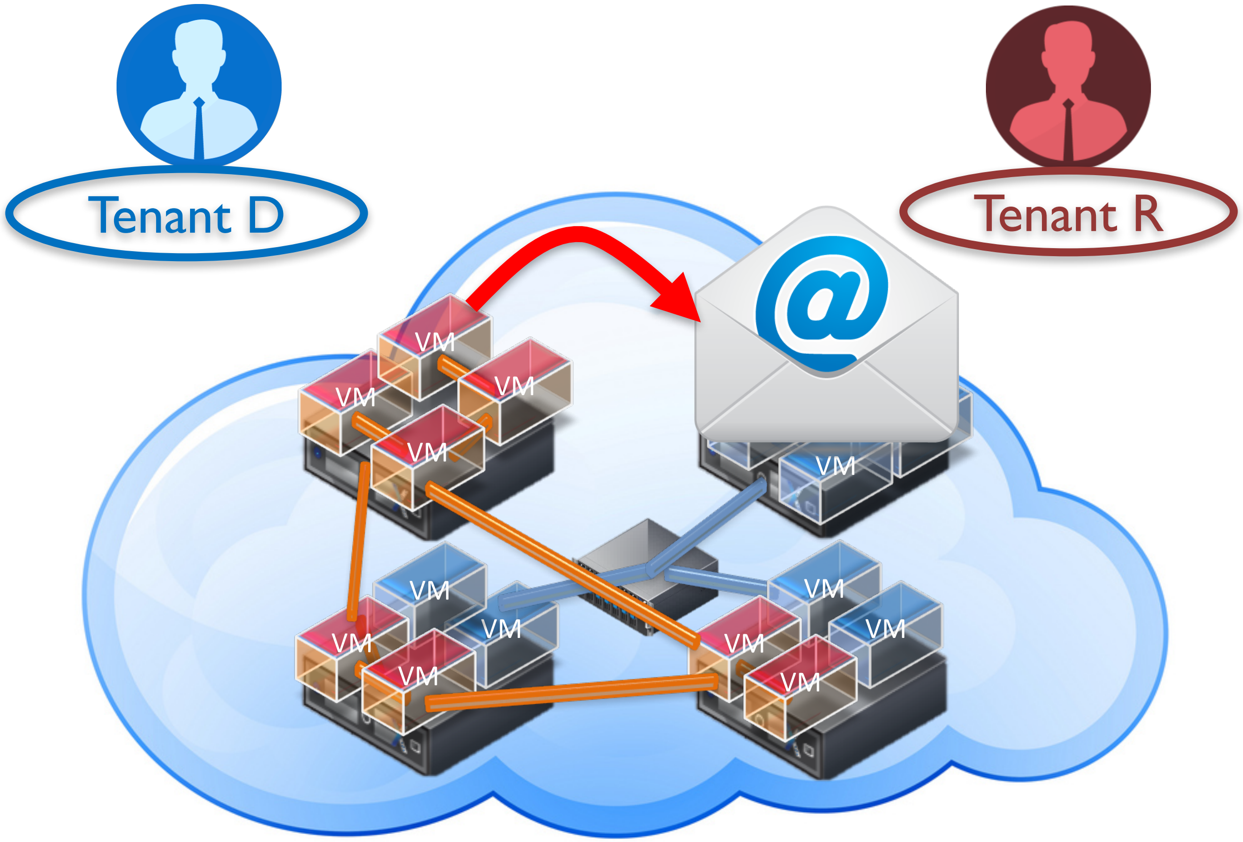
Isolation Breaches

One of the Biggest Security Concerns in Cloud



Isolation Breaches

One of the Biggest Security Concerns in Cloud

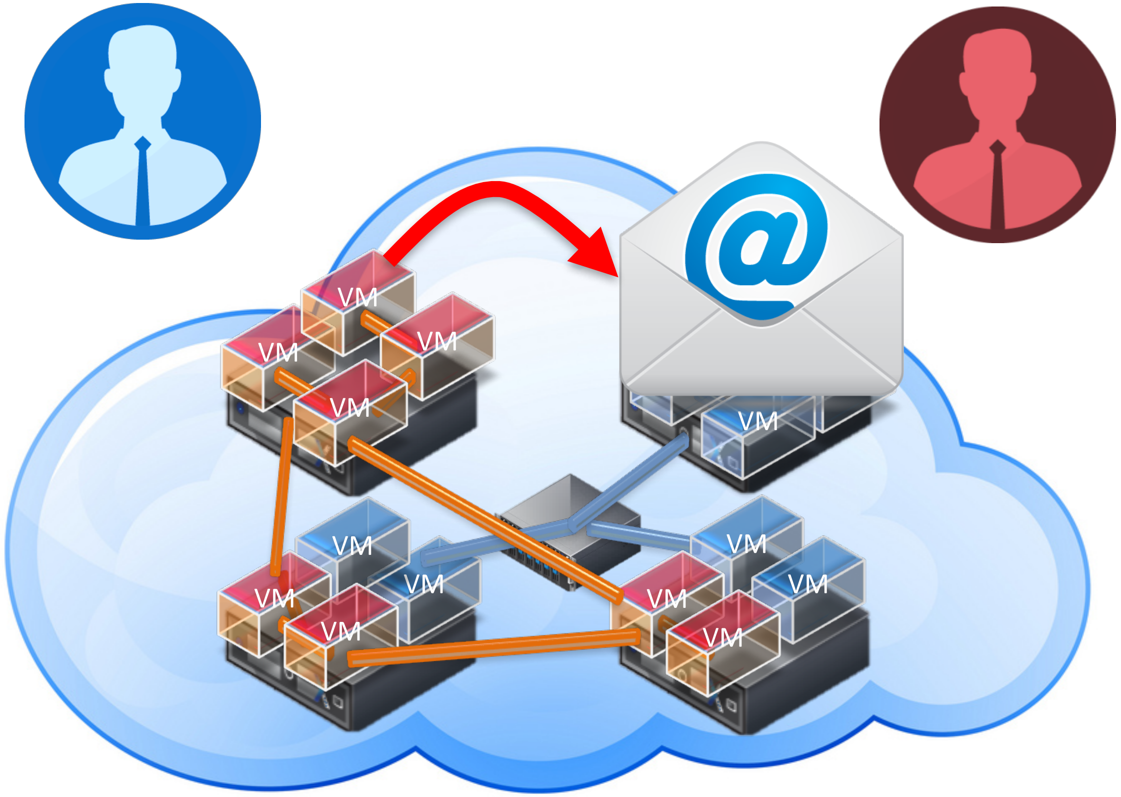


“Something” went wrong and D is hacked!

Isolation Breaches

One of the Biggest Security Concerns in Cloud

OpenStack real word vulnerabilities



[OSSA 2014-008]

Any tenant is able to create a port on another tenant's router!

Reported: 22.10.2013

Fixed: 27.03.2014

[OSSA 2015-021]

Security group rules are not effective on instances immediately!

Reported: 02.09.2015

Fixed: 11.09.2015

More on: https://www.cvedetails.com/vulnerability-list/vendor_id-11727/Openstack.html

Isolation Breaches

One of the Biggest Security Concerns in Cloud

One possible solution is: network isolation verification

[OSSA 2014-008]

Any tenant is able to create a port on another tenant's router!

Reported: 22.10.2013

Fixed: 27.03.2014

[OSSA 2015-021]

Security group rules are not effective on instances immediately!

Reported: 02.09.2015

Fixed: 11.09.2015

Network Isolation Verification

Challenges

1. Size of virtual networks: 150M+ VM pairs*

2. Diverse and distributed network functions
(L3/4 functions including virtual routing, NATing, firewalling)

3. Large data from heterogeneous sources

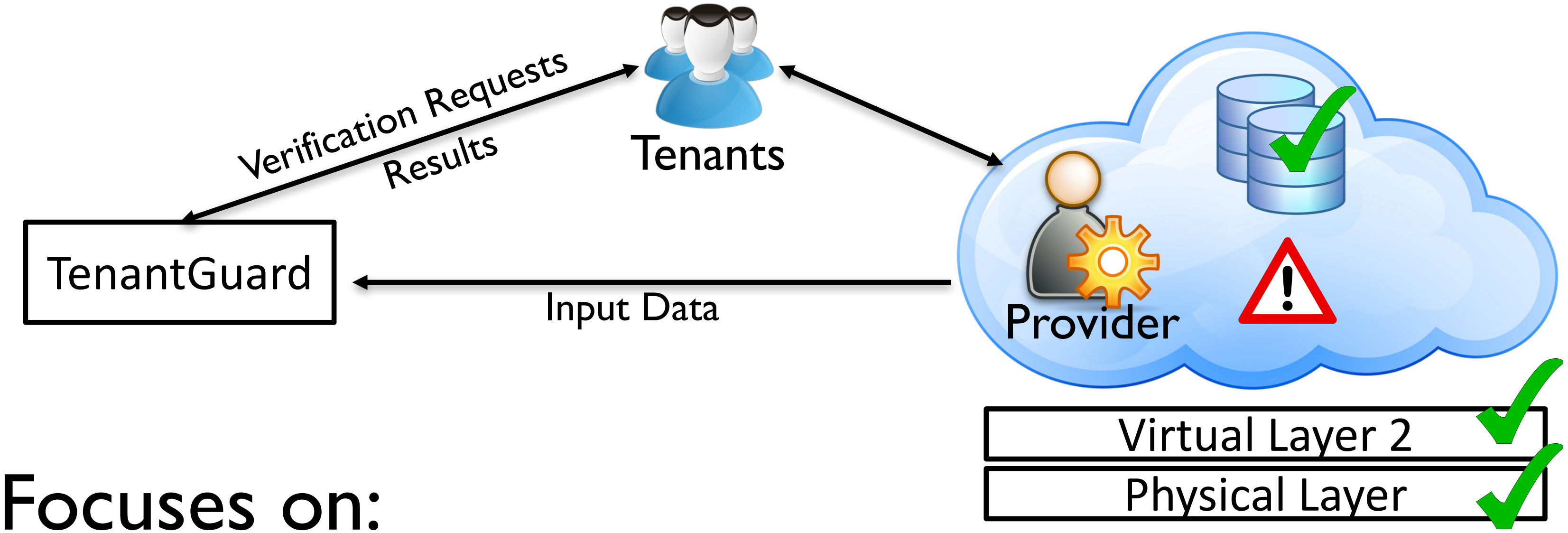
4. Quickly invalidating verification results

* OpenStack user survey, 2016. Available at: https://www.openstack.org

Existing Approaches

- Designed for physical networks
  - Not suitable for VM-level pair-wise reachability
- Focus on small to medium virtual infrastructure
  - Not designed for millions of VM pairs
- Can support VM-level reachability
  - Taking minutes to hours for over 100 million pairs

