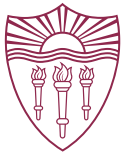


SABRES:
Secure, Adaptive, roBust, Resilient, and Efficient Slices

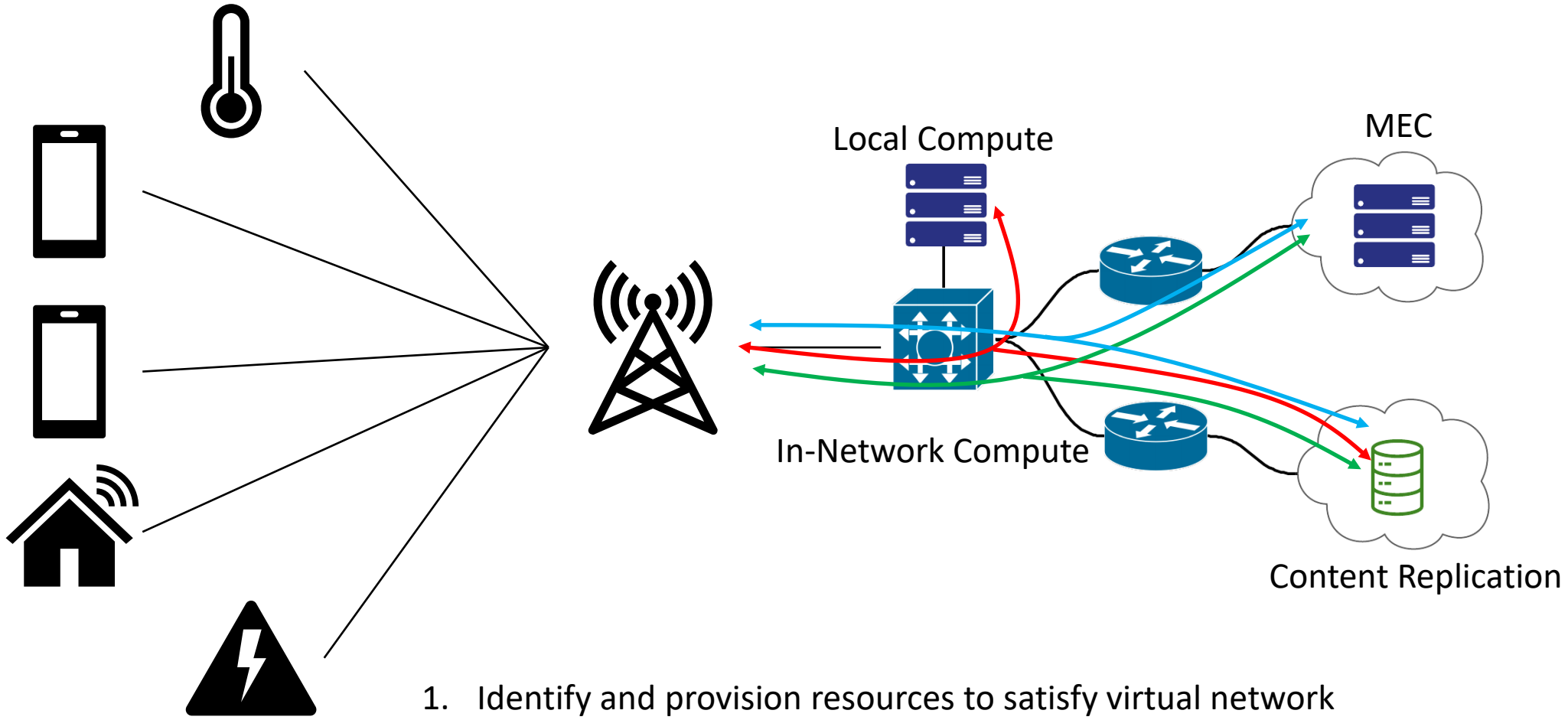
Erik Kline

University of Southern California, Information Sciences Institute

In partnership with: Duality Technologies, Lumen Technologies

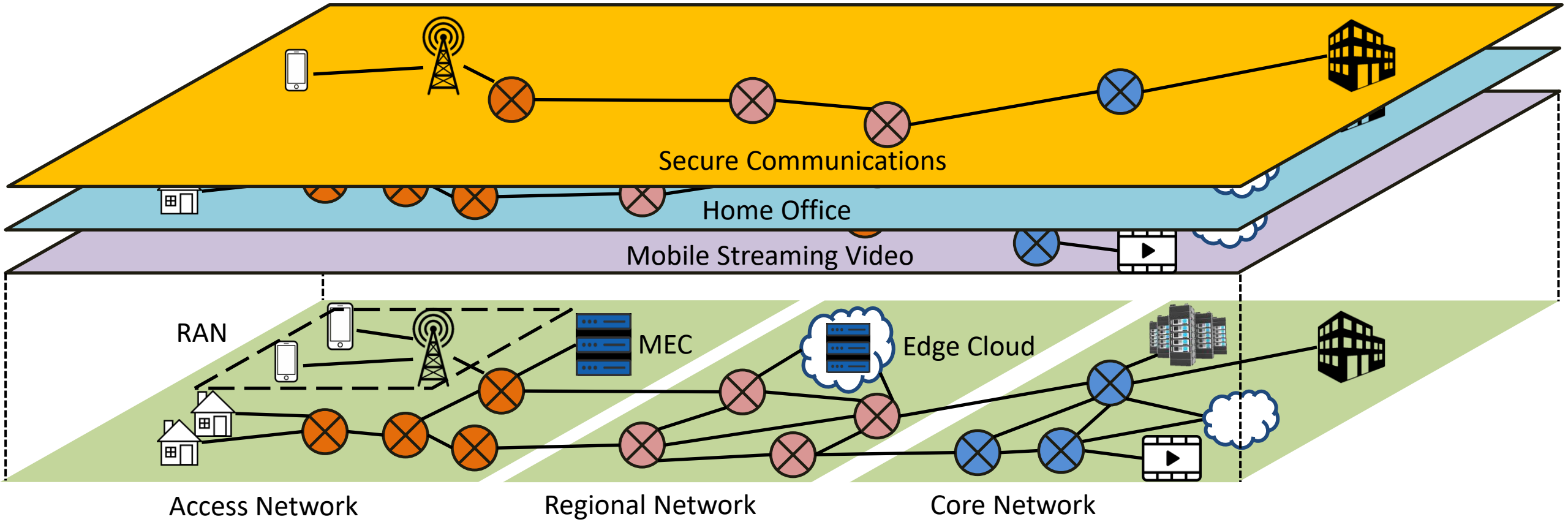
This research is developed with funding from the Defense Advanced Research Projects Agency (DARPA).
The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government

What SABRES is Trying to Do

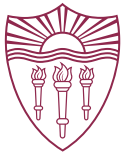


1. Identify and provision resources to satisfy virtual network
2. Validate that the slices constraints are met
3. Provide enhanced security and privacy

