LANCER Site Visit

Outline

- [15 minutes] Introduction (Nate & Wen)
  - Team Introductions
  - Technical Approach
- [10 minutes] Progress Since Kick-Off (Nate & Wen)
  - Executive Summary
  - Planned Trajectory for end of Phase I
- [15 minutes] Collaboration Efforts (Nate & Rebecca)
  - CAGE
  - Talking to Kryptowire
  - Network Action Space
- [60 minutes] Early Results
  - [20 minutes] NetKAT (Jules & Nate)
  - [30 minutes] Inverse RL (Nico/Rebecca & Wen)
  - [10 minutes] Aether: Pronto + OnRamp (Hussain & Nate)
- [20 minutes] Response to Crawl Questions (Everyone)
- [30 minutes] Budget & Contracting (Shailja & Nate)

Introduction

Progress

Progress

- Got going with Kryptowire TA1 Platform
- Started development using CAGE 2
- Started Modeling Red Agents Using Inverse RL
- Fast NetKAT implementation
- Standing Up Aether OnRamp

Trajectory

- Crawl (6 month)
- Walk (6 month)
- Run (6 month)

Collaboration Efforts

Cage Challenge Overview

- Scenario of a network attack
- Goal: Blue Agent (defensive) stops the Red Agent (malicious) without disrupting the Green Agent (normal users)
- Integrated with CybORG, a reinforcement learning gym

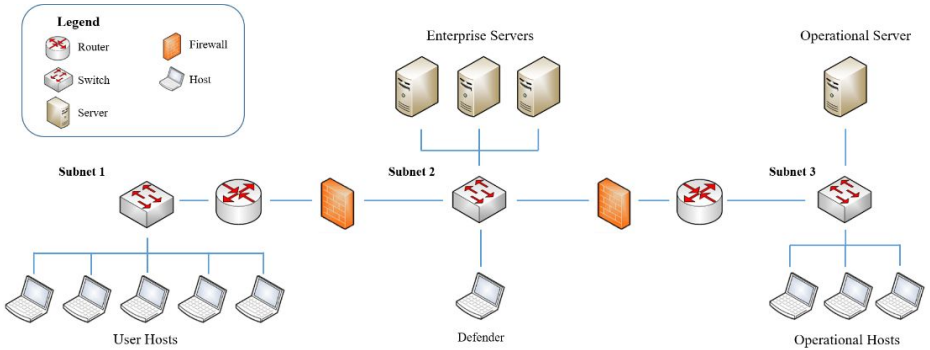


Figure 1: Network of the scenario and challenge problem (Cage Challenge 2)

Red Agent Actions

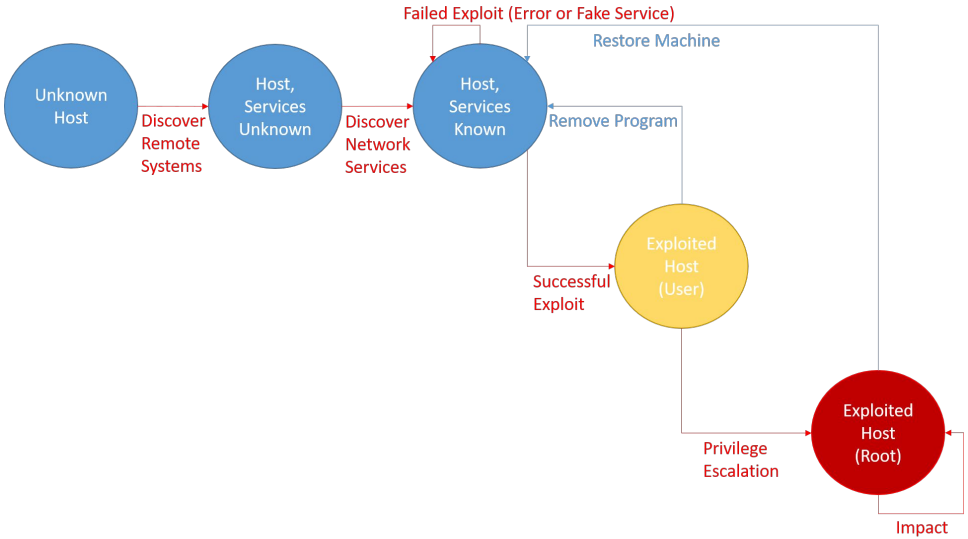


Figure 2: Effect of actions on host state (Cage Challenge 2)

