# Verifiable Service Mesh

Ramki Krishnan, Ned Smith: Intel

Diego Lopez: Telefonica

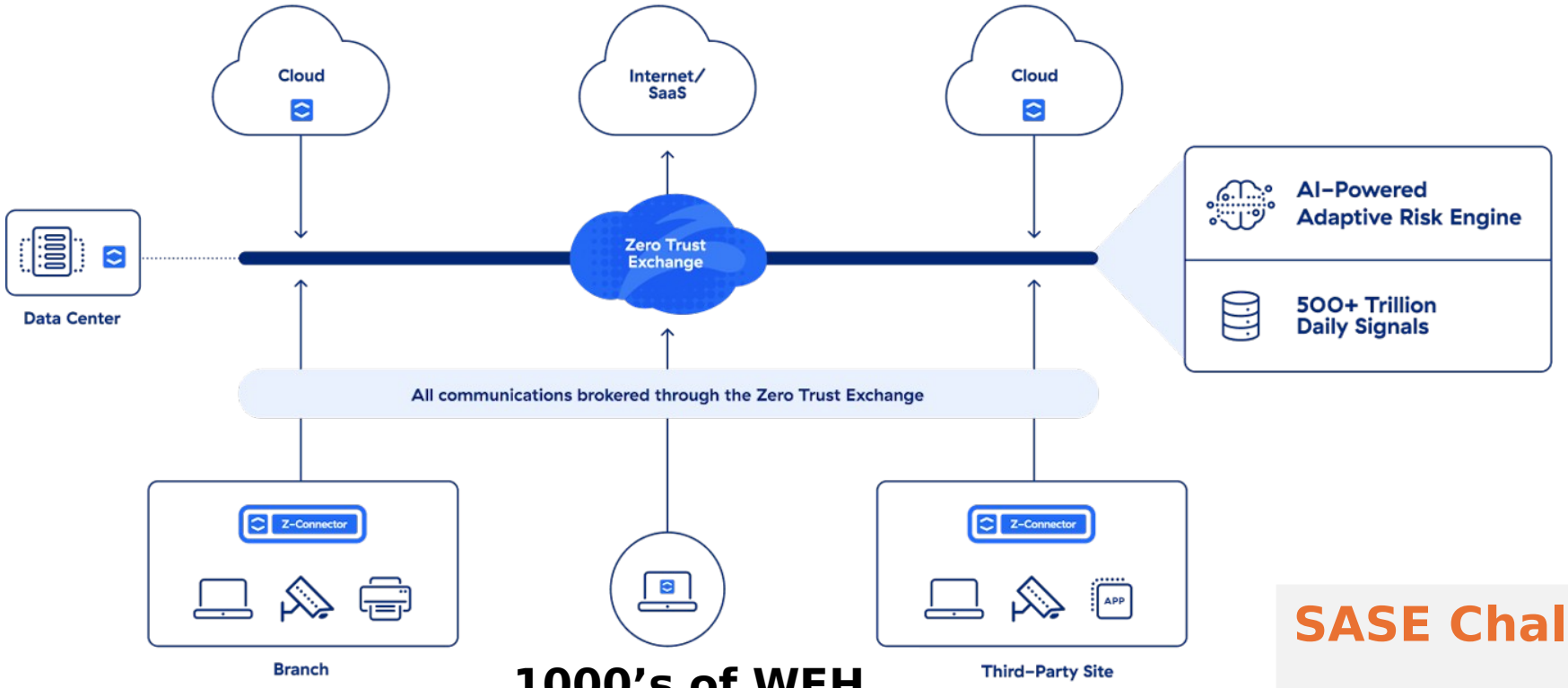
Srini Addepalli: Aryaka

# Introduction

- Secure Access Services Edge (SASE) is a use case for attestation
- RATS Architecture anticipates Verifier role can be divided across multiple nodes
- Consistency of trust implies:
  - Verifier inputs can be grouped
  - Timing of Verifier inputs / outputs can be coordinated across multiple domains
  - Securely connected nodes agree on the trust state of its peer

# Secure Access Services Edge (SASE) introduction

Connects users, sites, and clouds without routed overlays or VPNs




**1000's of WFH employees / affiliates**

Adapted from reference: https://www.zscaler.com/products-and-solutions/secure-access-service-edge-sase