Incorporating the Full Network

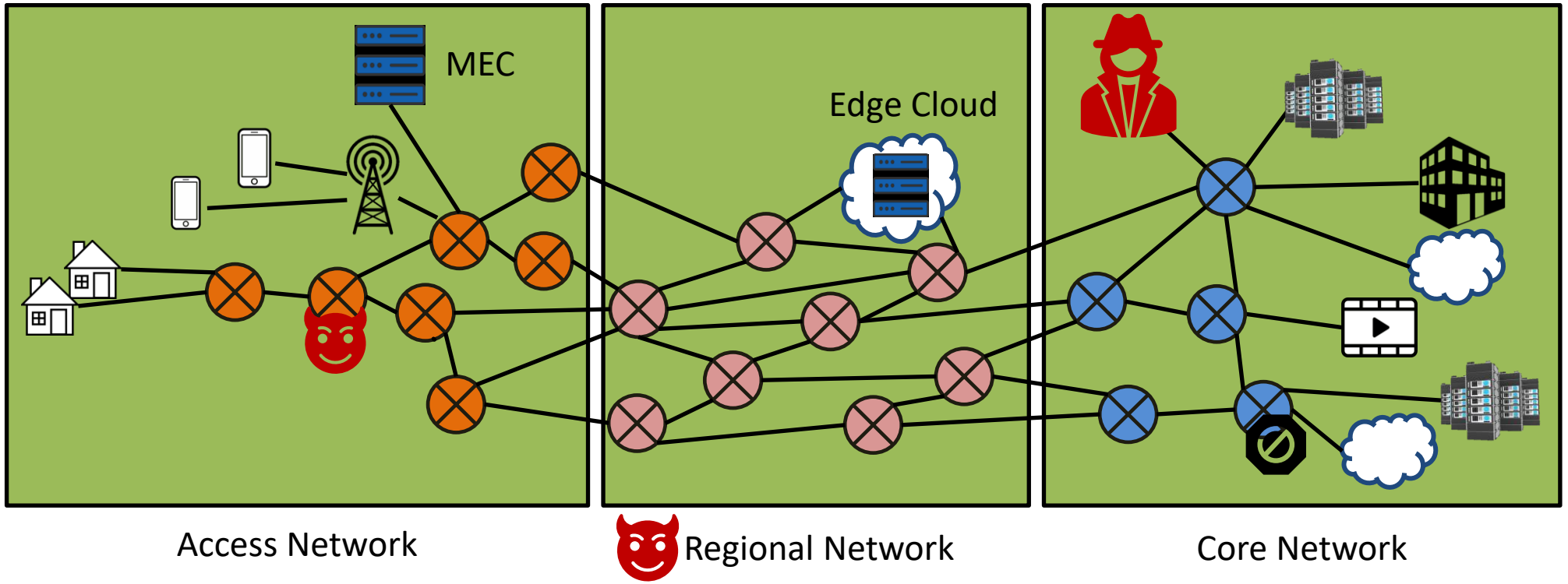


Goals



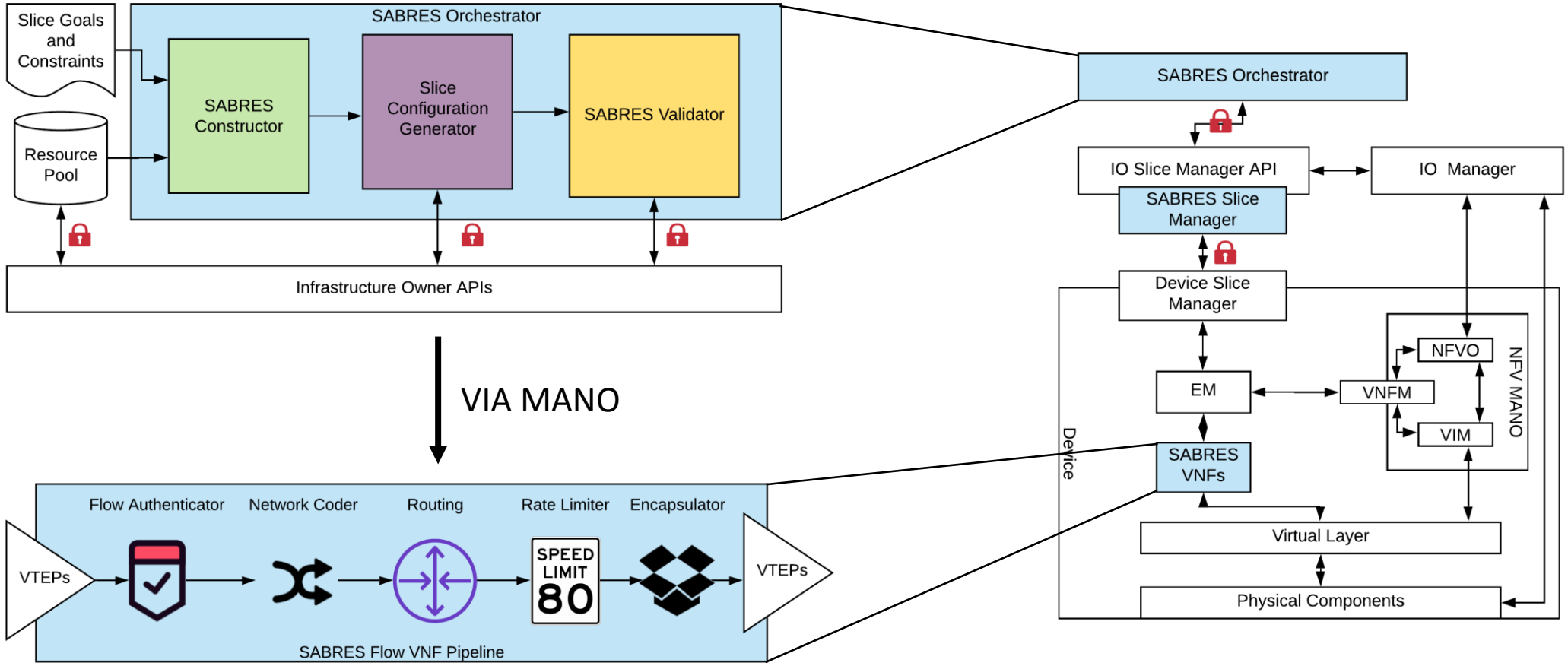
- Quickly determine and provision satisfiable slices of any size
  - Enable slices that consider many constraints, including quality of service, hardware/software properties, location, and security concerns
  - Slices should attempt to minimize resource usage
- Verify slice correctness
  - Ensure that the correct resources are provisioned, and packets are properly forwarded
  - Validate that the constraints are satisfied
  - Rapidly detect constraint failures
- Provide security guarantees for slice operators and infrastructure owners
  - Limit exposure of information to other, co-resident slices
  - Prevent leakage of information due to compromise or side-channels
  - Only allow access to the slice for authorized devices