Assumptions



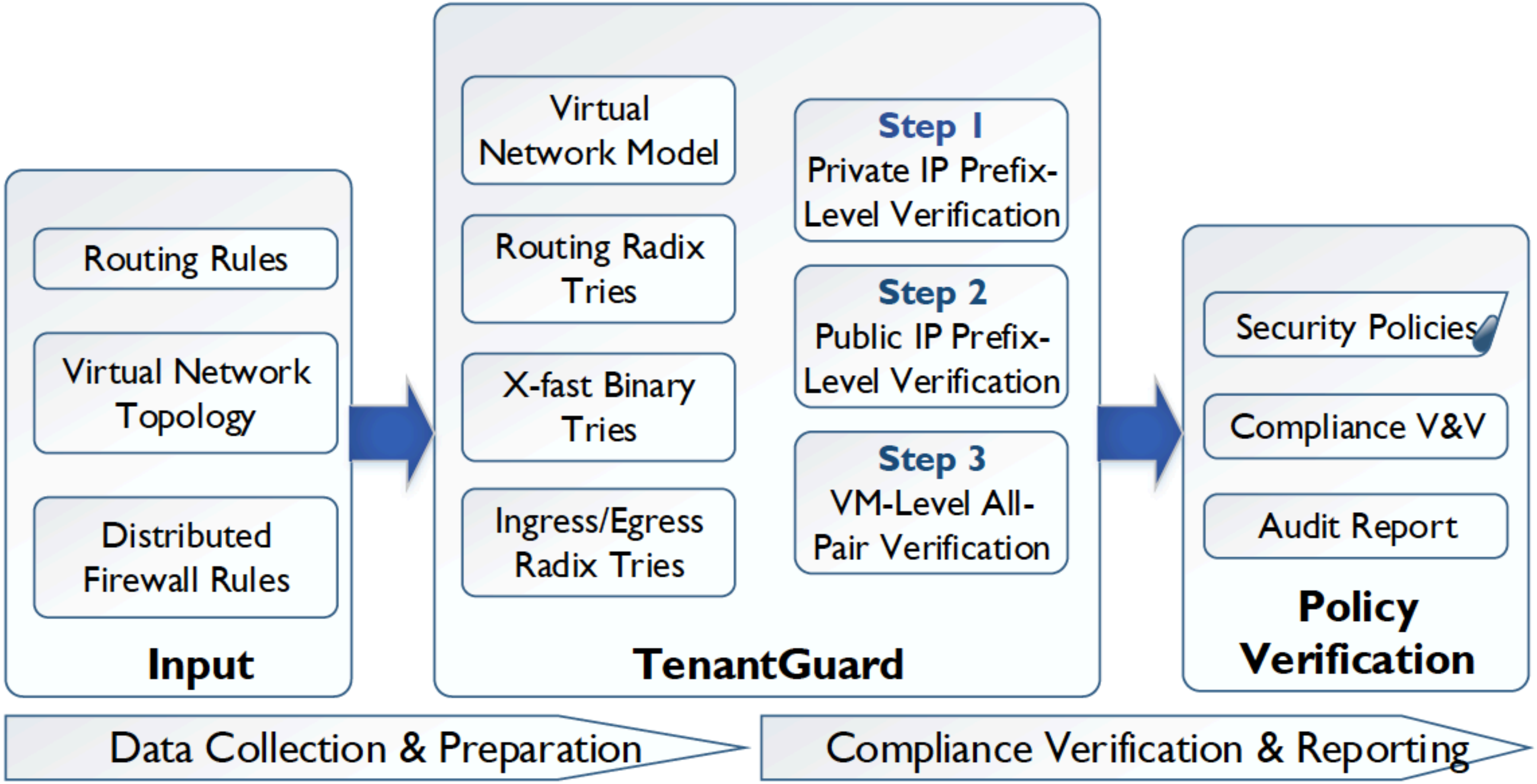
Focuses on:

- Verifying security properties specified by cloud tenants
- Not detecting any specific attack

Relies on:

- The correctness of input data
- Existing solutions at other layers
- No sensitive information in the verification results

TenantGuard: Architecture



Input

- Routing Rules
- Virtual Network Topology
- Distributed Firewall Rules

TenantGuard

- Virtual Network Model
- Routing Radix Tries
- X-fast Binary Tries
- Ingress/Egress Radix Tries

Step 1 Private IP Prefix-Level Verification

Step 2 Public IP Prefix-Level Verification

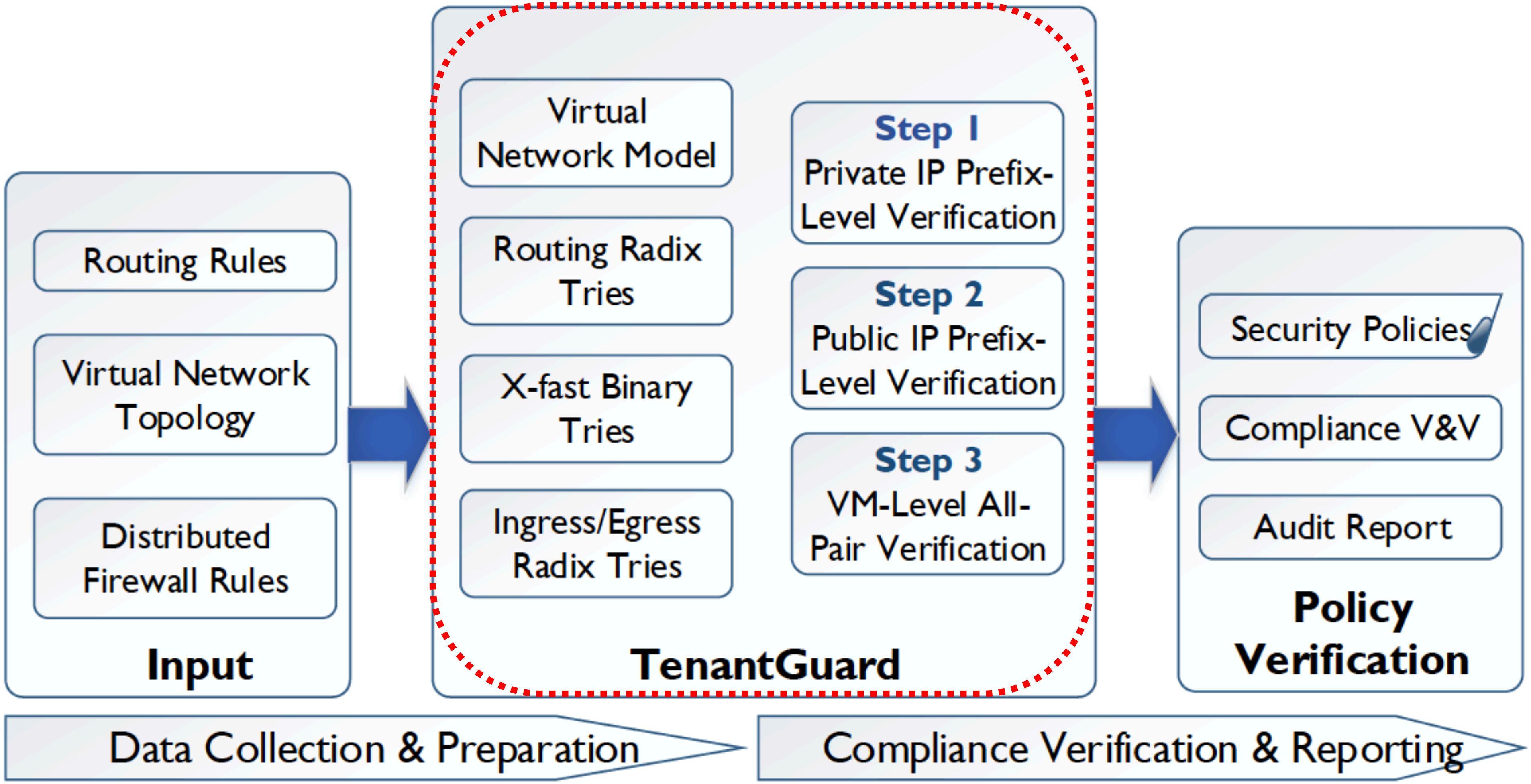
Step 3 VM-Level All-Pair Verification

Policy Verification

- Security Policies
- Compliance V&V
- Audit Report

Data Collection & Preparation → Compliance Verification & Reporting

TenantGuard: Architecture



Routing Rules

Virtual Network Topology

Distributed Firewall Rules

Input

Virtual Network Model

Routing Radix Tries

X-fast Binary Tries

Ingress/Egress Radix Tries

Step 1 Private IP Prefix-Level Verification

Step 2 Public IP Prefix-Level Verification

Step 3 VM-Level All-Pair Verification

TenantGuard

Security Policies

Compliance V&V

Audit Report

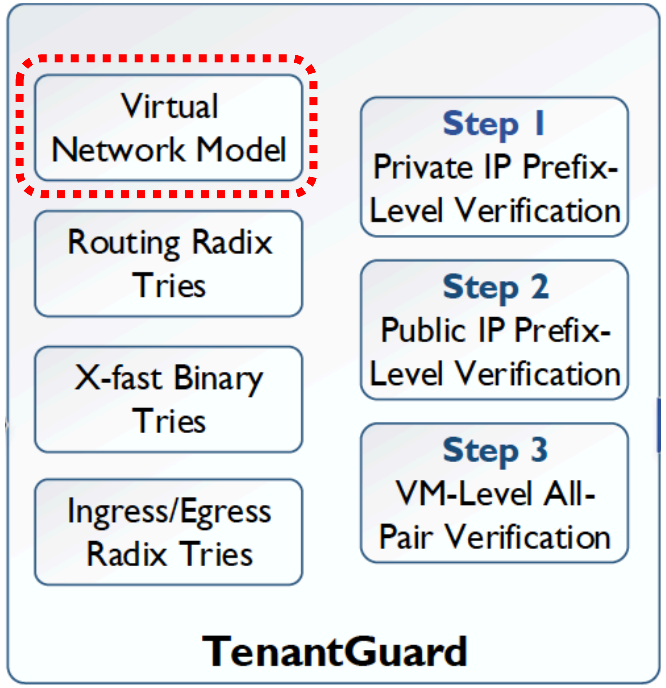
Policy Verification

Data Collection & Preparation

Compliance Verification & Reporting

Key Ideas

1. Hierarchical virtual network model (Router, subnet, VM)
2. Top-down verification approach (from prefix-level to IP-level)
3. Efficient data structures (Radix Trie and X-fast Binary Trie)



Virtual Network Model

Routing Radix Tries

X-fast Binary Tries

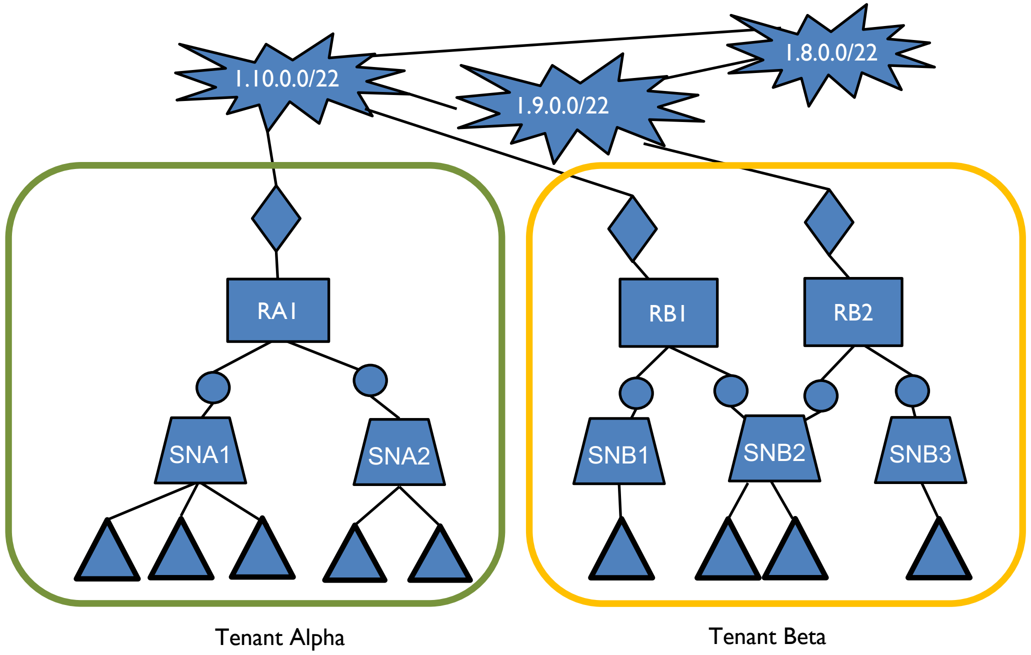
Ingress/Egress Radix Tries

Step 1
Private IP Prefix-Level Verification

Step 2
Public IP Prefix-Level Verification

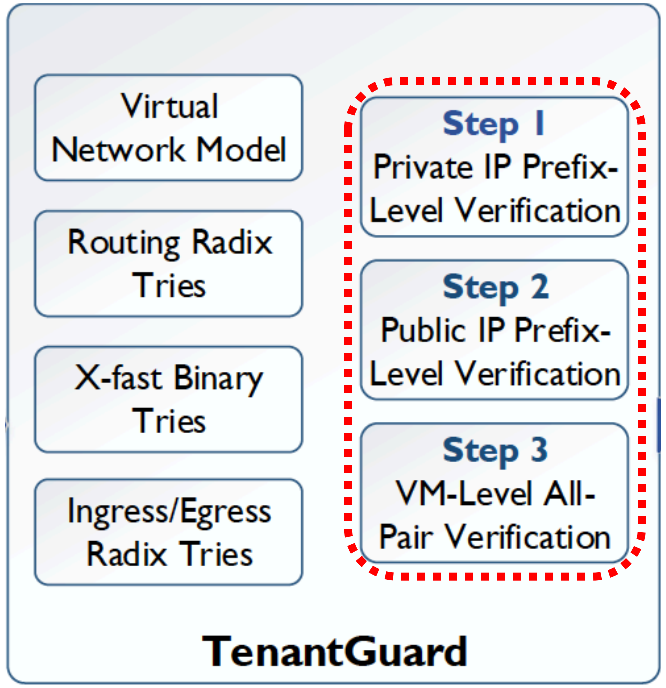
Step 3
VM-Level All-Pair Verification

TenantGuard



Key Ideas

1. Hierarchical virtual network model (Router, subnet, VM)
2. Top-down verification approach (from prefix-level to IP-level)
3. Efficient data structures (Radix Trie and X-fast Binary Trie)