**AI-Powered Adaptive Risk Engine**

**500+ Trillion Daily Signals**

**Security Service Edge (SSE)**

Cyberthreat Protection
(SWG, FWaaS, DNS Security, Sandbox, Browser Isolation)

Data Protection
(CASB, DLP)

Zero Trust User-App Access (ZTNA)

Digital Experience Management

**Zero Trust Networking**

Zero Trust SD-WAN

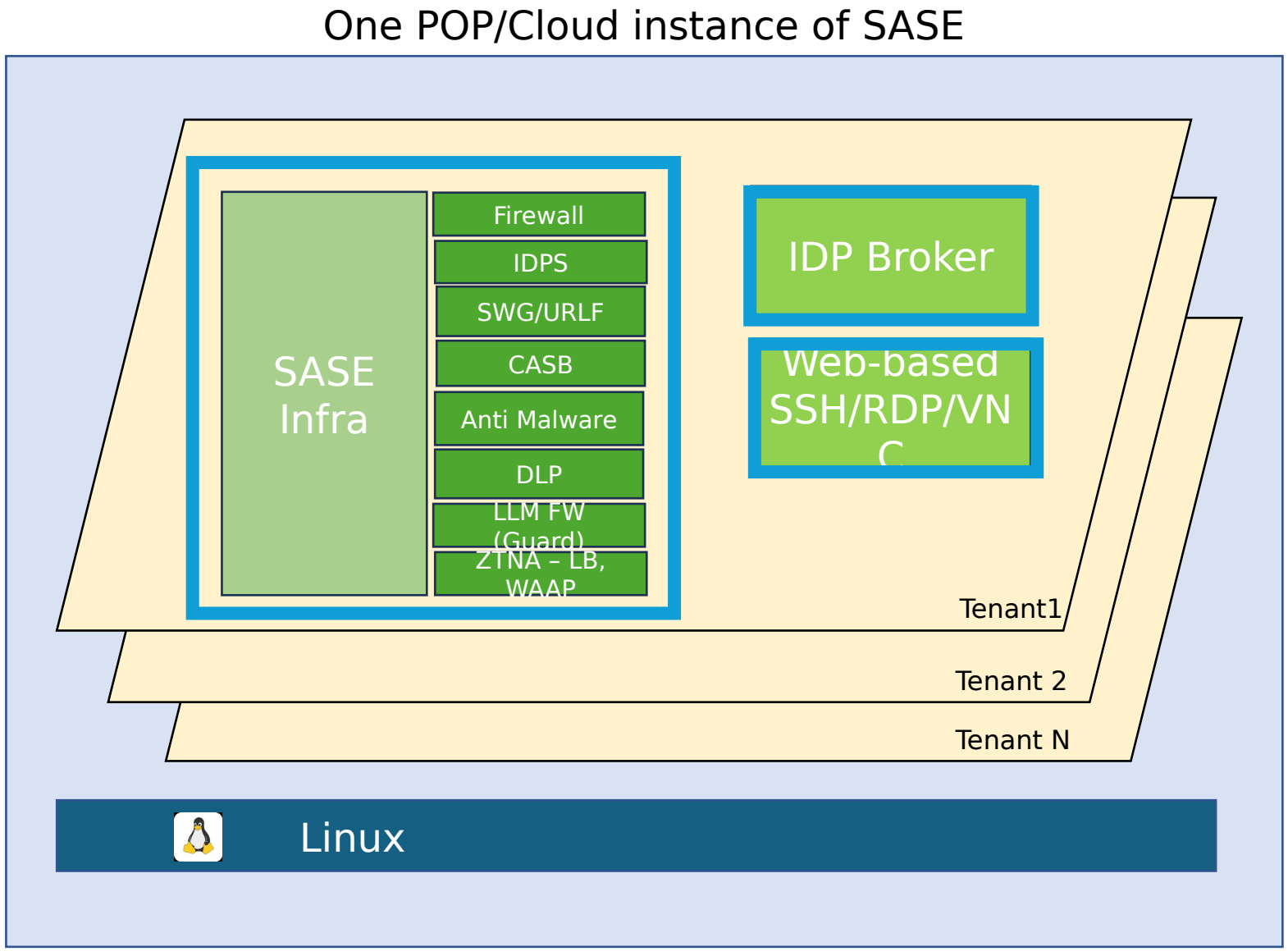
Workload Communications

## SASE Challenges

Possible attack surface increase due to

- Diverse infrastructure
- Diverse geographic locations with different regulations
- Terminate TLS connections - secret material (passwords etc.) in clear in