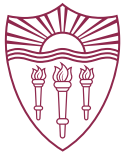
Assumptions



1. Not all devices on a path are under the control of the SABRES
2. SABRES devices may be subverted or fail
3. Multiple Potential Threats

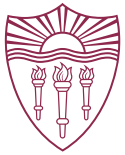
Architecture Diagram





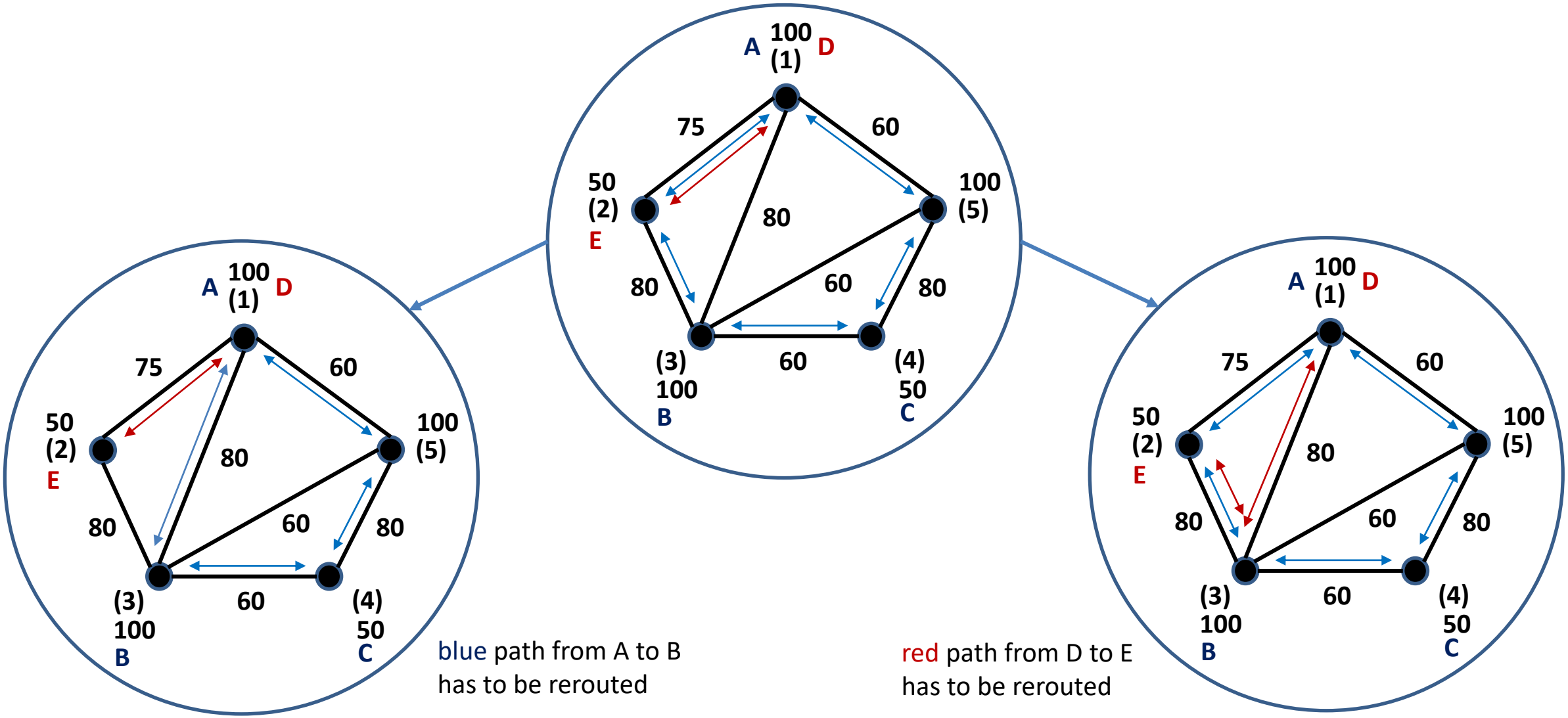
SLICE CONSTRUCTION

Constructing a Network Slice



- Inputs
  - Target Services (Sources/Sinks)
  - Constraints for the slice
  - Substrate network
- Step 1: Construct the virtual network request
  - Transform the constraints to a virtual network topology
  - VNR generation informed by application goals and security posture
- Step 2: Embed VNR into substrate
  - NP hard problem
  - Heuristically solved via Conflict-Based Search (CBS)
- Step 3: Deploy slice configuration via MANO (or alternative)