Early Results

RL Outline

- Platform - describe cage challenge
- Dataset
- Model baselines (BC/SL has these limitations…)
- IRL results
  - Focus on IRL, no progress on RL side yet
- Plan moving forward
- Tables with data, add graphical aids

Discuss overall approach

Why are we using imitation learning to learn the red agent

Definitions

Define RL terminology: states, actions, observations

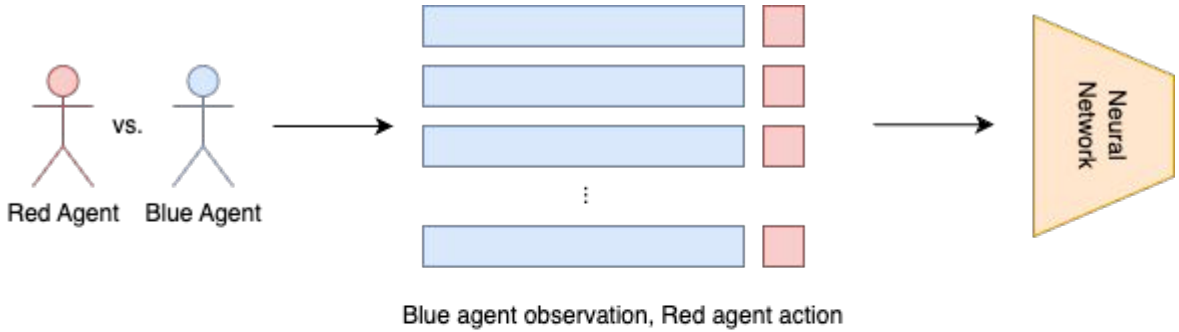
Imitation Learning Approaches

1. Behavior Cloning (BC)
    a.
2. Generative Adversarial Imitation Learning (GAIL)
    a.

Behavior Cloning (BC)

1. Collected data from environment with Blue, Green, Red agents
    a. (Blue agent observation, Red agent action)
2. Used data to train a neural network to predict the Red agent action from the Blue agent observation
3. Created a Red agent that used this neural network to determine the next action (learned agent)
4. Collected reward from environment with a Blue agent, Green agent, and the learned agent