# Multi-Tenant SASE Architecture and additional Challenges



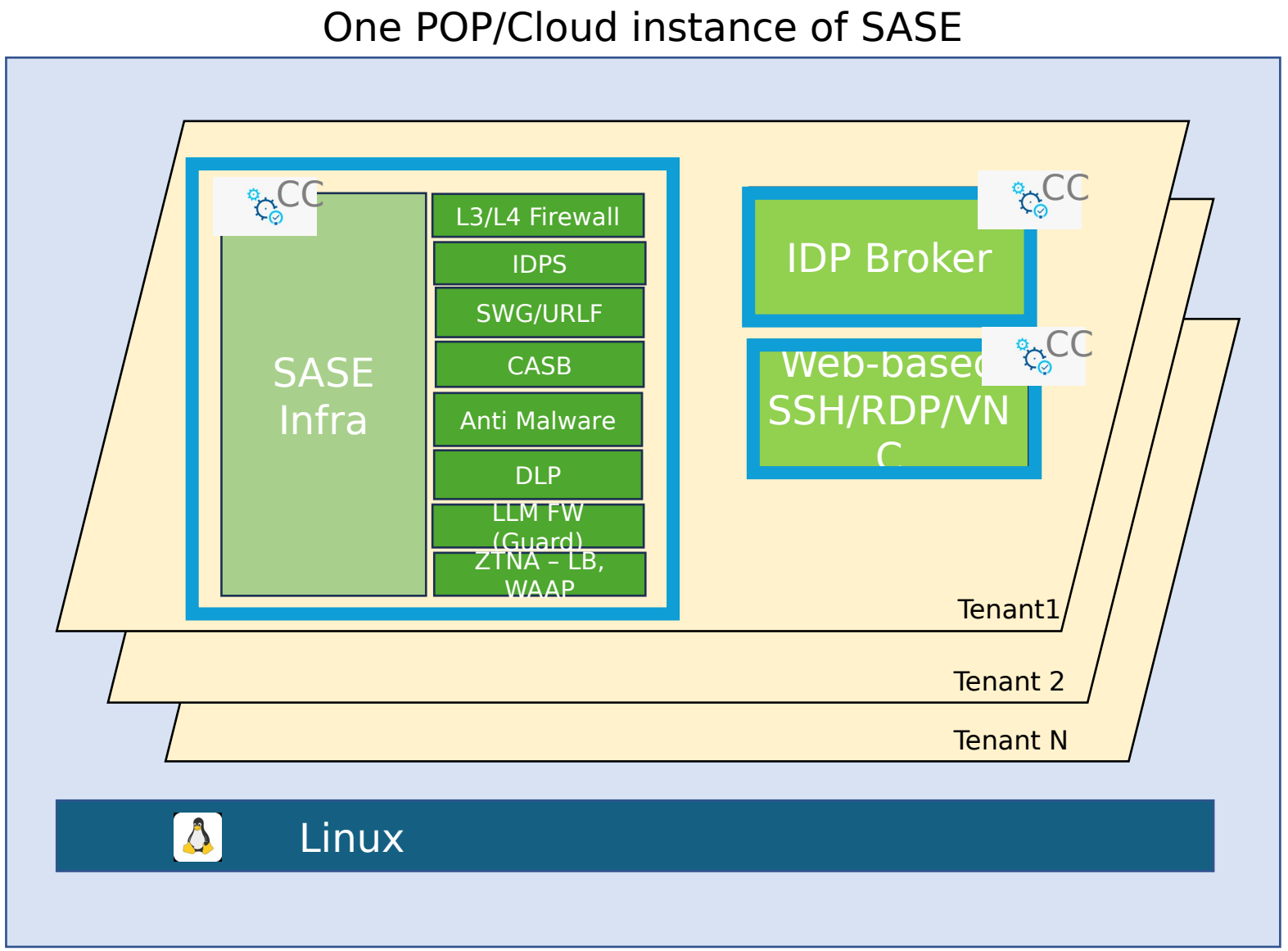

SASE/SSE does good job of security isolation & performance isolation among tenants

**The challenges are**

- Exploitation of vulnerabilities in OS (Linux etc.) to get root access and then getting hold of memory contents of SASE/SSE.