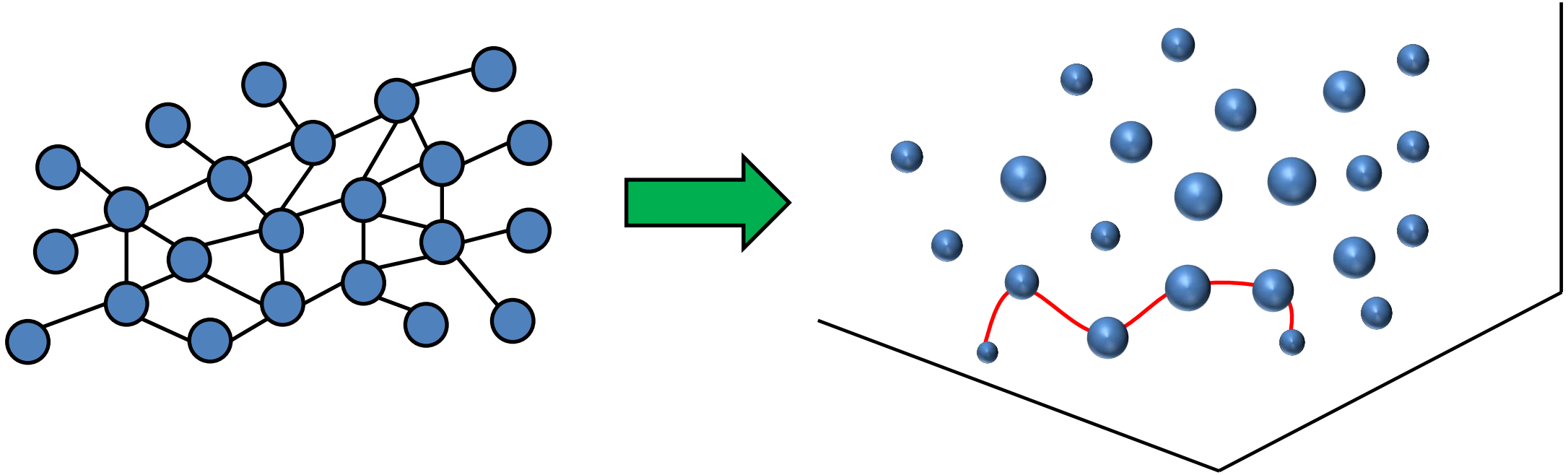
Conflict Based Search for VNE



blue path from A to B has to be rerouted

red path from D to E has to be rerouted

Low Level Embedding via FastMap

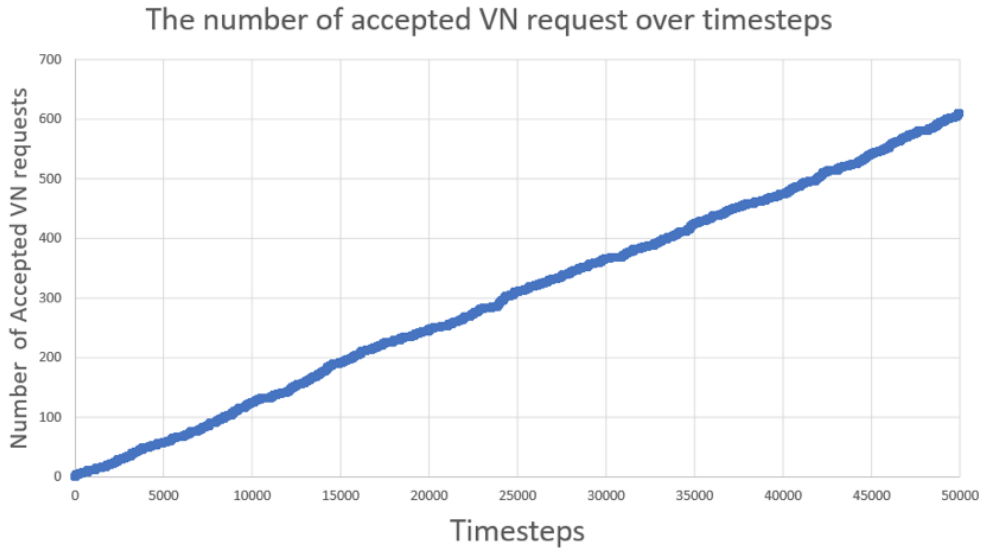


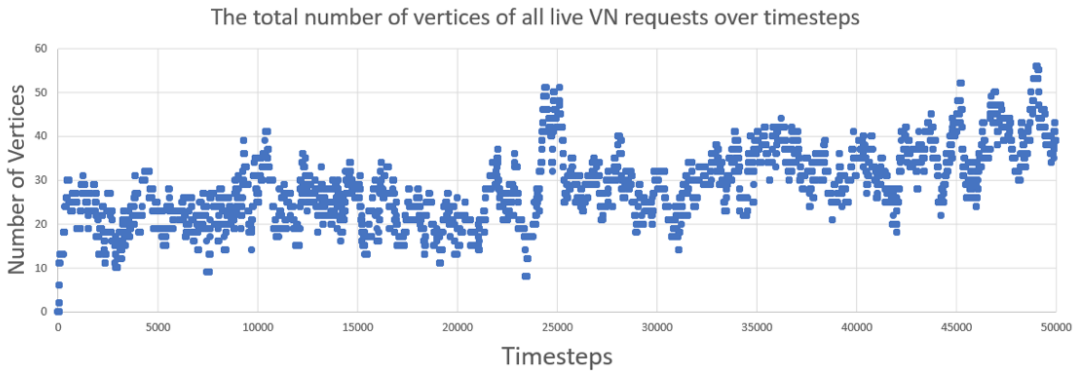
SABRES transforms the underlying graph into N-dimensional Euclidean space using FastMap

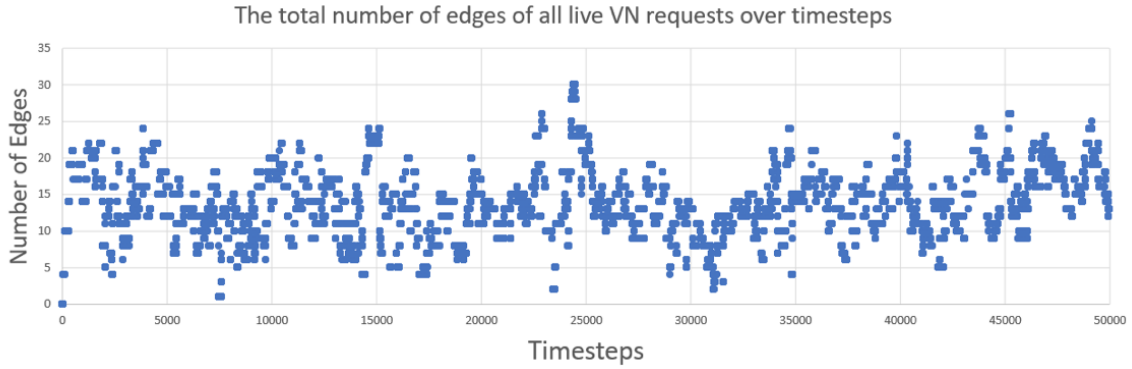
- Finds farthest pair of vertices using a pivot-changing heuristic
- Embeds the graph into n-dimensional space through a triangular projection
- Enables the use of directed search algorithms for shortest path computations, such as A*