Virtual Network Model

Routing Radix Tries

X-fast Binary Tries

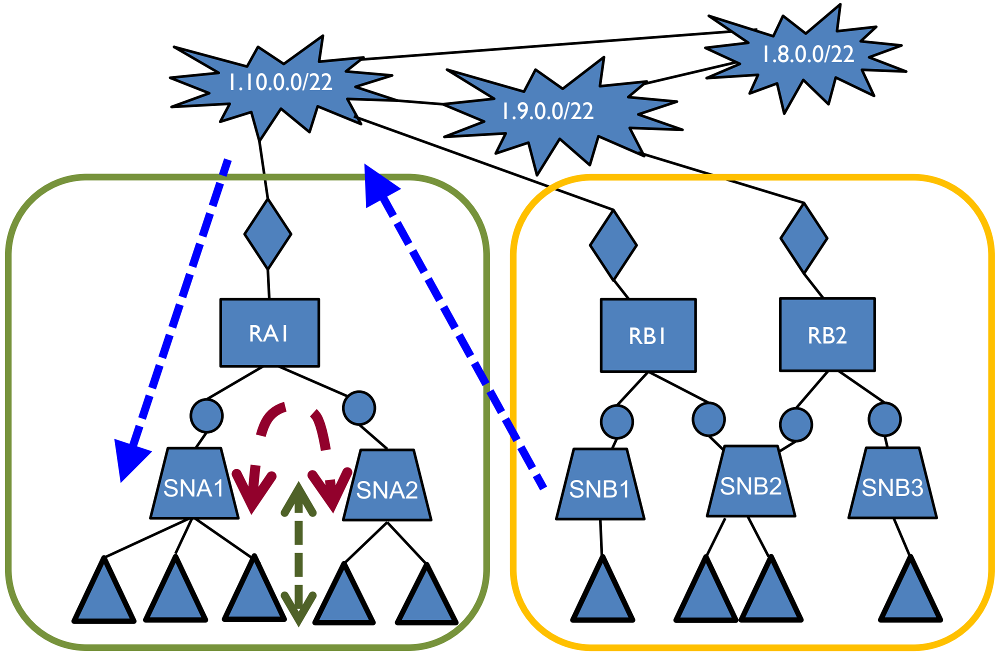
Ingress/Egress Radix Tries

Step 1 Private IP Prefix-Level Verification

Step 2 Public IP Prefix-Level Verification

Step 3 VM-Level All-Pair Verification

TenantGuard

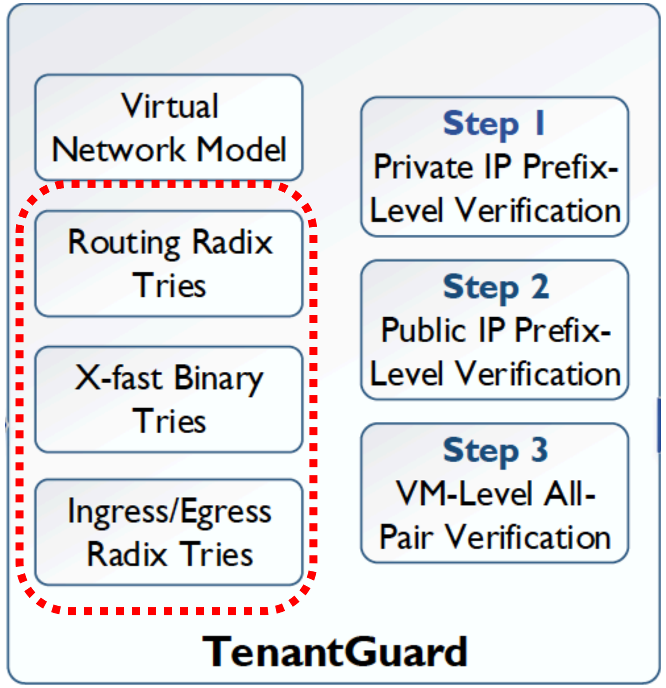


Tenant Alpha

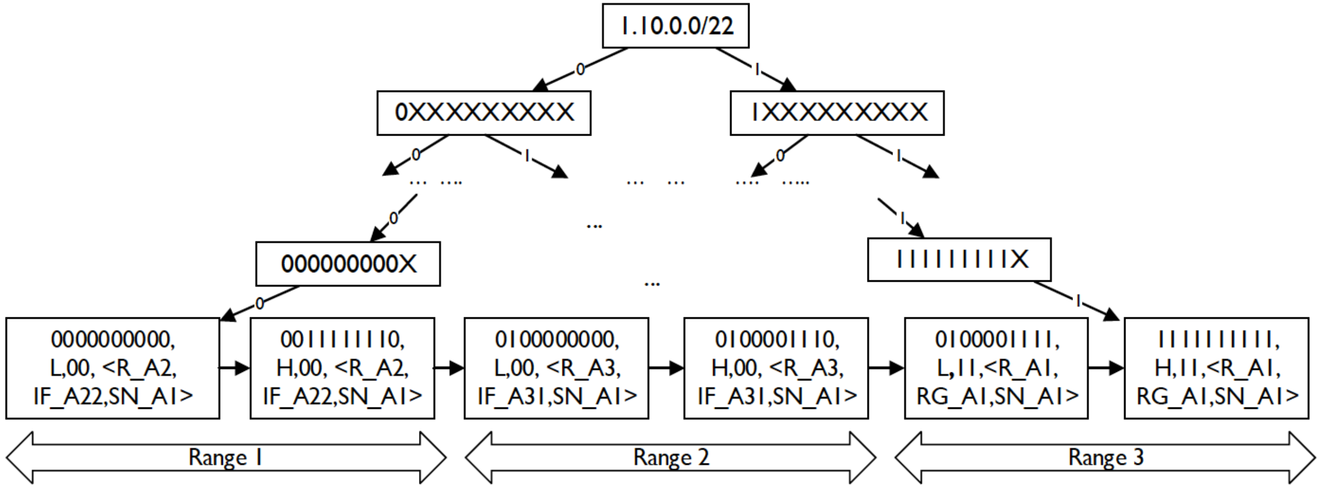
Tenant Beta

Key Ideas

1. Hierarchical virtual network model (Router, subnet, VM)
2. Top-down verification approach (from prefix-level to IP-level)
3. Efficient data structures (Radix Trie and X-fast Binary Trie)

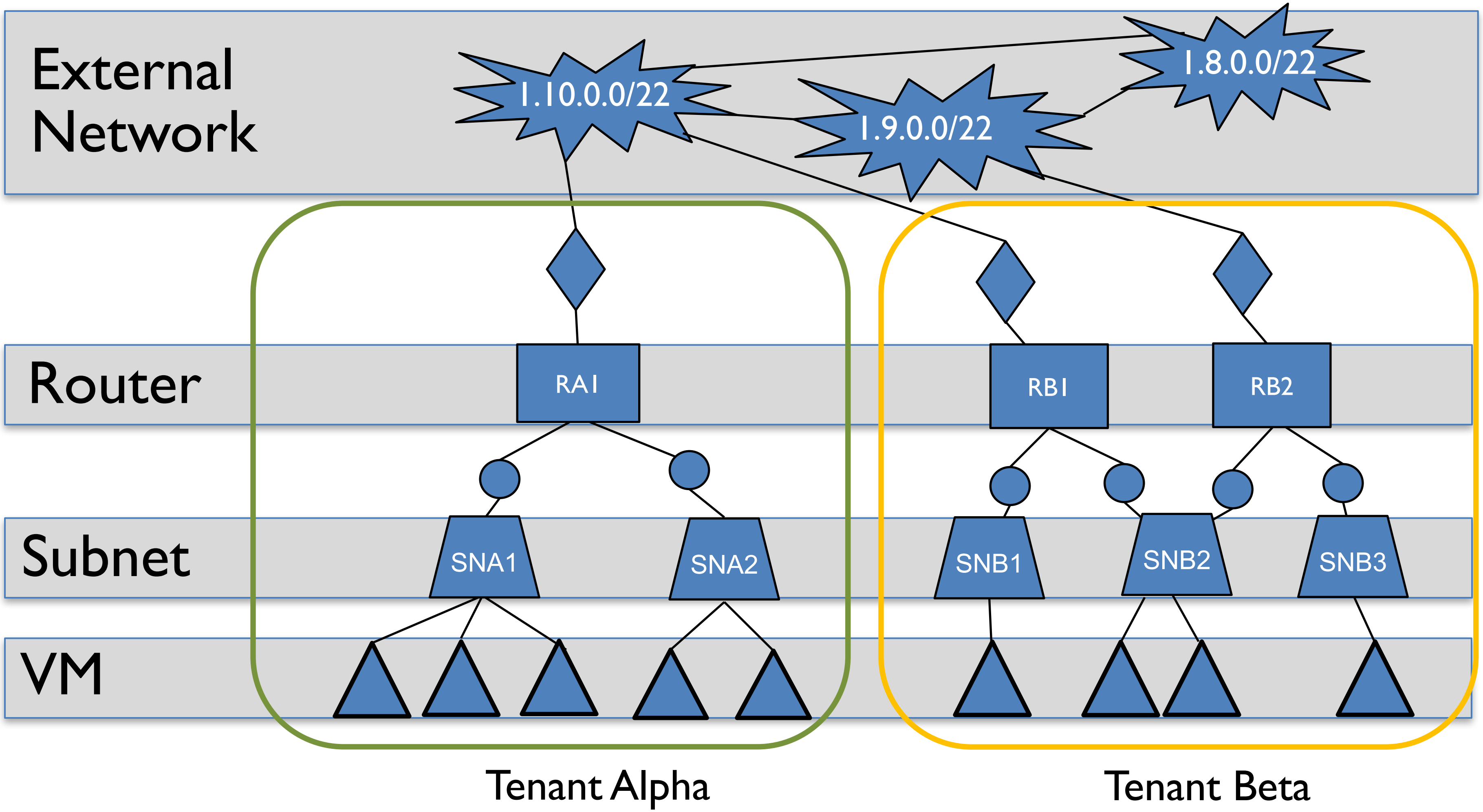


Virtual Network Model
Routing Radix Tries
X-fast Binary Tries
Ingress/Egress Radix Tries
Step 1
Private IP Prefix-Level Verification
Step 2
Public IP Prefix-Level Verification
Step 3
VM-Level All-Pair Verification
TenantGuard