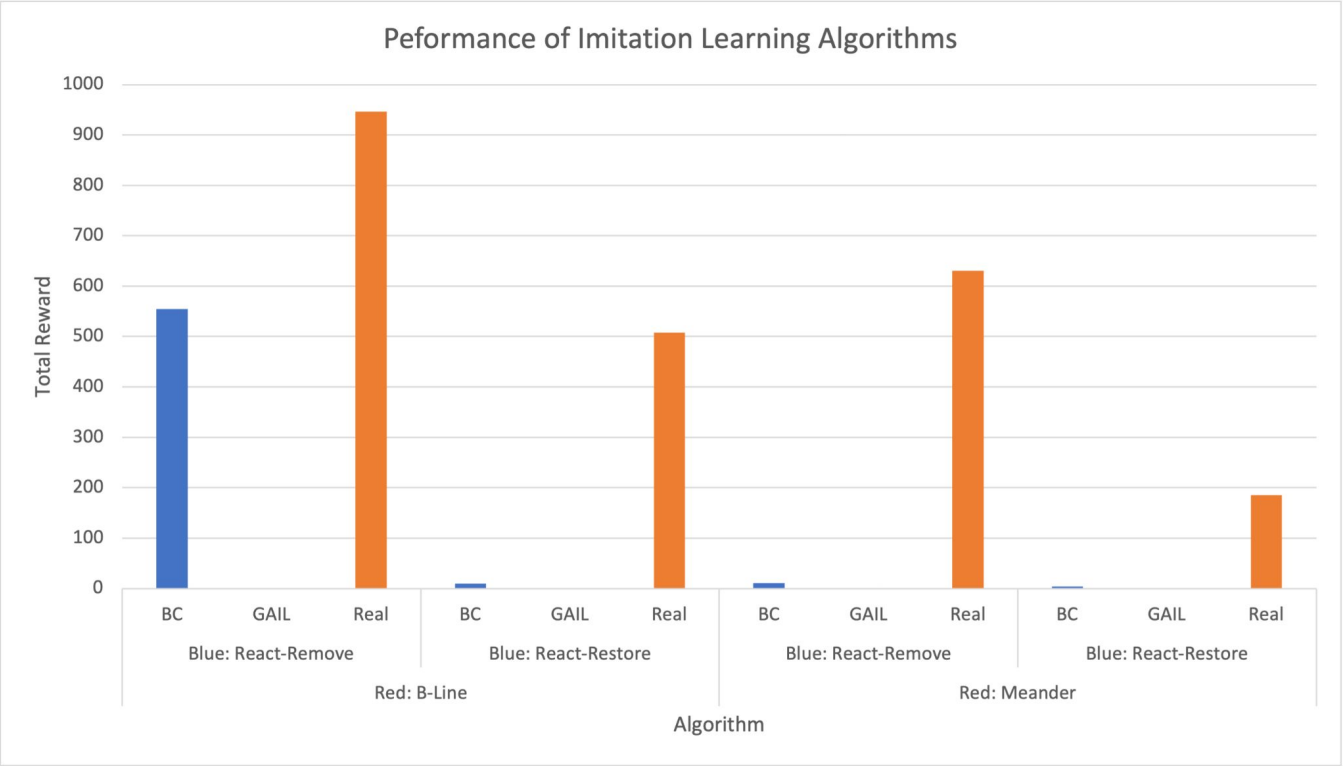
Behavior Cloning



Generative Adversarial Imitation Learning (GAIL)

Discuss distribution shift, use self driving visual examples

Results


Peformance of Imitation Learning Algorithms
Total Reward
1000
900
800
700
600
500
400
300
200
100
0
BC
GAIL
Real
BC
GAIL
Real
BC
GAIL
Real
BC
GAIL
Real
Blue: React-Remove
Blue: React-Restore
Blue: React-Remove
Blue: React-Restore
Red: B-Line
Red: Meander
Algorithm

Detailed results for IRL

Plot showing BC improvement with increasing number_of_states input

BC: 0 Prev. States, 50 eval. Episodes

Red Agent	Blue Agent	Steps per Episode	Dataset Collection Episodes	Train Loss	Train Accuracy	Validation Accuracy	Average Reward (50 trials)	Standard Deviation	Real Agent Average Reward (3 runs)	Real Agent Average Std. Dev.
B-Line	React Remove	100	100	0.16	0.95	0.93	556	361	947	193
B-Line	React Restore	100	100	0.64	0.77	0.77	-10.0	0.0	508	366
Meander	React Remove	100	100	0.71	0.72	0.67	11.1	39.5	630	259
Meander	React Restore	100	100	1.1	0.56	0.53	3.55	7.77	185	210

BC: 3 Prev. States, 50 eval. Episodes

Red Agent	Blue Agent	Steps per Episode	Dataset Collection Episodes	Train Loss	Train Accuracy	Val Accuracy	Average Reward (50 trials)	Standard Deviation	Real Agent Average Reward (3 runs)	Real Agent Average Std. Dev.
B_Line	React Remove	100	100	0.038	0.986	0.967	693.5753333	304.785803	946.5733333	192.988038
B-Line	React Restore	100	100	0.0372	0.987	0.965	483.7273333	335.9844907	508.29	366.4229701
Meander	React Remove	100	100	0.327	0.870	0.710	254.5386667	245.9676103	630.17	258.94
Meander	React Restore	100	100	0.615	0.762	0.587	77.03866667	141.286532	185.01	209.86

Next Steps

1. New IRL algorithms for improving modeling red agents;
2. Training RL agents against the learned red agents

NetKAT

KATch

A Fast Symbolic Verifier for NetKAT

Mark Moeller, Jules Jacobs, Nate Foster, Alexandra Silva (Cornell), Olivier Savary Belanger, David Darais, Cole Schlesinger (Galois), Steffen Smolka (Google)

The Control Plane and Network Defense Agents

Control Plane

- Computes routing tables
- Ensures network connectivity
- Enforces network policies

Network Defense Agents

- Detects and responds to network attacks
- Example: Security breach containment
- Example: DDoS mitigation
- **Action space?**
- **Modify routing tables?**

Neural and Symbolic AI

Neural AI Strengths

- Excellent pattern recognition
- Learns from experience
- Adaptability to new situations
- Suitable when explicit programming is difficult

Symbolic AI Strengths

- Excellent reasoning and planning
- Guarantees correctness
- Verifiable and explainable
- Ideal when strict compliance with rules is required

Neural AI and Symbolic AI in Network Defense

Neural AI

- Utilizes deep learning for real-time attack detection and response
- Adapts to evolving network threats
- Modifies routing tables dynamically
- Example: Detecting and rerouting traffic to mitigate DDoS attacks
- Example: Detecting and isolating compromised hosts