Experimental Results

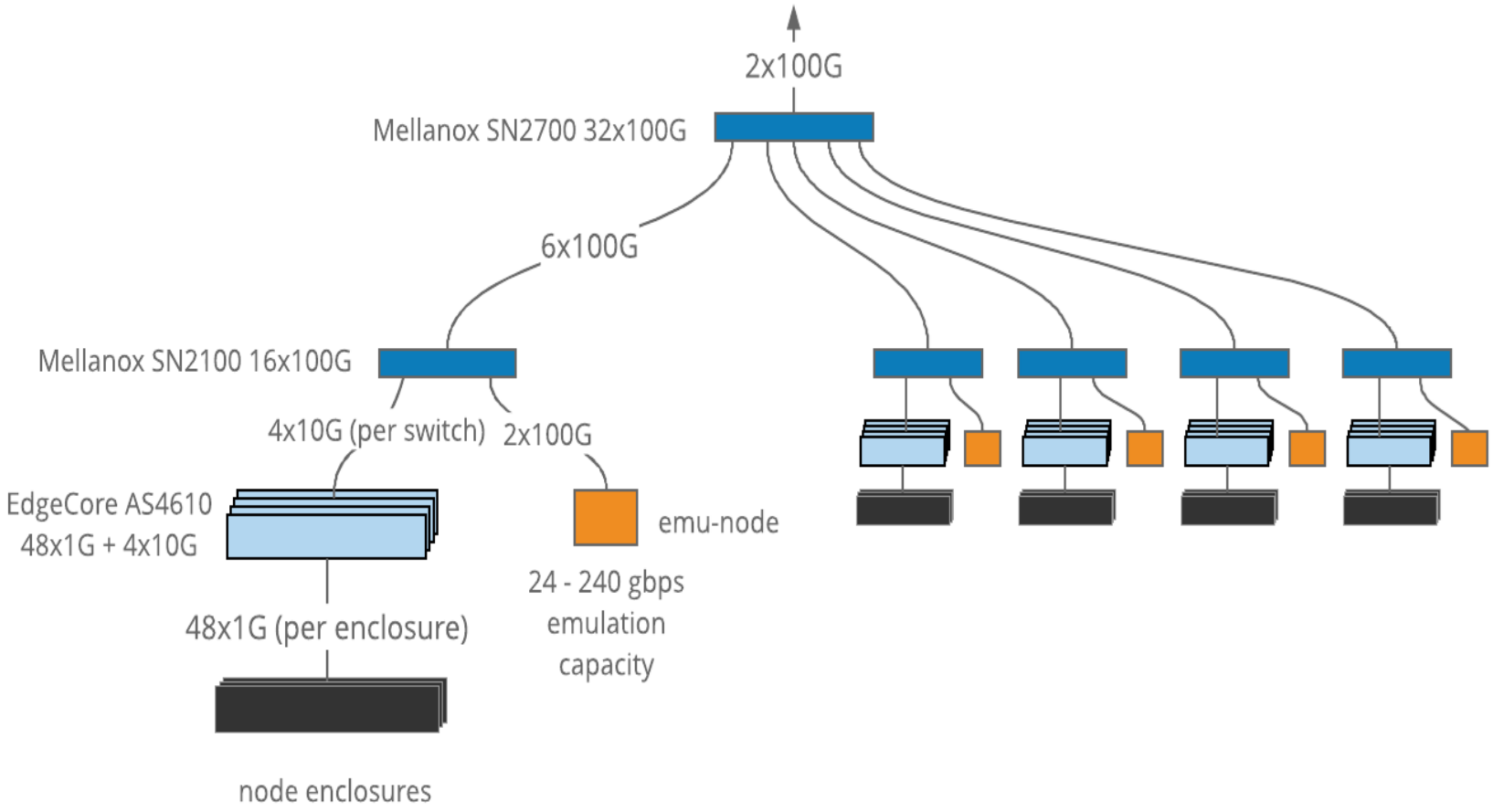
- Offline
  - SN with 100 vertices generated by GT-ITM (transit-stub).
  - 1967 VNRs generated using standard methods in literature (VNRs are not necessarily solvable).
  - Found solutions to 1248 VNRs and ***correctly*** declared 633 as unsolvable.
  - Timed out on 86 VNRs after 5000 high-level node expansions.
  - Total running time for all 1967 VNRs is 100 seconds.
- Online
  - 2000 VNRs over 50000 timesteps. On average, 4 VNRs every 100 timesteps.
  - Average lifetime of a VNR is 500 timesteps.

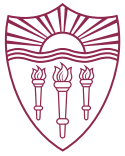






Configuring Live Devices



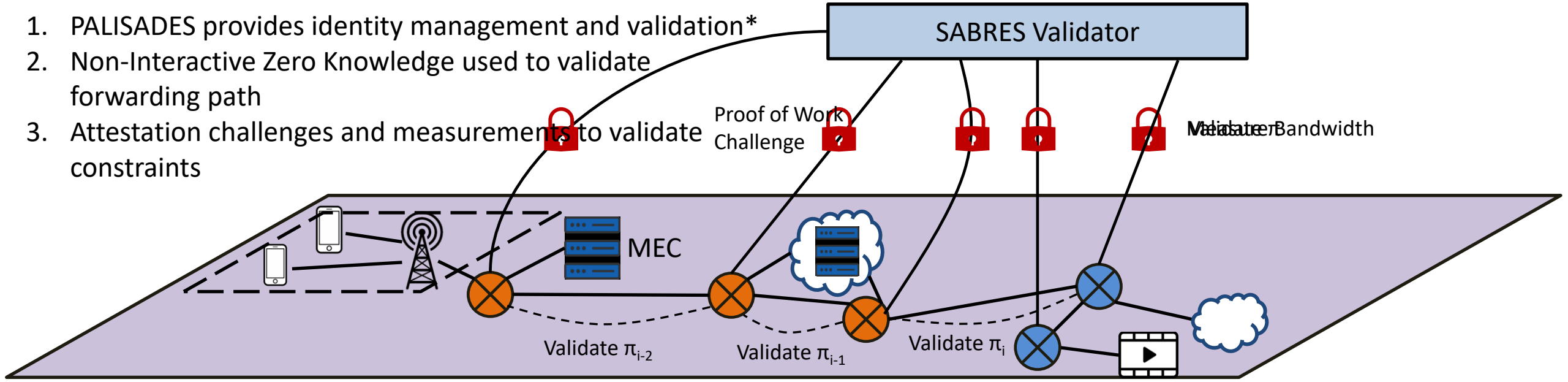


SLICE VALIDATION

Validating Slice Correctness

- SABRES assumes slices may be invalid, thus validation of slices requires:
  1. The nodes selected for the slice are correct
  2. The forwarding paths are correct
  3. The slice constraints are met

1. PALISADES provides identity management and validation*
2. Non-Interactive Zero Knowledge used to validate forwarding path
3. Attestation challenges and measurements to validate constraints

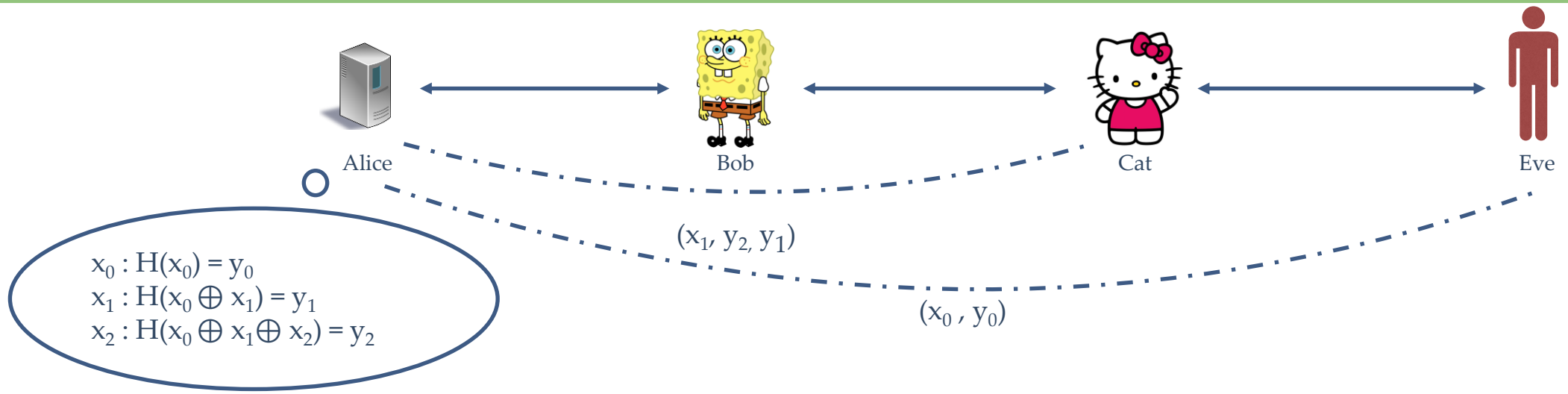


Forwarding Path Validation using Zero Knowledge

At the start of n-hop path, first node samples n-many independent strings ($x_1,...,x_n$) and compute each y_i as the result of applying hash H to all x_j; $j \geq i$ in an XOR combiner