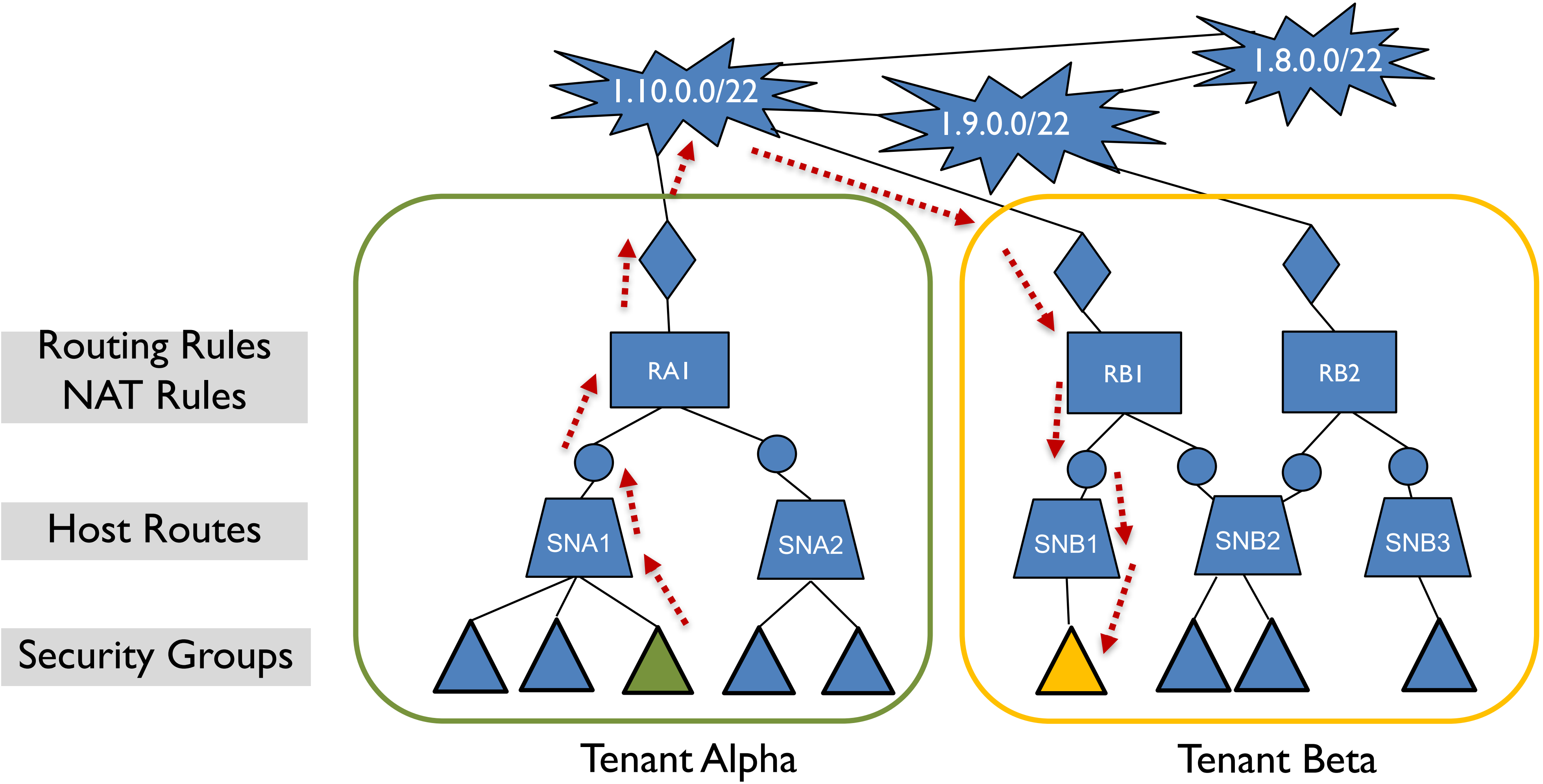


1.10.0.0/22
0
1
0XXXXXXXXX
1XXXXXXXXX
000000000X
111111111X
0000000000, L,00, <R_A2, IF_A22,SN_A1>
0011111110, H,00, <R_A2, IF_A22,SN_A1>
0100000000, L,00, <R_A3, IF_A31,SN_A1>
0100001110, H,00, <R_A3, IF_A31,SN_A1>
0100001111, L,11,<R_A1, RG_A1,SN_A1>
1111111111, H,11,<R_A1, RG_A1,SN_A1>
Range 1
Range 2
Range 3

Hierarchical Virtual Network Model

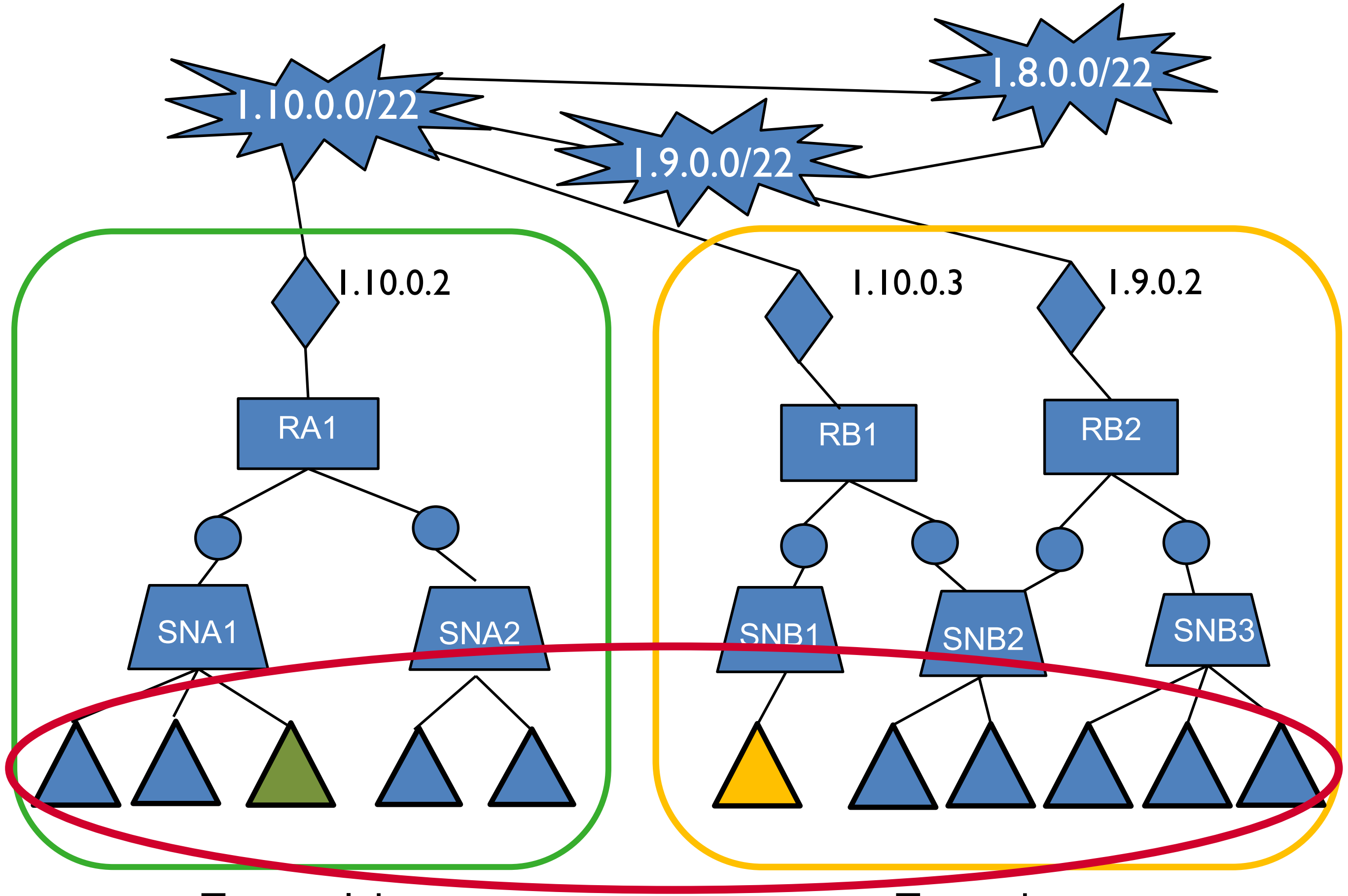


Hierarchical Virtual Network Model



Baseline Approach

Verifying every possible VM pair (e.g., over 150 million pairs!!)



Tenant alpha

Tenant beta

Top-Down Verification

Step one

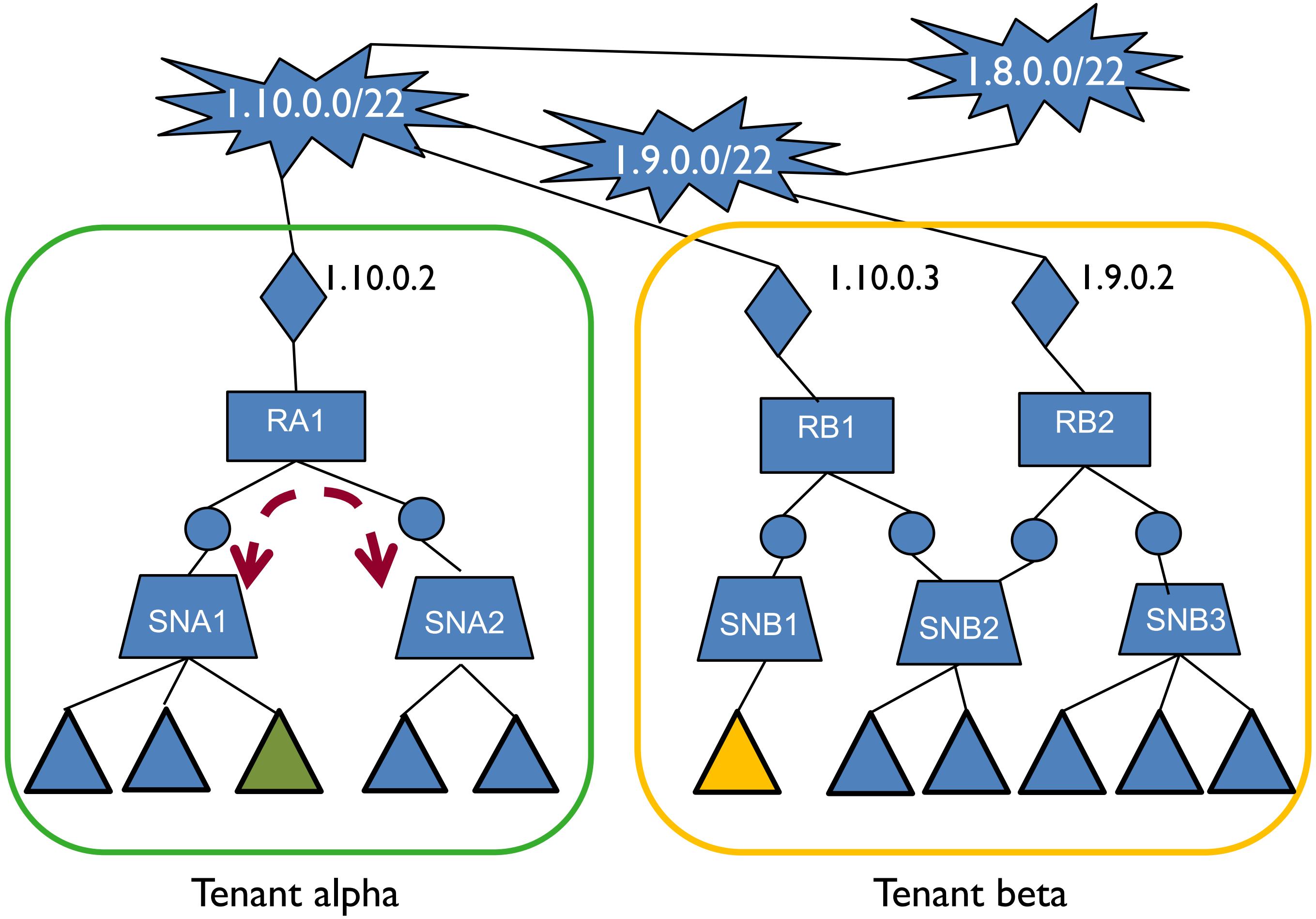
Check isolation between subnets within the same tenant environment