Symbolic AI

- Computes consequences of routing changes
- Ensures correctness of routing tables
- Verifies adherence to network policies and security rules
- Example: Validating routing paths for security compliance
- Example: Verifying reachability of critical network services

NetKAT

Network specification language for SDN
Verification of network policies

- Security properties, e.g. slice isolation
- Operational properties, e.g. reachability
- Verified in a common framework

Problem: NetKAT verification is slow
Not suitable for real-time network defense

KATch

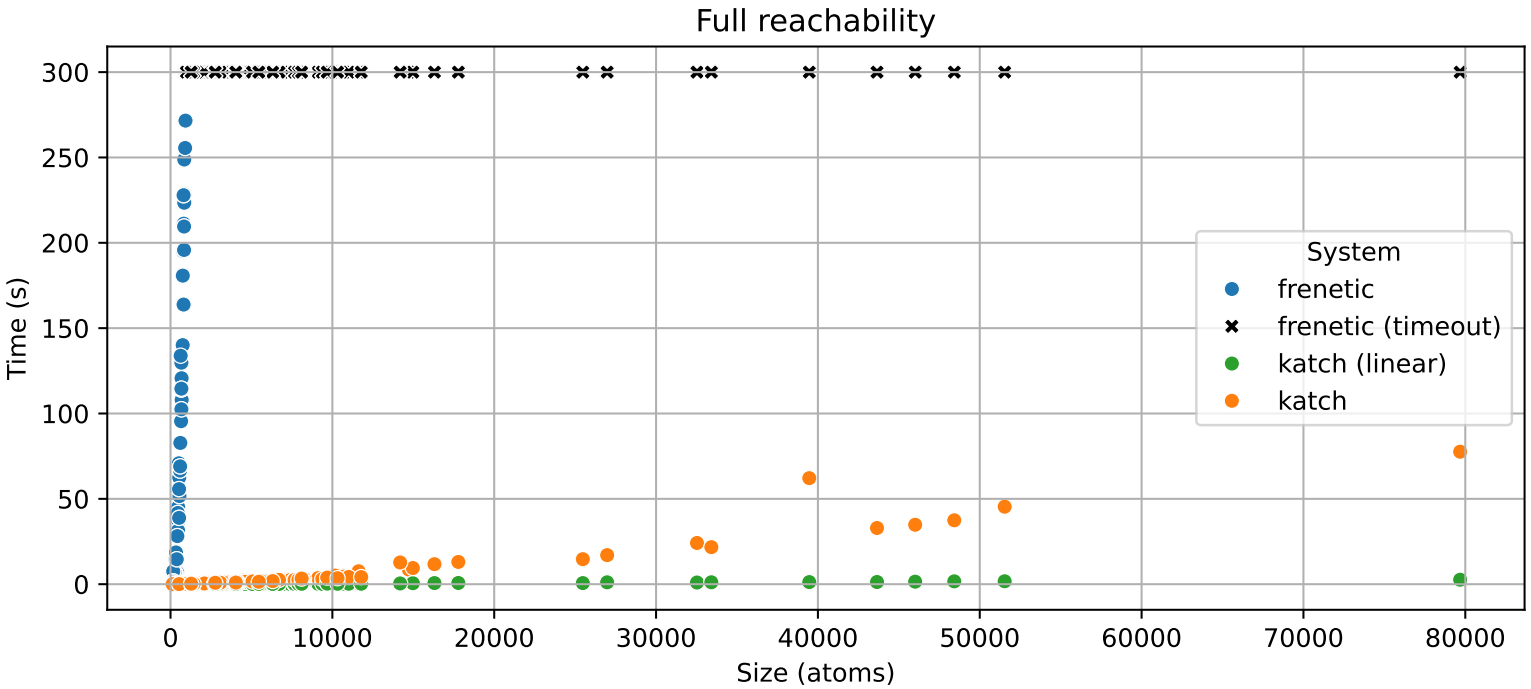
A Fast Symbolic Verifier for NetKAT

KATch

A new NetKAT verifier that is

- **Fast:** 1000× faster
- **Symbolic:** explains verification failures
- **Scalable:** handles larger networks