- Malicious insider who has access to the compute systems with higher privileges can scrape memory.

# Addressing SASE Challenges with Confidential Computing

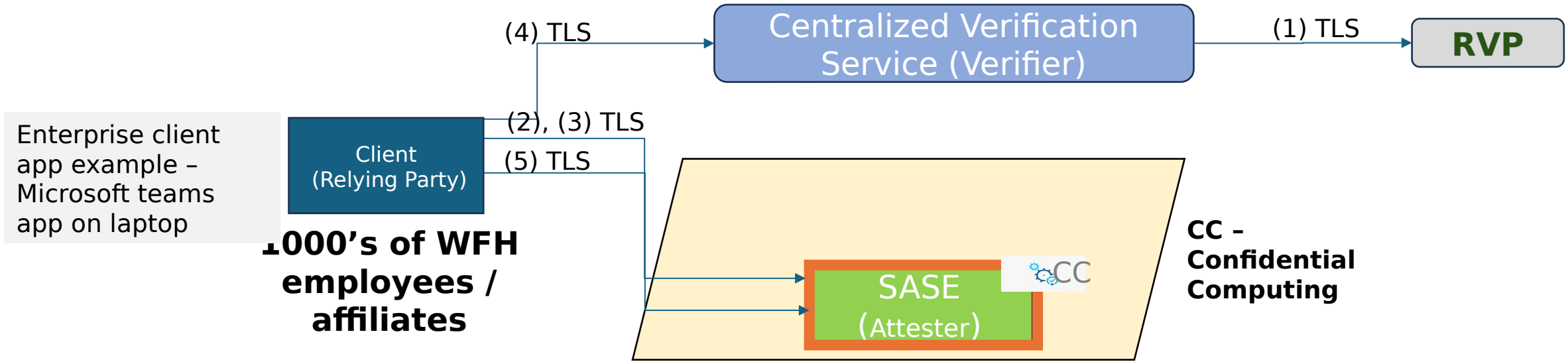



Each SASE process is protected **under Confidential Computing (CC)**

- Any root access exploiting vulnerabilities, malicious administrators can’t see the tenant specific containers’ memory.

Peace of mind for SASE service providers and confidence for tenants.

