MONDRIAN: Comprehensive Inter-domain Network Zoning Architecture

Jonghoon Kwon
ETH Zürich
jong.kwon@inf.ethz.ch

Claude Hähni
ETH Zürich
claude.haehni@inf.ethz.ch

Patrick Bamert
Zürcher Kantonalbank
patrick.bamert@zkb.ch

Adrian Perrig
ETH Zürich
adrian.perrig@inf.ethz.ch

***Abstract*—A central element of designing IT security infrastructures is the logical segmentation of information assets into network zones sharing the same security requirements and policies. As more business ecosystems are migrated to the cloud, additional demands for cybersecurity emerge and make the network-zone operation and management for large corporate networks challenging. In this paper, we introduce the new concept of an inter-domain transit zone that securely bridges physically and logically non-adjacent zones in large-scale information systems, simplifying complex network-zone structures. With inter-zone translation points, we also ensure communication integrity and confidentiality while providing lightweight security-policy enforcement. A logically centralized network coordinator enables scalable and flexible network management. Our implementation demonstrates that the new architecture merely introduces a few microseconds of additional processing delay in transit.**

I. INTRODUCTION

Network zoning has long been an essential part of the Internet security infrastructure, which logically partitions network and information assets into disjoint segments that share the same security requirements and policies, and functional similarities. Zones define the network boundaries and their defense requirements by explicitly stating the entities populating the zones, the entry points into the zone, and how traffic is monitored and filtered at these entry points. Informally, these zones are realized by a virtualized separation at layer 2 (e.g., IEEE 802.1q [28]) with firewalls at higher levels governing data transfers between zones [40].

Each zone is associated with a security level. By default, a host within a zone with a low security level cannot access a higher security zone. To realize access control based on the security level, firewalls are considered to be the most viable technology. However, operating firewalls in large enterprises is often challenging for network operators and security architects. The access control for network zones might be dynamic, which requires complex management schemes to accommodate a myriad of policies. While there are advanced technologies such as virtual firewalls [3], [12], distributed security enforcement [38], [69], and Unified Threat Management (UTM) [53], newly designed to enforce access control polices in extremely dynamic networks, network zone management and modeling still remains cumbersome [21], [54].

Network and Distributed Systems Security (NDSS) Symposium 2021
21-25 February 2021, Virtual
ISBN 1-891562-66-5
https://dx.doi.org/10.14722/ndss.2021.24378
www.ndss-symposium.org

Bridging geographically distant network zones is very challenging today. In general, network zones are created not only for security purposes but also because of geographical, operational, or organizational factors. Large enterprises with geographically distributed branch networks, and possibly collaborative partners' networks need to be interconnected. Given that distant network zones exchange information over an untrusted network (e.g., the Internet), there is a risk that the communication exposes security-sensitive information during transit. To mitigate such threats, administrators leverage additional security mechanisms (e.g., IPsec [32] and TLS-VPN [55]) which ensure confidentiality and integrity of the transmission over the untrusted network by encrypting and authenticating the data with shared cryptographic keys. Nonetheless, these technologies bring forth new challenges such as management scalability [19] and compatibility issues with other security solutions [34]—universal agreement with business partners on building collaborative security infrastructure is often problematic.

MONDRIAN is a new network zoning architecture that secures inter-zone communication—which operates on layer 3, supporting heterogeneous layer 2 architectures—while ensuring scalable cryptographic-key management and flexible security policy enforcement. MONDRIAN flattens the current hierarchically-complex network zone topology into a collection of horizontal zones connected to a unified security gateway, called Zone Translation Point (TP), thus simplifying large enterprise networks. By interconnecting zones through TPs, complex zone restructuring operations become easier with respect to new zone initializations or zone migrations. The TP ensures source authentication, zone transition authorization, and illegitimate access filtering by acting as a secure ingress/egress point for network zones. A logically centralized control unit provides management scalability on zone classification and policy enforcement, and mediates cryptographic key establishment. Since poor security practices, complicated controls, and rushed updates cause security breaches [37], the management scalability and flexibility offered by MONDRIAN provide strong, transparent, and efficient controls, minimizing human error and enhancing the security of enterprise networks.

A secure zone transition is performed in three steps: i) the TP acquires access policies for each network zone from its controller, ii) the TP issues a cryptographically protected authorization token if a given zone transition request is permitted, and iii) the network forwards only packets with a valid token. By leveraging the notion of secure tunneling between two endpoints (i.e., a pair of local and remote TPs), confidentiality

and integrity of the zone transition packets are ensured, while keeping the overhead of the authentication process small. For scalable key management, we employ a key establishment system that enables dynamic key derivation and ensures perfect forward secrecy.