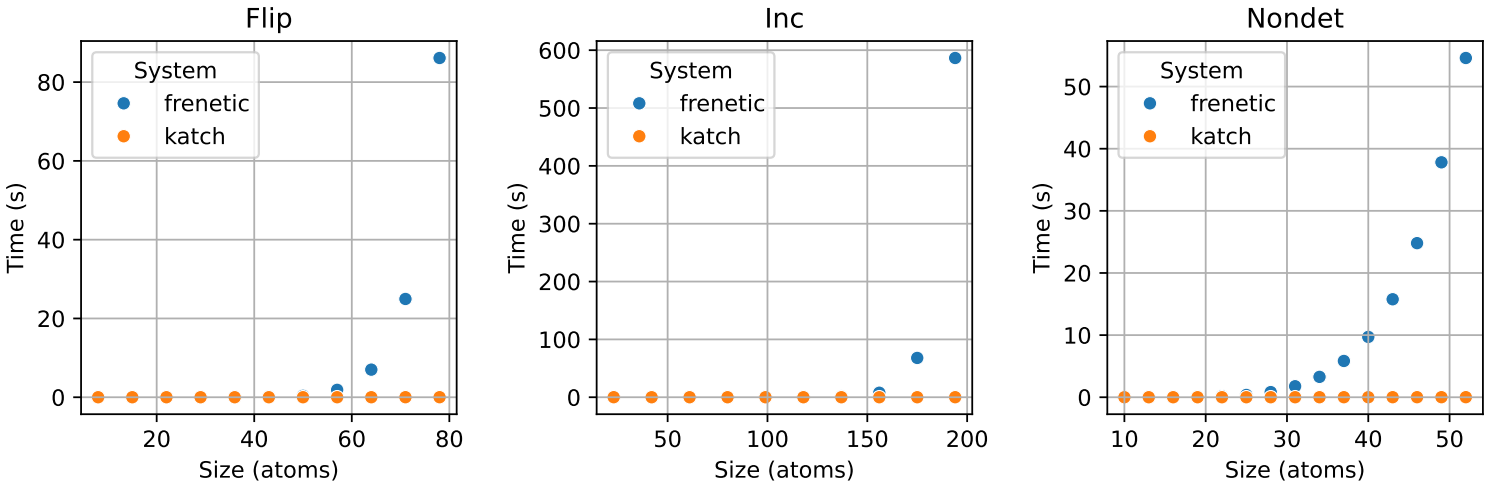
Full Reachability



Detailed comparison: (un)reachability and slice isolation

Name	Size (atoms)	Reachability		Unreachability		Slicing		Min Speedup
		KATch	Frenetic	KATch	Frenetic	KATch	Frenetic	
Layer42	135	0.00	0.04	0.00	0.04	0.01	0.07	7×
Compuserv	539	0.01	0.36	0.01	0.38	0.01	0.85	36×
Airtel	785	0.01	0.83	0.01	0.84	0.02	2.08	83×
Belnet	1388	0.01	3.17	0.01	3.16	0.04	7.99	200×
Shentel	1865	0.02	4.01	0.02	4.00	0.04	9.80	200×
Arpa	1964	0.01	4.32	0.02	4.32	0.05	10.99	216×
Sanet	4100	0.04	23.46	0.03	25.23	0.12	62.70	522×
Uunet	5456	0.04	81.54	0.04	81.92	0.15	204.85	1366×
Missouri	9680	0.11	161.28	0.10	165.85	0.27	519.46	1658×
Telcove	10720	0.09	464.15	0.08	465.27	0.28	1274.24	4551×
Deltacom	27092	0.31	2392.56	0.30	2523.03	0.75	7069.54	7718×
Cogentco	79682	0.97	22581.39	0.88	23300.87	1.78	53066.82	23280×