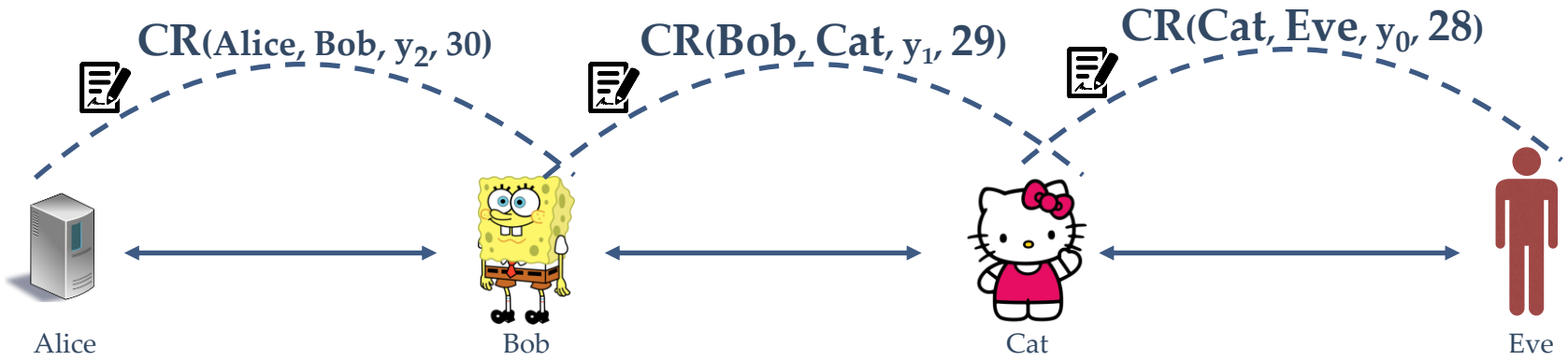
First node provides the last node with (x_0, y_0) and $(i)^{th}$ intermediary with (x_i, y_{i+1}, y_i)

Intermediary nodes also receive ZK proofs to guarantee y_i is well-formed *without revealing all of* x_i



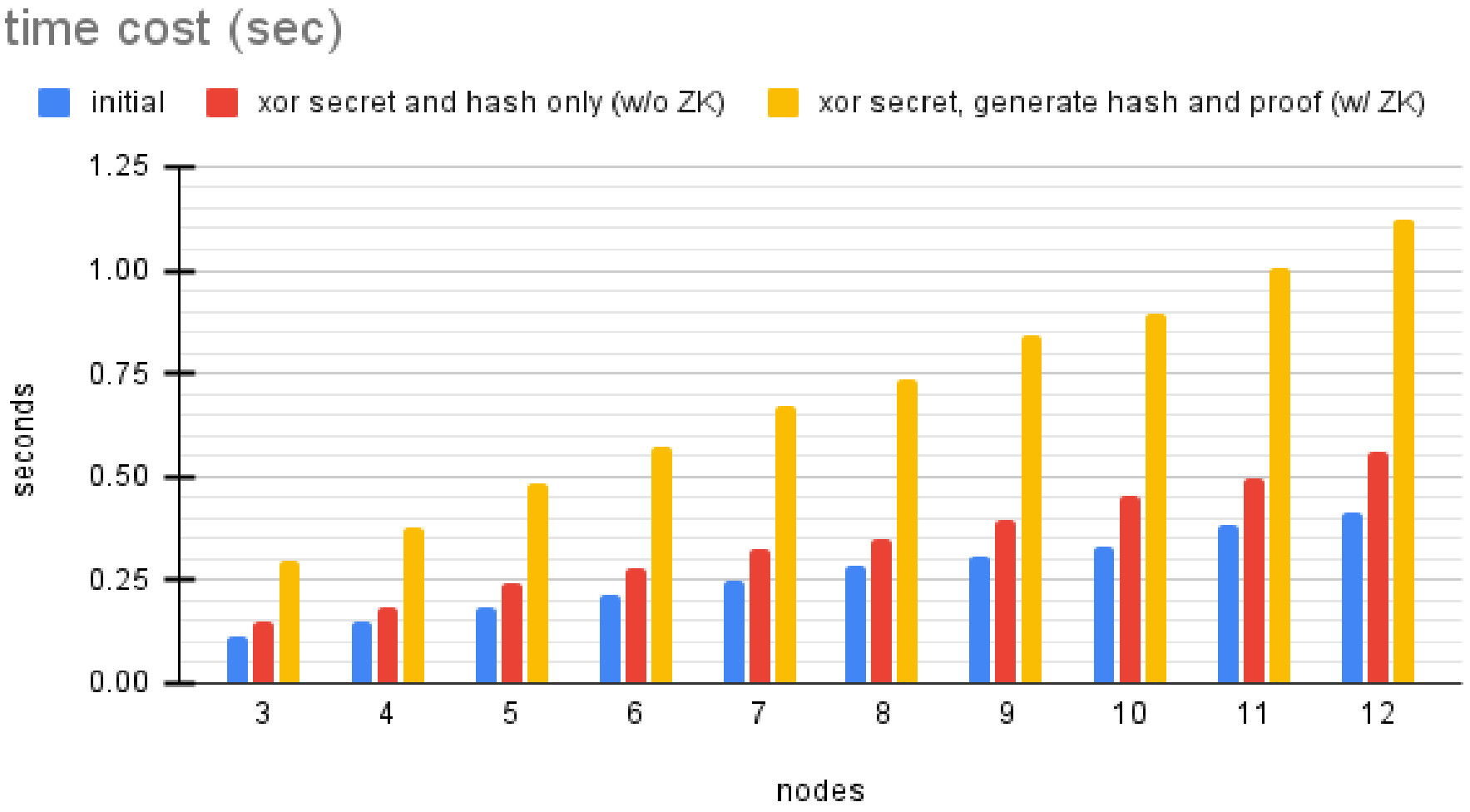
Forwarding Path Validation using Zero Knowledge

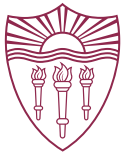
Each node validates that their predecessor and successors contain the correct y. The entire path can be validated from destination to source by revealing the string (hash pre-image) needed for the (i+1)-th node to validate the path with the i-th node



Strategy enables the SO to detect invalid forwarding and take action against the deviant node(s).