We provide an implementation of MONDRIAN that ensures secure zone transition for both intra- and inter-domain communication at line rate, while requiring no network-stack changes from end hosts. We extensively evaluate this implementation to demonstrate the practical viability of MONDRIAN. The results show that the TP introduces negligible processing delay; less than $500\,\mathrm{ns}$ of additional delay for intra-domain zone transition and approximately $2.5 \sim 3\mu\mathrm{s}$ for inter-domain zone transition. We further provide in-depth analyses for security and practical considerations.

The main contributions of this paper are the following:

- We introduce MONDRIAN, a new security architecture that enables secure, flexible and viable network zoning and inter-zone communication for large enterprise networks.
- We introduce an inter-domain transit zone that simplifies the current hierarchical zone structure, enabling flexible network zone management.
- We implement MONDRIAN as an opensource project.

II. NETWORK ZONING

Using a case study we explore how network zoning is realized in modern enterprise networks, and later we derive the main challenges we confront.

A. Case Study

Most enterprise networks have embraced the notion of layered security classification, that can be broadly split into intranet, extranet, and opennet [54]. The opennet is the least trusted network (e.g., the Internet) which is an inhospitable region where live threats exist, whereas the intranet is the most trusted network hosting business-critical systems and sensitive information. Since the intranet has rigorous access control mechanisms to protect information assets from exposure to the opennet, enterprises are forced to operate another security layer (extranet, also known as demilitarized zone or DMZ) in between, which exposes the publicly accessible services to the opennet, while reducing the attack surface.

Over time, the layered network structure has become more sophisticated [45] due to extreme changes in network environments—diverse demands from customers, partners and employees accessing enterprise networks with a variety of devices. As a result, many enterprise networks comprise a large number of zones defined by operational, organizational, and most importantly security factors. Figure 1 depicts a real-world use case for network zones running on inter-domain level with multiple involved autonomous systems (ASes). Zone transition can be categorized into three main types.

Intra-domain Zone Transition. Within a local network, multiple devices such as servers, databases, and hosts are connected through network switches. These devices are assigned with a unique IP address that belongs to a logically

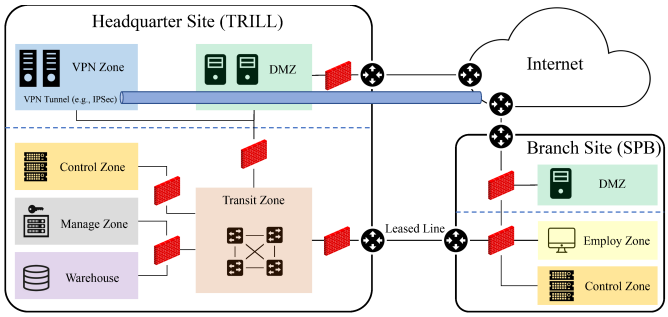

Fig. 1: Network zoning use case for large enterprises. Network zones are realized with heavy use of security middleboxes (e.g., Firewalls).

isolated network zone. These zones commonly consist of multiple subnets, often realized with a layer 2 virtualization technology (e.g., VLAN). Each zone is protected by a set of security middleboxes, e.g., firewalls, intrusion-prevention systems (IPS), and intrusion-detection systems (IDS), which enforce predefined security policies for all traffic passing through.

To maintain the zone-based trust model, an access permission to one zone is not considered to be valid for other zones. That is, an entity must obtain access permissions from all zones on the path when accessing a non-adjacent zone. This trust model however often complicates policy management and enforcement, especially for large enterprise networks. To resolve this complication, the current practice introduces the notion of a dedicated zone transition point, called Transit Zone.

A Transit Zone acts like a patch panel allowing zones to be interconnected without the need of a dedicated link between each pair of zones. The Transit Zone sits in the middle of multiple zones and mediates access between zones that need to communicate with each other. It is commonly comprised of only forwarding devices (e.g., switches), interconnecting the attached zones via various ingress/egress points on which security middleboxes enforce the security policies. In a nutshell, the Transit Zone reduces the depth of zone hierarchies and thus simplifies the network zone design and management.

Inter-domain Zone Transition. To ensure that geographically distributed zones can securely communicate with each other, enterprises employ various networking technologies. The most common choice is connecting two remote sites with a physical leased line, (e.g., Layer 2 circuit). Enterprises can lease these lines from Internet service providers and make use of them to bridge local networks. However, purchasing leased lines is costly and might raise trust issues towards the service provider.

An alternative is a virtual private network (VPN). A VPN uses encryption and authentication to create a virtual tunnel between two local networks, thwarting information leakage during transmission over the public Internet. While a VPN can achieve data confidentiality, typically yet another layer of overlay protocols is required to achieve virtual separation of zones. The use of such overlay protocols, however, has the disadvantage that all interconnected sites need to deploy the same protocol since such protocols generally do not offer interoperability.