Synthetic combinatorial benchmarks



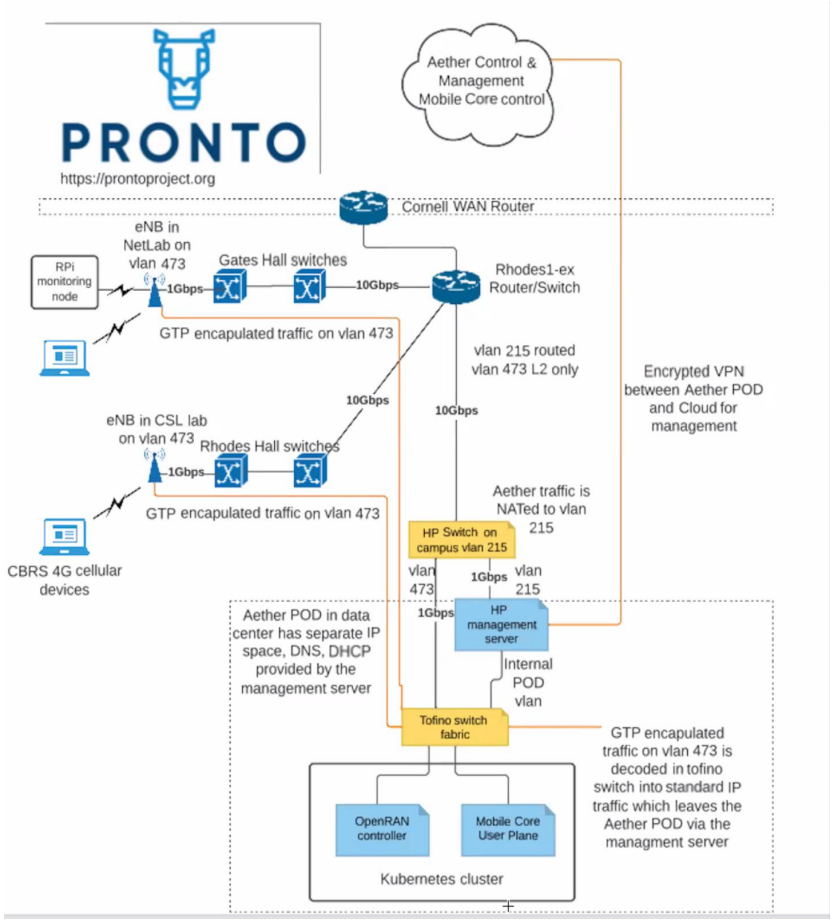
Conclusion

NetKAT verification can be fast

Can we combine neural and symbolic AI?

Pronto Cornell Network

Previous Cornell Network



AetherOnRamp

Private Enterprise 5G network

- operational cluster that is capable of running 24/7 and supports live workloads on Cornell Network.

Mobile Network two main subsystems :

1. RAN - manages radio resources(spectrum)
2. Mobile Core - provide packet data network to mobile subscribers

Aether

Pronto pods Demo

- Demo setup with the end to end connectivity
  - End to end connectivity, Raspberry PI reachable from the Intel server
  - Grafana Dashboards
    - Gov for traffic, syncing from the UEs
- Working Demo -
  - MotoG phone connecting to the Aether APN(Access Point Name) within Pronto Network

Aether OnRamp Progress

- Demo the setup, ideally with UEs and end to end connectivity
  - Current issues - Radio connectivity not syncing with the local Aether Core
  - Demo Setup - Grafana Dashboards

Establish Secure Channel and Communication

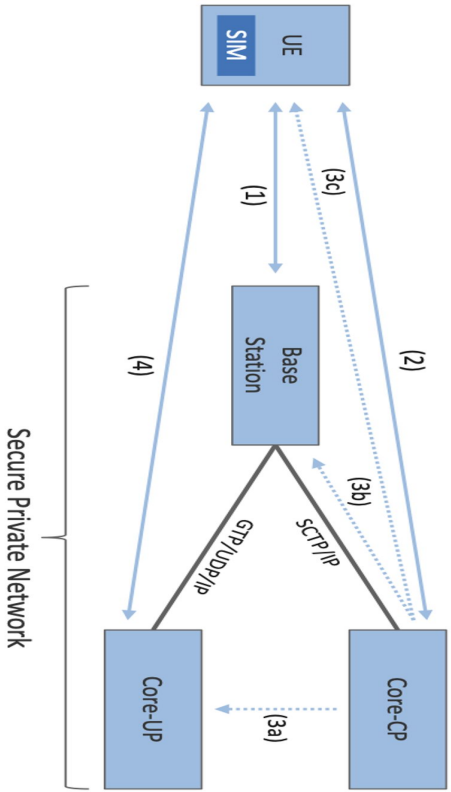
Each SIM has global identifier :

- IMSI - unique 15 digit code

UE registration with Core :

- Communicates with nearby station
- Base station forwards request to the control plane, only gets auth relevant mapping in Core.

UE traffic routed using user plane



Crawl Questions

- Learn about one another's approaches, find integration points, and collaborate on shared infrastructure
- What network should we model first and what workflows should be present?
- What agent actions will be simulated and executed?
- What is a ‘good’ resiliency criteria and how will we judge whether your approach is successful?
- What data types are needed for each performer and what data can be provided by each performer?
  - Data for attackers
  - Reward function for defenders (domain knowledge, Inverse RL)
- How do we collaborate on API design and code interfaces?
- What open-source technology can enable an end-to-end integration demo quickly?
- Who is the intended operator of your approach and what is the desired impact/benefit to their job?