ScaleGuard: Large-Scale HIP-VPLS with Customer-Side Data Plane Encryption

Introduction

As network infrastructures continue to expand, ensuring secure and scalable communication between distributed sites has become a critical requirement. Traditional IP-based addressing, which ties device identity to network location, introduces security and management limitations, particularly in large, multi-site environments. The Host Identity Protocol (HIP) addresses this issue by introducing a cryptographic identity layer, decoupling identity from routing and enabling stronger access control and authentication.

To provide seamless communication across geographically distributed offices, many organizations rely on Virtual Private LAN Services (VPLS), which emulate a single broadcast domain over a wide-area network. When combined with HIP, this forms the foundation of Host Identity Protocol-based VPLS (HIPLS), a secure overlay that ensures only authenticated and authorized endpoints can participate in the virtual network.[1]

While HIPLS provides a strong foundation for secure control-plane and data plane communication within the provider network, additional protection is required at the data plane for customer sites, particularly when sensitive traffic traverses untrusted provider networks. This project focuses on enhancing HIPLS by enabling customer-side encryption of data frames before they enter the provider network, ensuring end-to-end confidentiality between trusted sites.

Moreover, another key challenge lies in scaling the existing HIP-based multi-party key exchange (as shown in Figure 1) mechanism to large network sizes (100–200 nodes) while maintaining performance and reliability. This project addresses that challenge by integrating customer-edge encryption with a scalable group keying system, evaluating the system’s ability to protect data traffic while operating across a large number of sites.

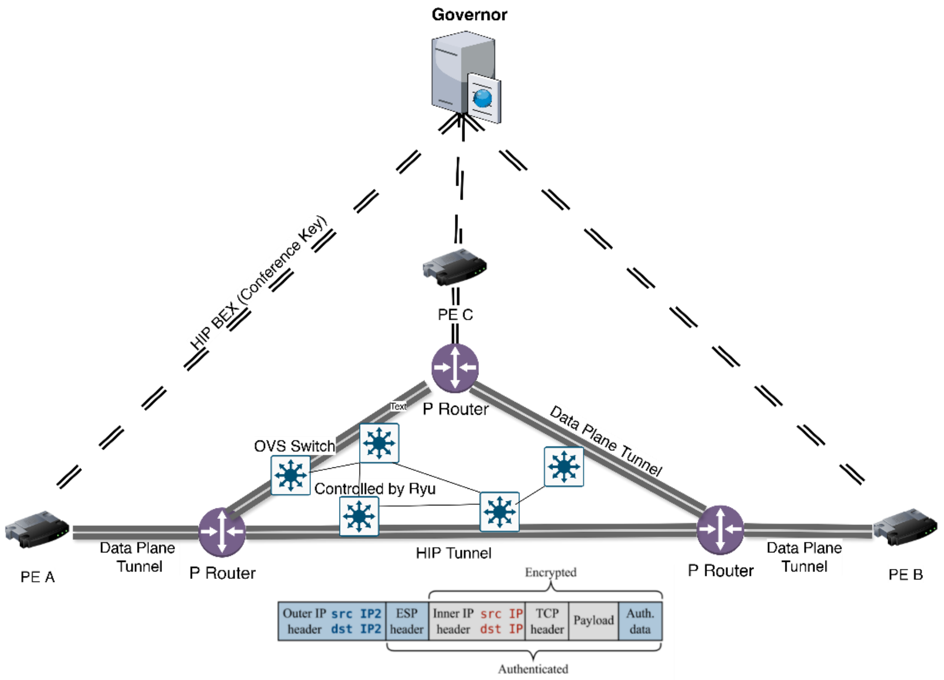
The outcomes of this project will contribute to building secure, zero-trust network overlays that support enterprise-grade scalability, with potential applications in sectors requiring strong data privacy and communication integrity.

[1] For real-world scenarios, visit https://www.tempered.io/

Project Description:

This project aims to implement and evaluate a scalable data plane encryption solution within a HIPLS environment. The primary focus is on extending the current multi-party key exchange protocol to support large-scale deployments (100–200 nodes) while enabling customer-edge encryption of traffic before entering the provider network. Using Mininet as the emulation platform, the project will ensure that only authorized nodes can participate and communicate securely, with encrypted data traversing untrusted infrastructure.

Figure 1: Secure multi-party HIP-based VPLS



Project Objectives:

1. **Scalable Multi-Party Key Exchange in HIPLS:**
   - Extend the existing HIP-based VPLS (HIPLS) framework to support **large-scale deployments** (100–200 nodes) using an efficient group key agreement protocol.
   - Optimize the protocol for scalability, minimizing computational and communication overhead as the number of nodes increases.

2. **Customer-Side Data Plane Encryption:**
    - Implement customer-edge (CE) encryption using the derived session key to encrypt all outbound data before it enters the untrusted HIP-VPLS network.
    - Ensure that only authenticated CE nodes can decrypt and access traffic, enforcing end-to-end confidentiality at the edge.
    - Use the proposed KDF-based key derivation and OTP-based authenticator to securely exchange the session key across authenticated CEs, enabling confidential communication over untrusted channels.
    - Use *Mininet* to simulate network topologies of varying sizes and link conditions to evaluate the behavior of the key exchange and data encryption mechanisms.
    - Measure key performance indicators such as group key setup time, encryption overhead, and throughput impact under large-scale conditions.

Project Deliverables:

1. **Source Code:**
    - Complete source code for the scalable multi-party key exchange and customer-side encryption and decryption modules integrated at CE nodes
    - Scripts to automate testbed setup, key distribution, and data encryption flows in simulated environments.
2. **Technical Documentation:**
    - User manual for deploying, configuring, and testing the system under different network sizes and topologies using Mininet.
    - Developer documentation describing the group key exchange protocol, session key derivation process, and integration with the HIPLS control and data planes.
3. **Simulation Results and Analysis:**
    - Detailed report on the system's performance under scale, including metrics such as key exchange time, encryption overhead, and throughput under load.
    - Evaluation of scalability, encryption impact, and limitations, with discussion on potential improvements and use cases in real-world zero-trust deployments.

Requirements:

- Completing the Computer Networks and Distributed Systems course (TDTS04-TDTS06-TDTS11 or any related course) is essential.
- **Advanced** Python programming skills (base repo and mininet were written purely in Python). Experience with modular programming and navigating medium-sized codebases **(~15,000 lines)** is important. Students are not expected to write from scratch, but must be able to read, understand, and extend existing components.
- Solid understanding of network security fundamentals, including cryptographic algorithms and protocols such as Diffie-Hellman (DH), RSA, AES, HMAC, and IPsec. Students will work directly with cryptographic libraries.
- Competence in socket programming and working with low-level network protocols for implementing and testing communication between hosts.

Readings:

- Host Identity Protocol (HIP): Towards the Secure Mobile Internet: https://www.wiley.com/en-us/Host+Identity+Protocol+(HIP)%3A+Towards+the+Secure+Mobile+Internet-p-9780470772904
- Primer on Host Identity Protocol: https://www.ida.liu.se/~TDDE21/info/primer-host-identity-protocol-whitepaper.pdf
- https://github.com/strangebit-io/hip-vpls
- https://asecuritysite.com/kdf/
- https://asecuritysite.com/kdf/HKDF