Step two

Check isolation between different tenant environments

Step three

Check VM-isolation only for subnets found to be reachable



Top-Down Verification

Step one

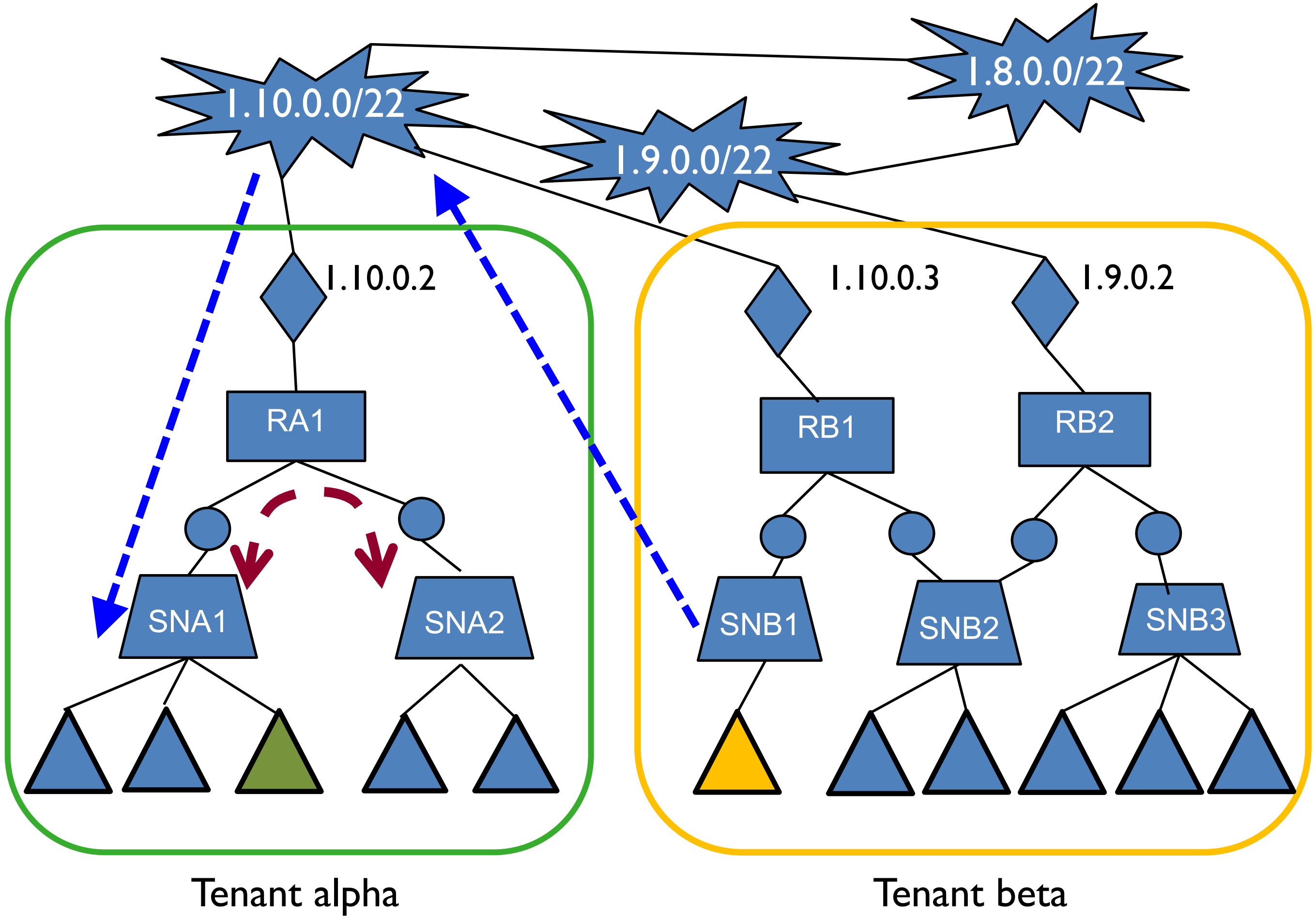
Check isolation between subnets within the same tenant environment

Step two

Check isolation between different tenant environments

Step three

Check VM-isolation only for subnets found to be reachable



Top-Down Verification

Step one

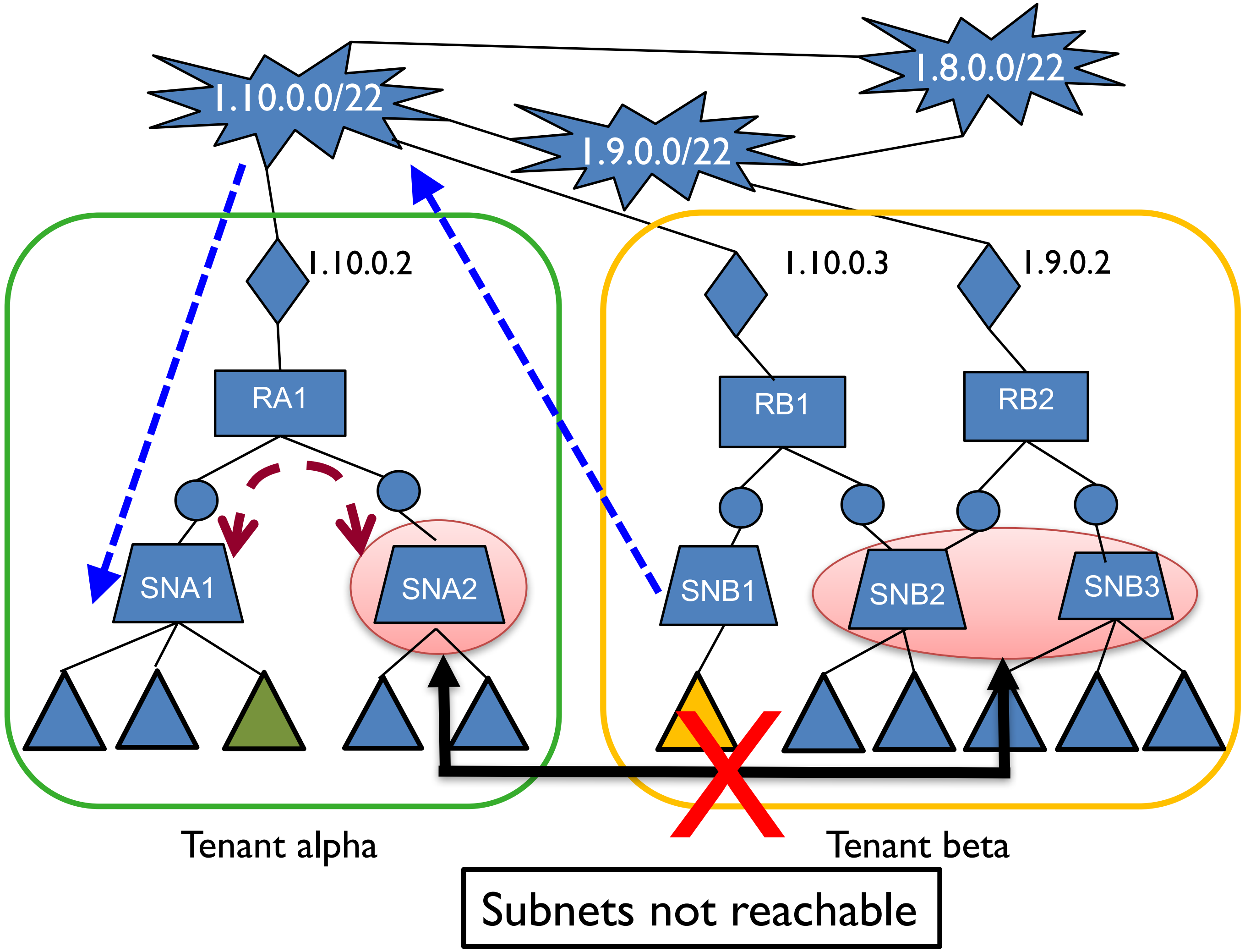
Check isolation between subnets within the same tenant environment

Step two

Check isolation between different tenant environments

Step three

Check VM-isolation only for subnets found to be reachable



Top-Down Verification

Step one

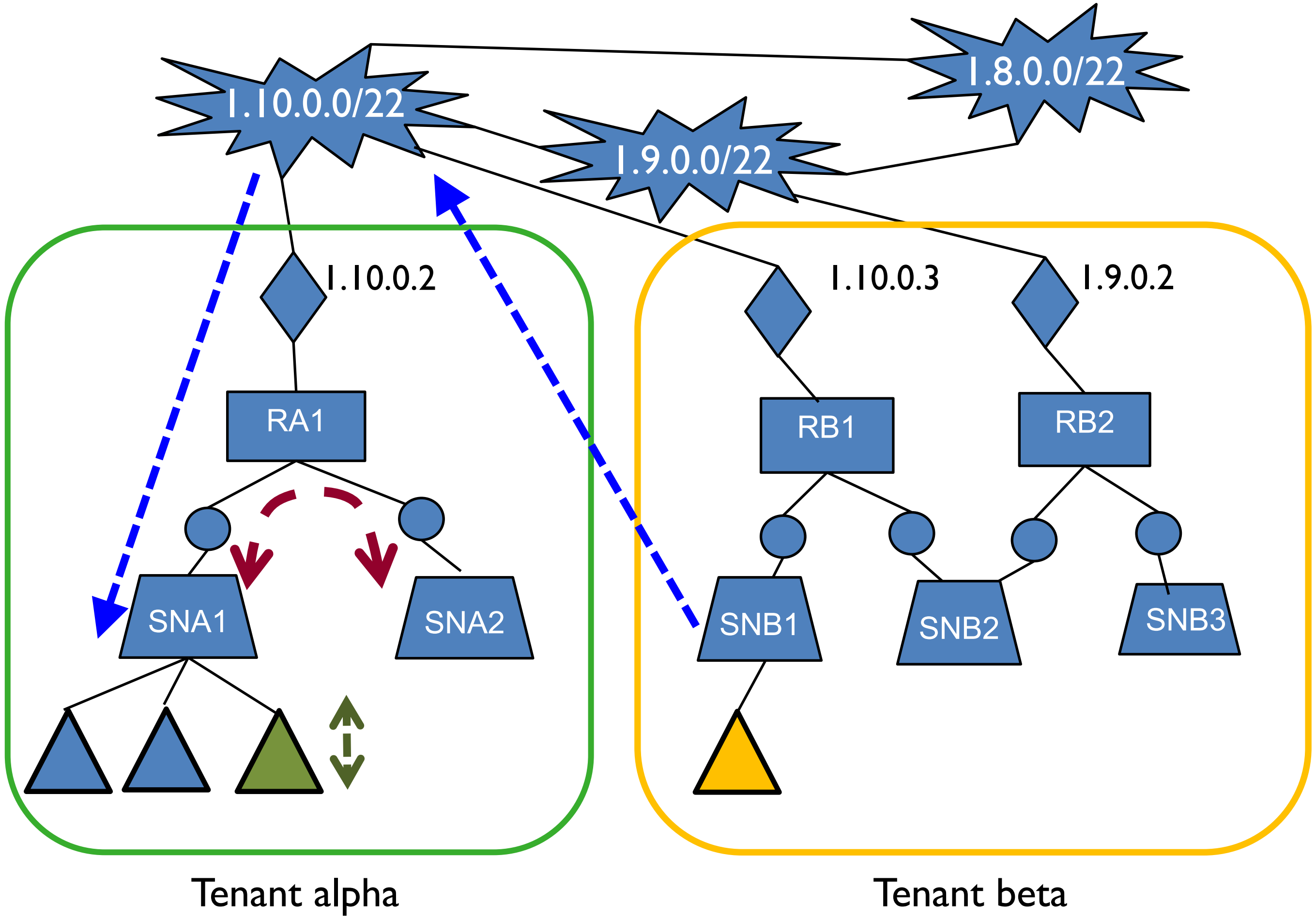
Check isolation between subnets within the same tenant environment