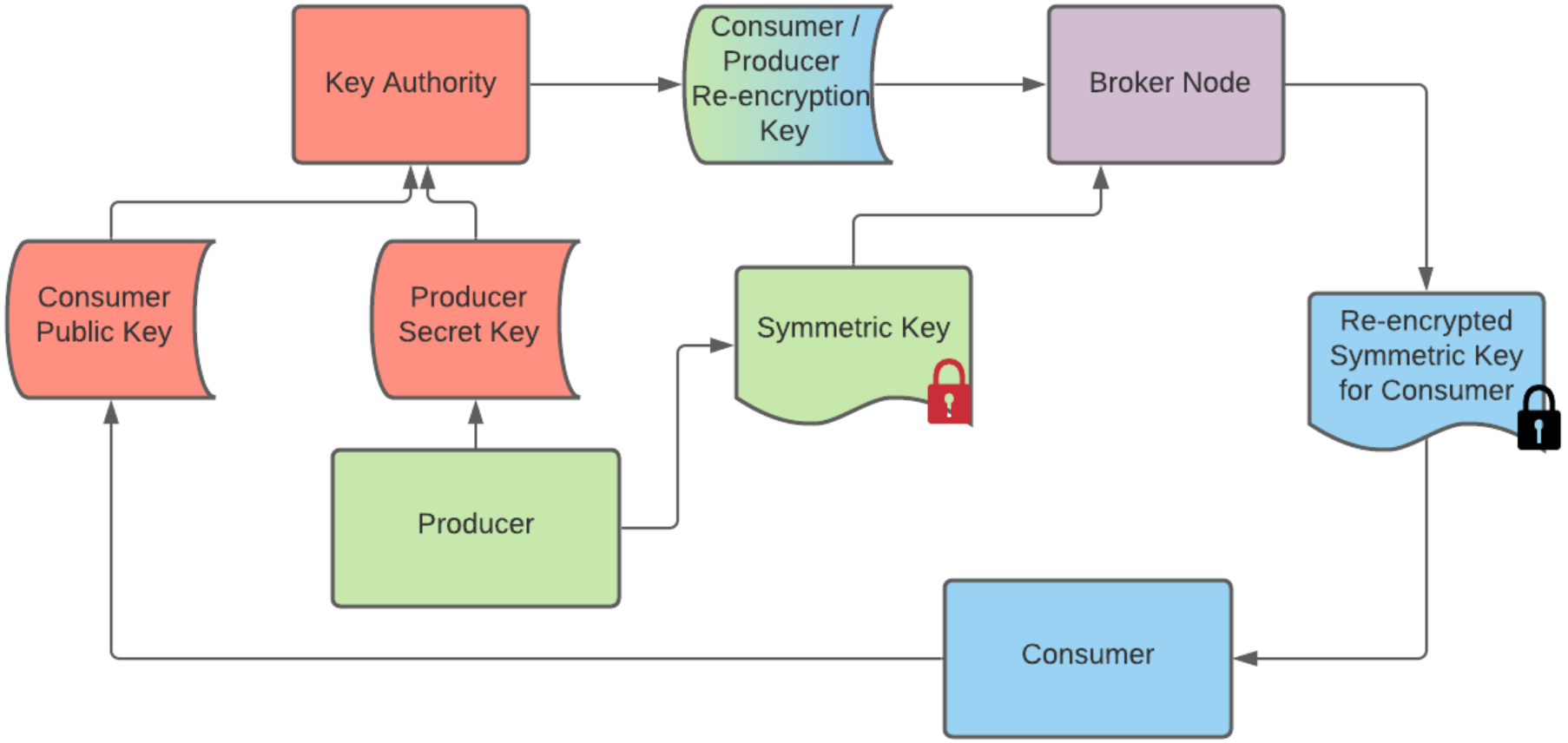
Microbenchmark





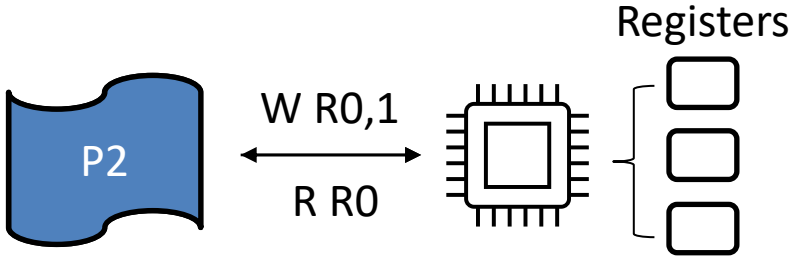
ENHANCED SECURITY CAPABILITIES

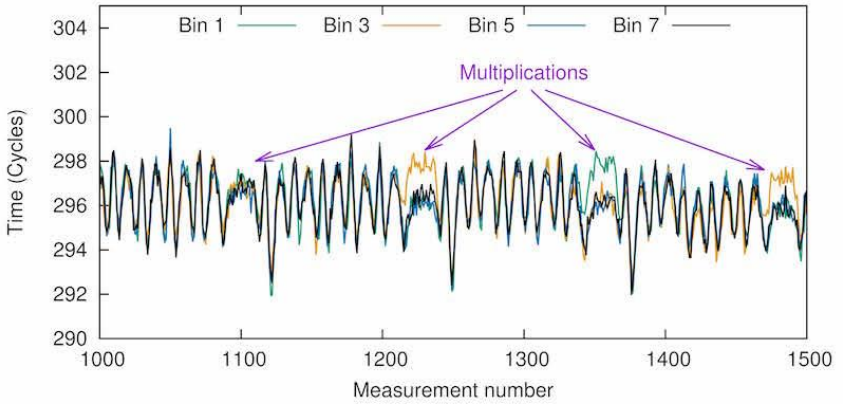
Post Quantum Key Distribution



Preventing Side-channel Information Leakage

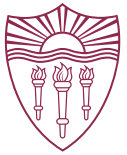
- Co-resident slices or processes may be able to extract information through side-channels





- Mitigations exist but are costly if used constantly
  - Serialization-barriers used for Spectre attacks result in a ~70% performance penalty
  - Other mechanisms such as cache clearing and superfluous no-ops results in massive penalties
- We make mitigations practical by using the defense during critical regions
  - Critical regions are marked within source, imparting mitigation during compilation
  - Mitigation may include expensive software mitigations or hardware mitigations if available