# CC Attestation solution (background check example) for SASE



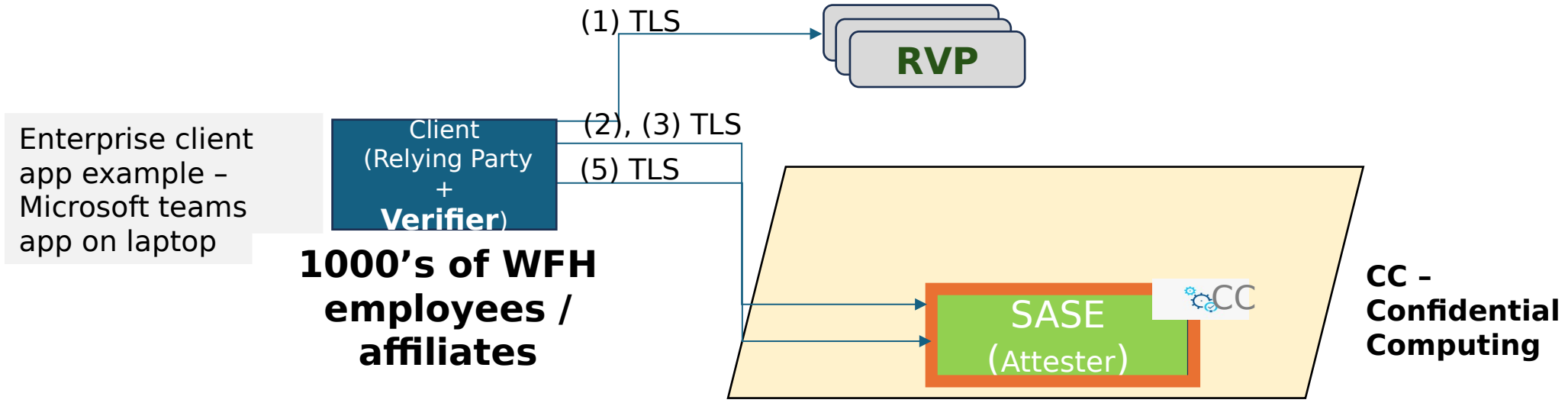

1. Centralized verifier periodically (e.g., once a day) opens new TLS connection to the RVP and fetches new reference values
2. Client opens new TLS connection to SASE
3. Client get SASE CC workload evidence
4. Client opens new TLS connection to centralized verifier, supply SASE CC workload evidence & get attestation token
5. Client verifies attestation token and opens new TLS connection to SASE before accessing backend

**The challenge**

- **Centralized verification can be a bottleneck for 1000's of WFH employees who are simultaneously using the verification service.**

Background check model reference: Section 5.2 in https://datatracker.ietf.org/doc/rfc9334/

# Approach – distribute the verification function to the client for scalability




1. Client verifier periodically (e.g., once a day) opens new TLS connection to the RVP and fetches new reference values.
2. Client opens new TLS connection to SASE
3. Client get SASE CC workload evidence
4. Client connects to local verifier, supply SASE CC workload evidence & get attestation token
5. Client verifies attestation token and opens new TLS connection to SASE before accessing backend

**Benefit**

- **Verification is distributed to 1000's of clients addressing the centralized verifier scalability bottleneck.**
  - **Note: Client verifier needs to sync with RVP(s) periodically (e.g., once a day?)**

# Key architectural aspects to address with distributing the verifier to the client

1. Running client verifier in CC
    1. Centralized Verifier runs in CC
    2. Verifier security promise doesn’t change due to distribution

2. Provisioning 1000’s of client CC verifiers in a scalable way.
    1. Orchestrate Verifier + RP policy
    2. Verifier community might not be a homogenous organizational domain

3. Address eventual RVP consistency model in 1000’s of client CC verifiers.
    1. RVPs continuously update RVs
    2. Timing of Verifier inputs to ensure consistent Attestation results (ARs)

4. Secure channel (e.g., TLS) should be bound to SASE host and Client platforms via attestation
    1. SASE TLS endpoint load balancers don’t tolerate long session establishment latency
    2. Freshness of SASE Evidence is determined by SASE update management

# RATS Recommendations?

Orchestrate Verifier + RP policy

- Unclear if/how draft-ietf-rats-posture-assessment MPS groups apply to bundling of RV / Endorsement objects
- Unclear if/how WIMSE domain (group ID) works for nodes that are not in the domain that manages MPS bundling

Timing of Verifier inputs to ensure consistent ARs

- Unclear if/how draft-ietf-rats-epoch-markers BELL can be coordinated across multiple independent RVPs
- Unclear what constitutes ”consistency” of trust given a constantly changing Internet.

TLS endpoint binding

- Unclear if/how AR is bound to TLS endpoints

# Appendix

# Verifiable Service Mesh – Applies to broad set of use cases

- Intra-enterprise microservice interconnect (app to app within enterprise)
- Enterprise WFH employee connecting to Enterprise/Cloud apps (client-to-app)
  - Client app to Enterprise app via Ingress SASE
    - TLS connection terminating in SASE
  - Ingress SASE <-> Cloud service app
    - TLS connection
- Enterprise  X  App connecting to Enterprise  Y App (cross-domain app-to-app)
  - Federated AI example - Communication between collaborator, aggregator using TLS
- Enterprise branch DC to main DC secure connectivity (VPN connectivity within enterprise)
  - Enterprise branch router <-> Enterprise main router
    - IPSEC tunnel (attested IKEV2)

*Assumptions*

- *SASE, Cloud service could be from different vendors.*
- *Routers could be from different vendors.*