Step two

Check isolation between different tenant environments

Step three

Check VM-isolation only for subnets found to be reachable



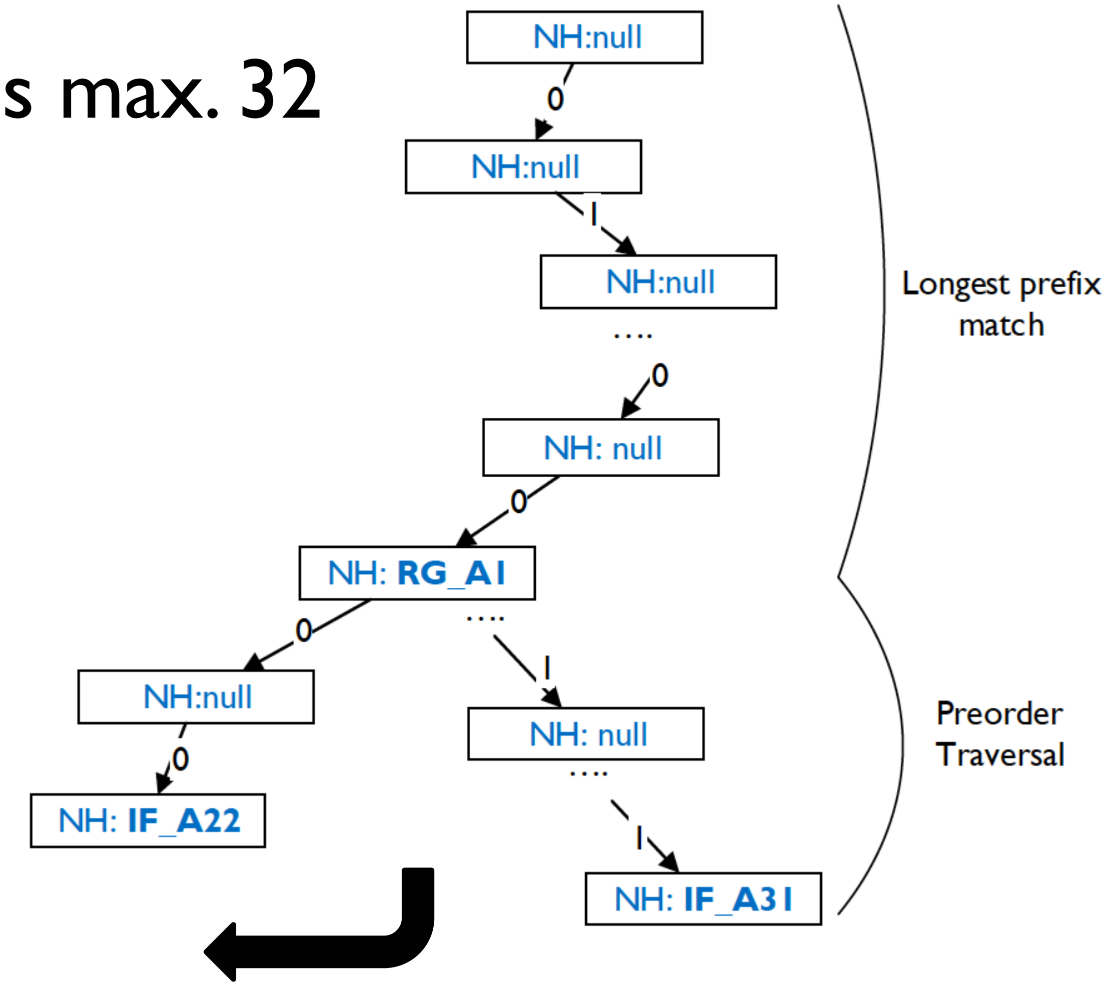
Efficient Data Structure

Capturing Routing Rules

Matching rule is O(L), here L is max. 32

Rules in Router R_A1

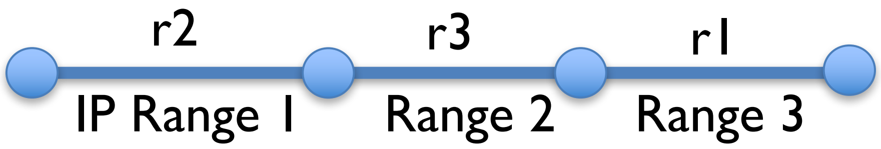
Rule	Prefix	Next-Hop
r0	10.0.1.0/24	IF_A12
r1	1.10.0.0/22	**RG_A1**
r2	1.10.0.0/24	**IF_A22**
r3	1.10.0.0/28	**IF_A31**



NH:null
0
NH:null
1
NH:null
....
0
NH: null
0
NH: **RG_A1**
0
....
1
NH:null
0
NH: **IF_A22**
NH: null
....
1
NH: **IF_A31**

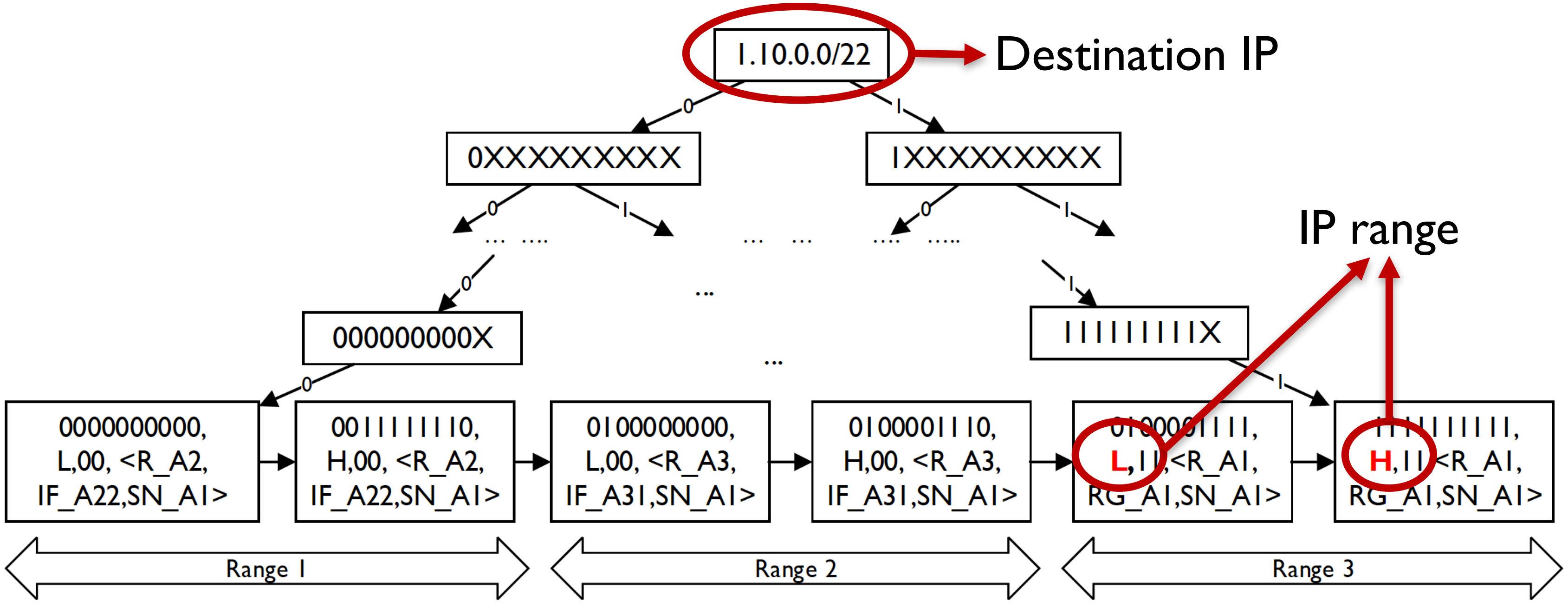
Longest prefix match

Preorder Traversal



Efficient Data Structure
Storing Intermediary Results

- Storing results of matching routing rules against IP ranges
- Searching is O(logL), here L is max. 32

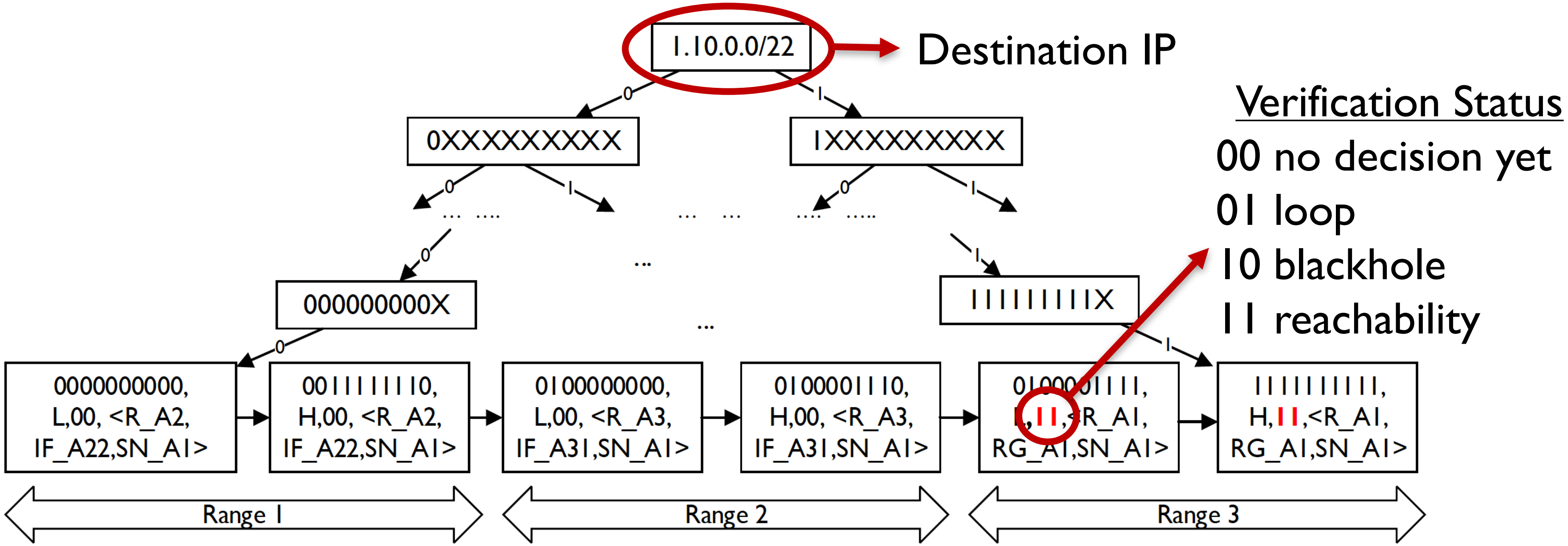


1.10.0.0/22
Destination IP
0
1
0XXXXXXXXX
1XXXXXXXXX
IP range
0
1
0
1
... .
... ...
....
0
...
1
000000000X
111111111X
...
0
1
0000000000, L,00, <R_A2, IF_A22,SN_A1>
0011111110, H,00, <R_A2, IF_A22,SN_A1>
0100000000, L,00, <R_A3, IF_A31,SN_A1>
0100001110, H,00, <R_A3, IF_A31,SN_A1>
0100001111, L,11,<R_A1, RG_A1,SN_A1>
1111111111, H,11,<R_A1, RG_A1,SN_A1>
Range 1
Range 2
Range 3

Efficient Data Structure

Storing Intermediary Results

- Storing results of matching routing rules against IP ranges
- Searching is O(logL), here L is max. 32