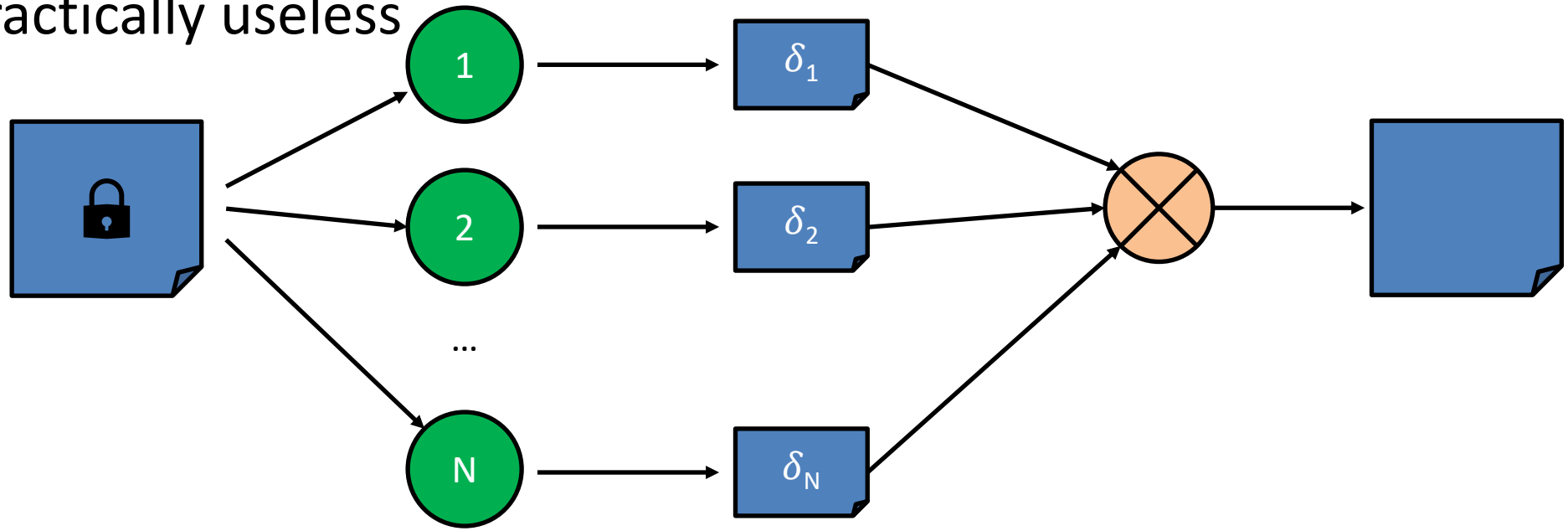
Side-channel Mitigations are not Enough



- Mitigations are necessary but insufficient
  - Only defend against extant, known attacks (or related attacks)
  - Does not protect information leakage through remote observation
  - Does not provide protection against the Infrastructure Owner (or subverted hardware)
- Reduce utility of leaked information
  - Obfuscation and data distribution requiring a more powerful adversary
  - E.g., disrupted traffic patterns reduce the ability of an adversary to discern the application(s) using a slice
  - Multiple capabilities, enabling different cost/benefit tradeoffs

Reduced Utility of Leaked Keys

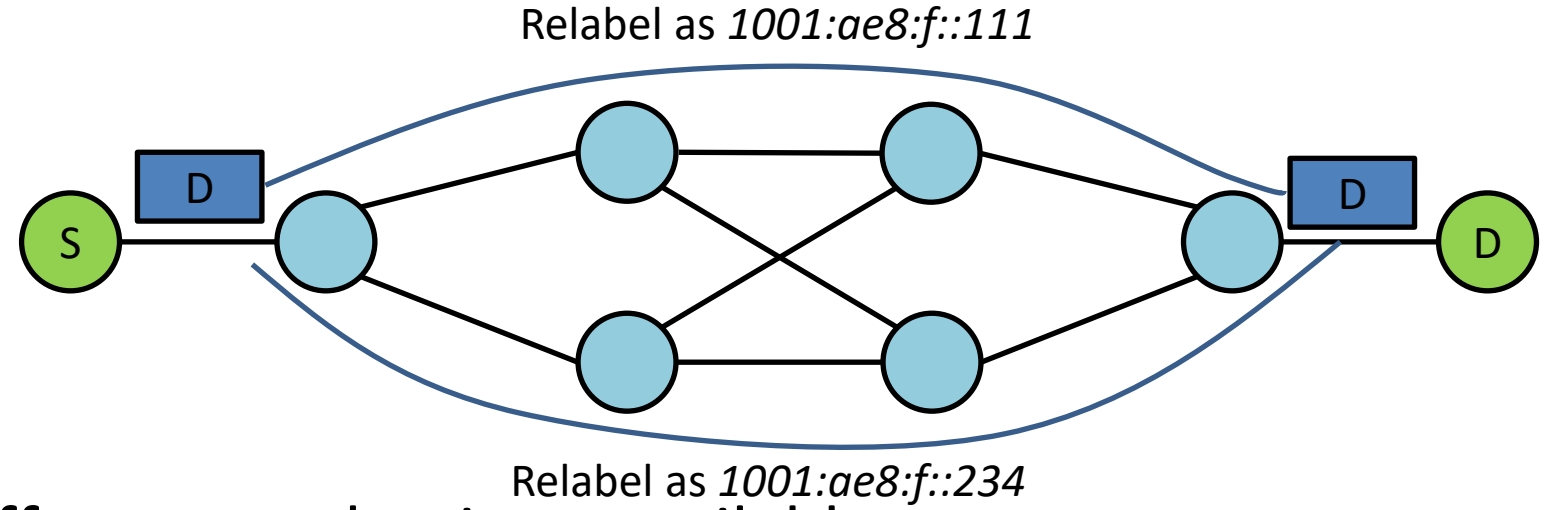
- PALISADES-based threshold and multi-key encryption capabilities makes leaked keys practically useless



- Adversary must extract or subvert all N keys to decrypt content
- Threshold, multi-key, and PRE can conduct encrypted computation

Obscuring Flow Metadata

- Flow metadata (e.g. sources, destinations) potentially leak information
- Metadata can be obscured through alternate labels

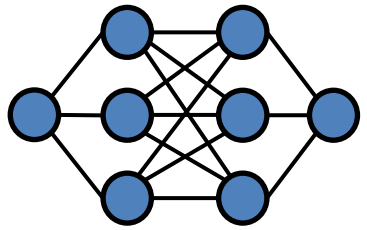


- Multiple different mechanisms available
  - Hide with in IPv6 /64 space
  - Alternatively, IPv4 port space provides some obscurity
  - Pre-building and selecting SABRES MPLS-based tunnels

Flow Disruption via Network Coding

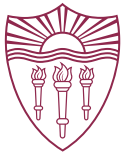
- Split K input packets across N output packets using Linear Encoding

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{1,n+1} \\ a_{2,n+1} \\ \dots \\ a_{n,n+1} \end{bmatrix}$$



  - Merge and split at multiple sites across the network
  - Traditionally used to provide error resilience
- Naturally obscures flow information
  - Distributes flows across network, obscuring which packets correspond to which flow
  - Flow timings and packet sizes completely perturbed
- Leverage linear network codes to quickly distribute flows across paths
  - Linear network codes are highly efficient to calculate
  - Naturally provides multicast capabilities

Questions



- Feel free to reach me at kline@isi.edu