1.10.0.0/22

Destination IP

0XXXXXXXXX

1XXXXXXXXX

000000000X

111111111X

0000000000, L,00, <R_A2, IF_A22,SN_A1>

0011111110, H,00, <R_A2, IF_A22,SN_A1>

0100000000, L,00, <R_A3, IF_A31,SN_A1>

0100001110, H,00, <R_A3, IF_A31,SN_A1>

0100001111, L,11,<R_A1, RG_A1,SN_A1>

1111111111, H,11,<R_A1, RG_A1,SN_A1>

Range 1

Range 2

Range 3

Verification Status

00 no decision yet

01 loop

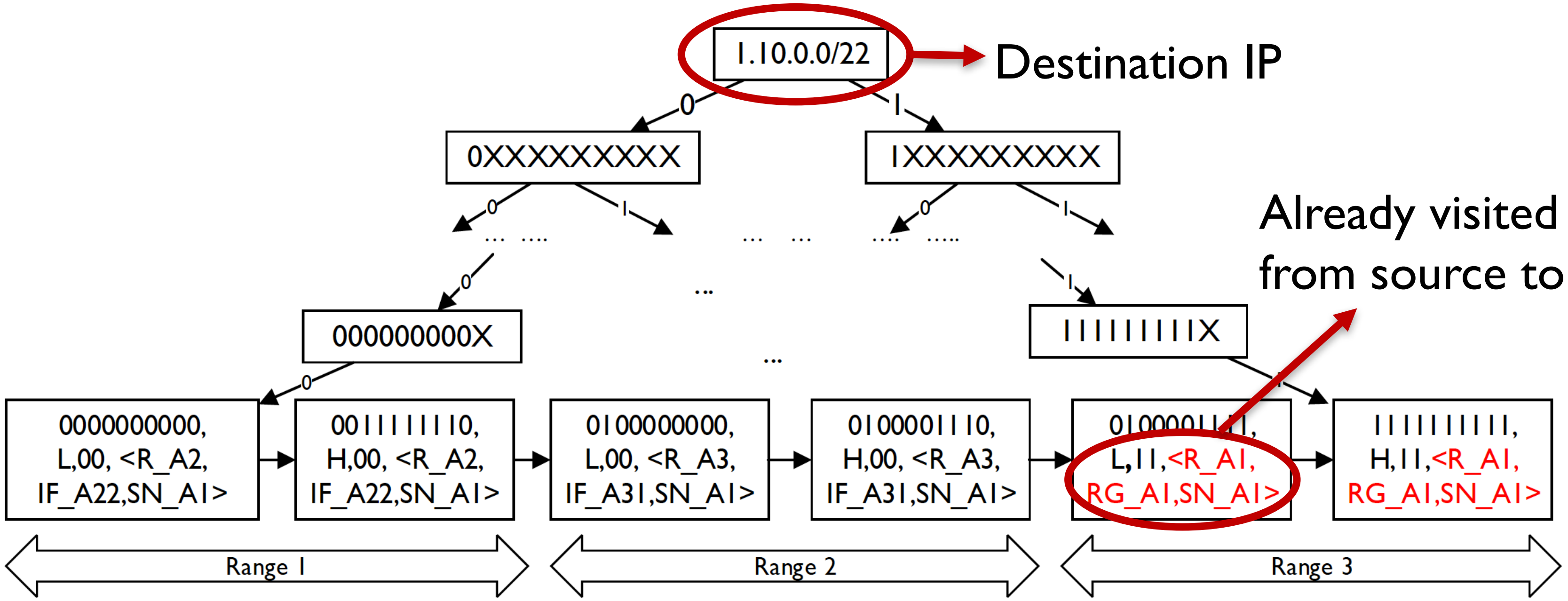
10 blackhole

11 reachability

Efficient Data Structure

Storing Intermediary Results

- Storing results of matching routing rules against IP ranges
- Searching is O(logL), here L is max. 32



Destination IP

Already visited nodes from source to this IP range

Incremental Verification

Graph update

Radix trie creation/deletion

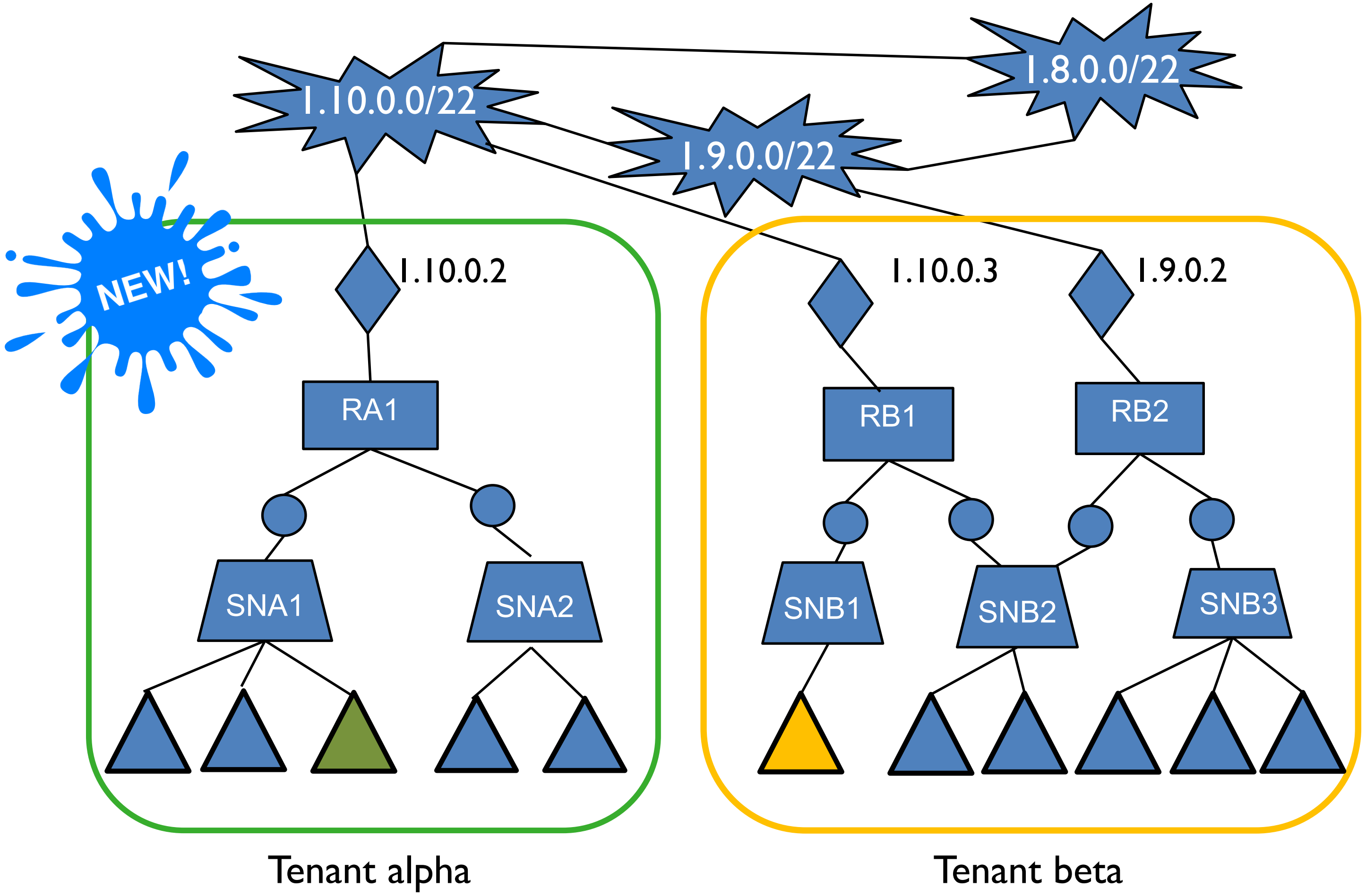
Radix trie update

X-fast trie creation/deletion

X-fast trie update

VM-level isolation verification

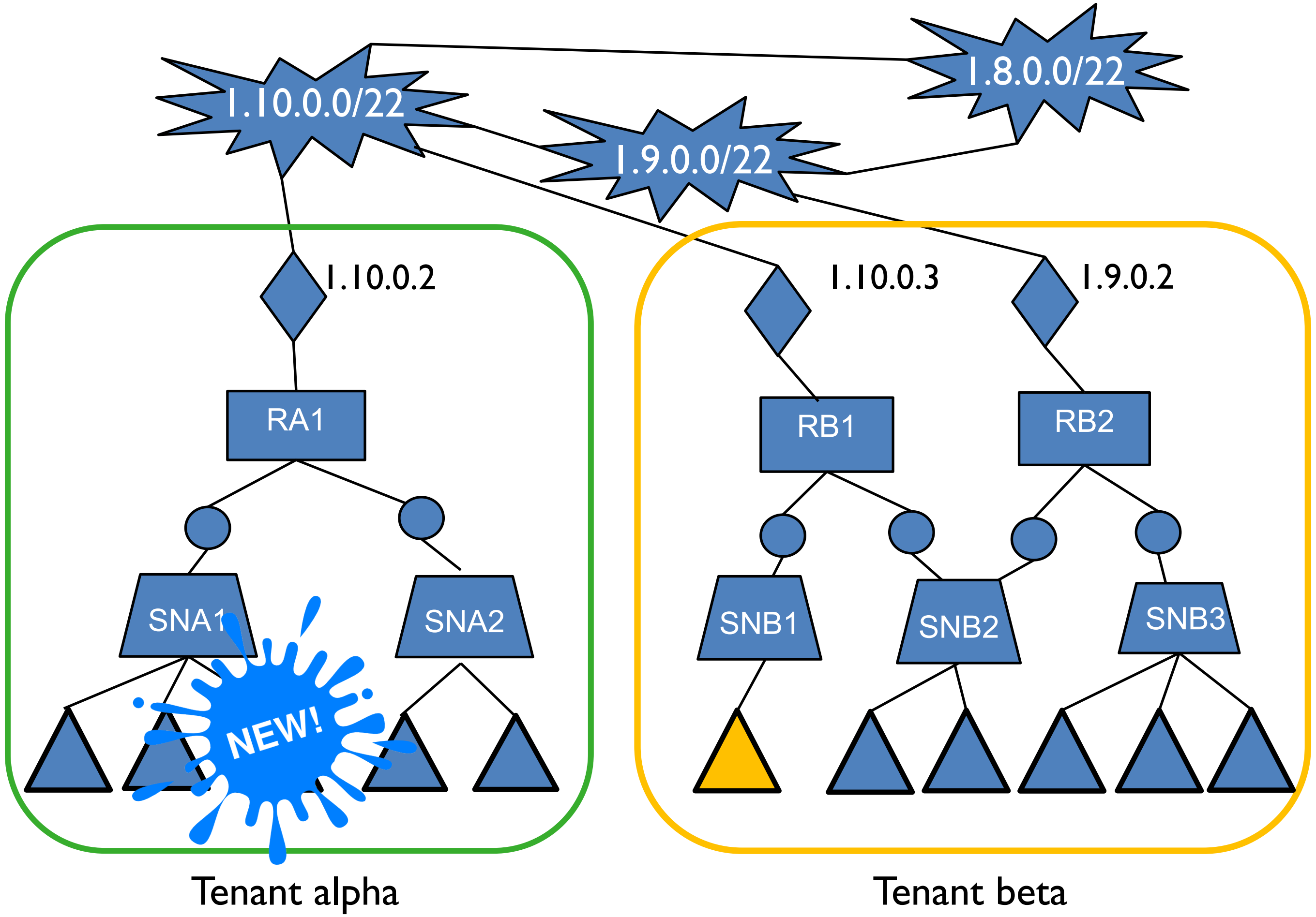
Security group verification



Incremental Verification

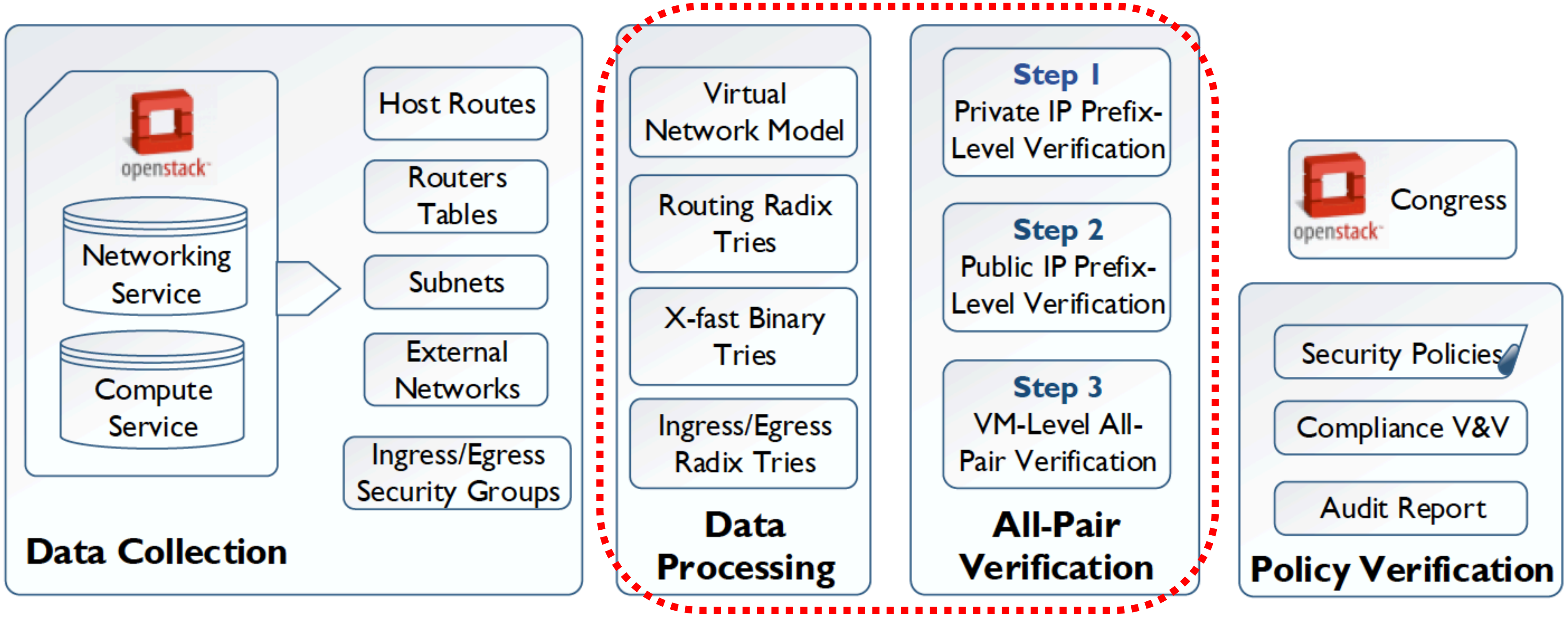
Adding a Security Group

- Graph update
- Radix trie creation/deletion
- Radix trie update
- X-fast trie creation/deletion
- X-fast trie update
- VM-level isolation verification
- Security group verification

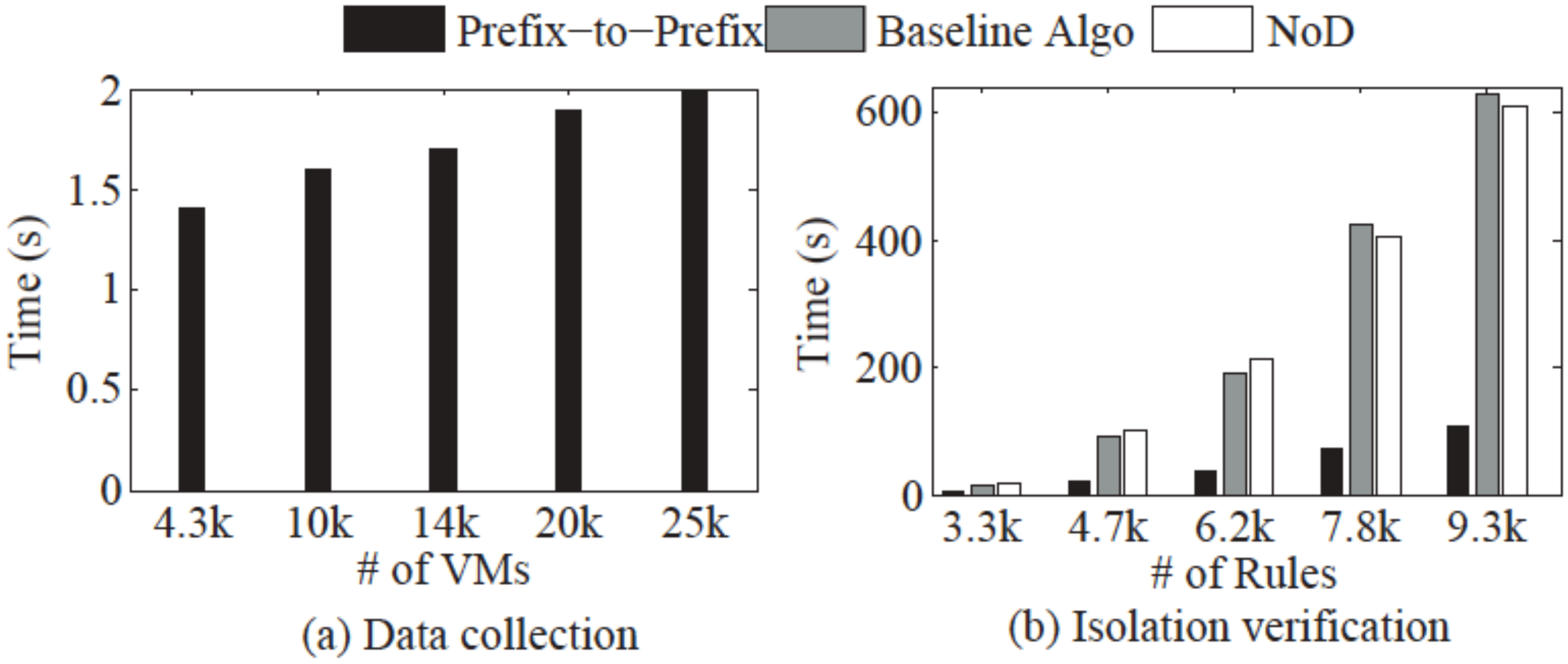


Application to OpenStack

- OpenStack Kilo with one controller and 80 compute nodes
- Parallelization of reachability verification with Apache Ignite
- Integration to OpenStack Congress



Performance Evaluation



(a) Data collection

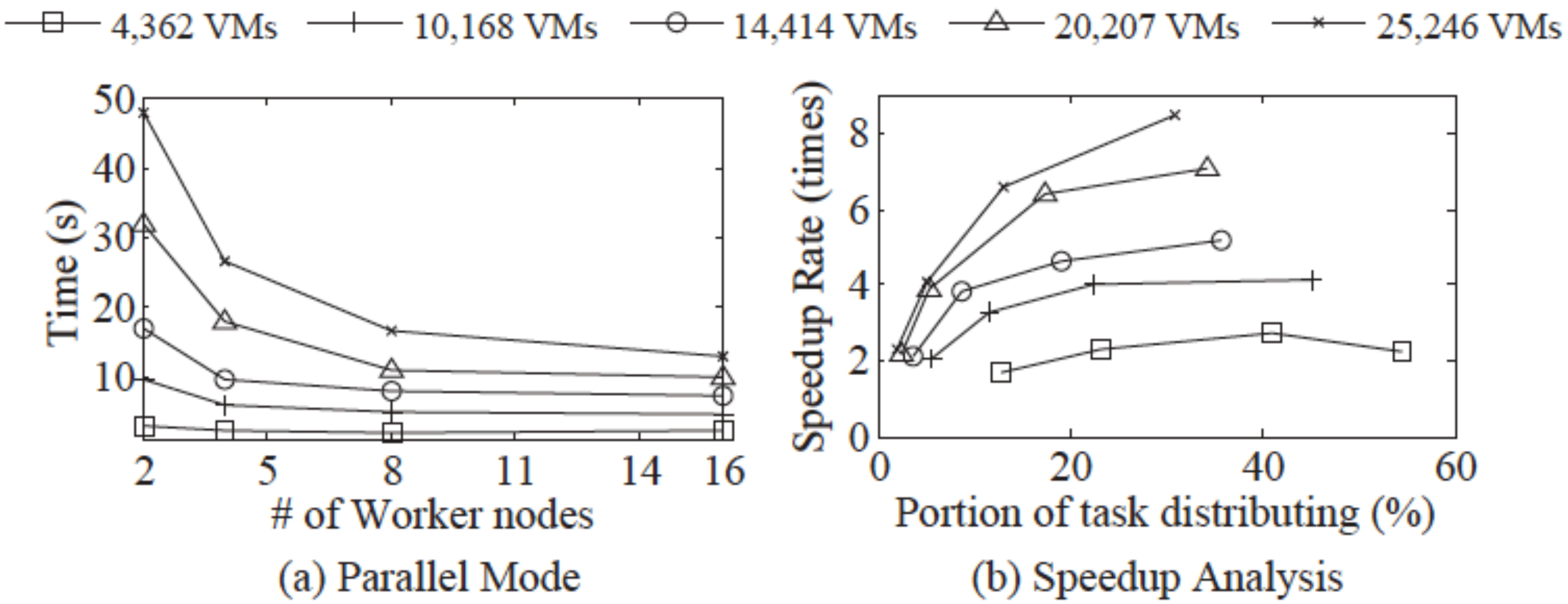
(b) Isolation verification

Data collection and processing time vary from 1.5 to 2 seconds

TenantGuard performs 82% faster than the baseline

N. P. Lopes, N. Bjørner, P. Godefroid, K. Jayaraman, and G. Varghese. Checking beliefs in dynamic networks, NSDI'15.

Further Performance Improvement



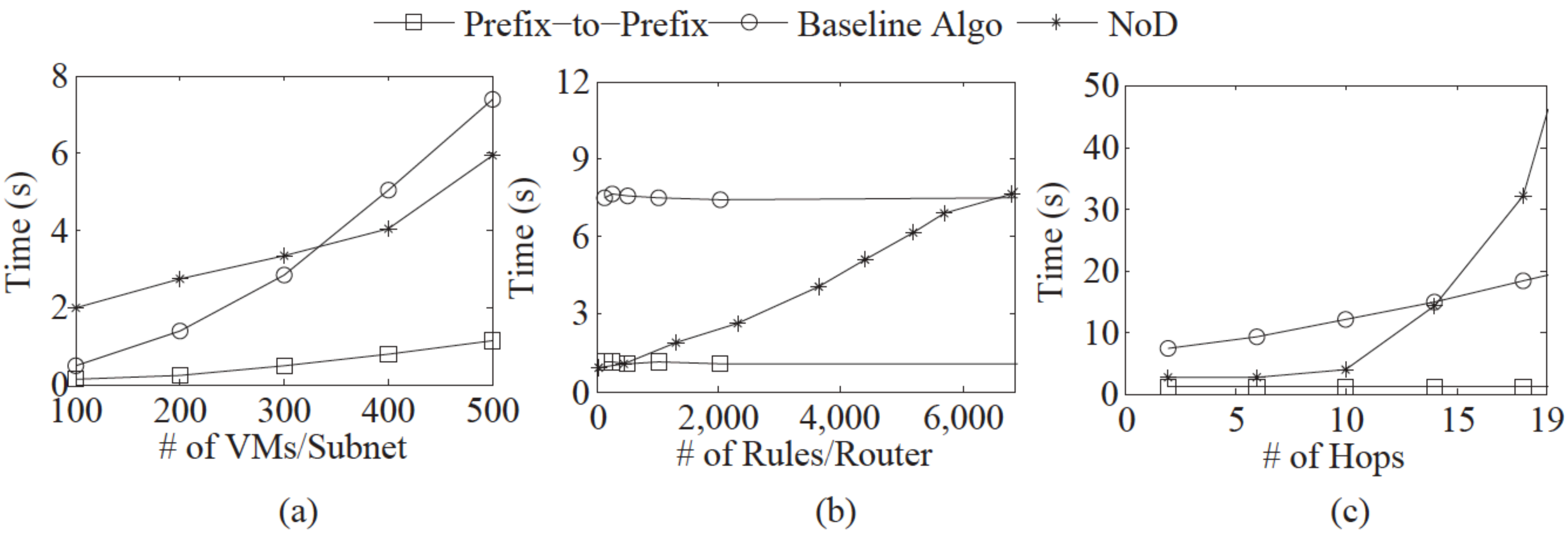
(a) Parallel Mode

(b) Speedup Analysis

Reachability between 168 millions VM pairs in 13 seconds

Relationship between cluster size and speedup gain

Identifying Performance Factors



Number of VMs and hops have less effects due to the reduced complexity and design

Number of routing rules has almost no effect due to the use of Radix and X-fast tries

Conclusion

- Future work
  - Integrating existing tools at other layers (physical, L2)
  - Ensuring integrity of input data
  - Addressing privacy issues from the verification results
- Summary
  - TenantGuard, a VM-level network isolation verification system
  - Integrated our approach to OpenStack
  - Reachability for over 150 million VM pairs in 13 seconds

Project webpage: arc.encs.concordia.ca

Corresponding author: Suryadipta Majumdar (su_majum@encs.concordia.ca)

Thank you

Backups

Experimental Settings

- Test Environment
  - Two series of datastes
    - SNET (represents small to medium networks)
    - LNET (represents large networks)
  - NoD (NSDI'15) and a baseline algorithm
- Real Cloud
  - Ericsson research cloud
  - Mainly to evaluate the real world applicability of TenantGuard
  - Only observed a minor incompatibility issue due to version mismatch