Traffic from the Internet. Traffic not originating from cooperative (trusted) networks can be classified into the following three types: i) public traffic, ii) authorized traffic, and iii) malicious traffic. The first case covers customers who access the enterprise's public services, e.g., Web servers. This traffic in general ends up at the demilitarized zone (DMZ) hosting only public services that require exposure to the Internet. The second case refers to the traffic coming from temporarily authorized devices. For example, a legitimate employee outside the enterprise's premises—working from home with a personal device—may get a temporal permit to access restricted zones via a VPN. The last category comprises attack packets which are to be filtered by the security middleboxes in the frontline of defense.

B. Challenges

Secure Zone Transition. Transmitting security-sensitive data between zones in different physical locations (e.g., data center to branch site) over the public Internet poses a challenge. Security level information is lost in transit, requiring that the data is re-authenticated and filtered again on the receiving site even though source and destination could be part of the same logical zone. Today's overlay protocols are often used to overcome the restriction of losing security level information in transit. This however introduces new challenges: difficulties in deployment per zone, computational overhead, and poor management scalability.

Interoperability. To support seamless interconnection between security zones in different networks, we consider interoperability as another challenge. Even if security-level information persists in transit, different zones might not be built on the same internal protocols. For example, large enterprise networks often lease a physical network infrastructure from Internet service providers or cloud service providers, in which a different layer-2 protocol is running (e.g., Shortest Path Bridging (SPB) [27] vs. Trill [48]). The use of different protocols eventually results in the incompatibility of zone translation and makes it difficult for end systems in different zones to communicate with each other seamlessly.

Management Scalability. In current local network zoning architectures, administration is being considered a tedious, time-consuming, and labor-intensive task. For example, simply adding a new zone might require existing policies to be thoroughly reviewed, updated, and re-distributed to the local network entities. The management complexity dramatically increases in a wide-area network (WAN) environment.

III. Overview

This section provides an overview of MONDRIAN. We first elaborate on the fundamental goals of this research, along with requirements, and design choices (§III-A). We then sketch MONDRIAN including a brief introduction of each component and associated workflow (§III-B). Finally, we describe our threat model (§III-C) and state our assumptions (§III-D).

A. Design Principles

Goal. The fundamental goal of this work is to build an architecture that weds local network zoning with inter-domain routing to achieve lightweight interoperability, secure zone transition, and incremental deployability—thereby enabling secure, scalable, and flexible network zoning on a global scale. That is, an administration domain expresses zone definitions and corresponding zone transition rules, and deploys the policies to distributed network entities. These policies force the network to only forward authorized packets protected by cryptographically secured authenticators, ensuring secure and sustainable zone-to-zone communication.

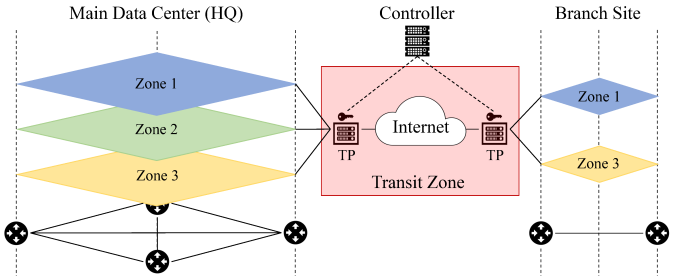

Fig. 2: An overview of the MONDRIAN architecture. The inter-domain transit zone interconnects physically and logically distributed network zones with unified security policy enforcement.

Desired Properties. We consider the following properties to achieve this goal.

- Data confidentiality: through a constructive approach, the zone transition protocol ensures that no information is exposed while being transmitted via the public Internet.
- Management scalability: logically centralized orchestration empowers network administrators to easily migrate network topologies, update policies and mirror abstract network zones into the real network.
- Efficiency: the cryptographic primitives introduce only minor performance overhead in terms of latency, bandwidth, and operational costs.
- Deployability: the MONDRIAN architecture requires minimal changes to the existing network infrastructure in order to achieve compatibility. Furthermore, firewalls and VPN devices at each entry point of every zone can be replaced with one MONDRIAN gateway, saving operational costs for the same level of network security.

B. MONDRIAN Overview

Entities in MONDRIAN. Figure 2 illustrates an overview of MONDRIAN. Different branch sites of an enterprise are interconnected over a WAN (e.g., the Internet). Each site contains multiple, logically separated zones connected to the single Zone Translation Point (TP) at the corresponding site. The TP is a designated gateway for zone transition, operating on layer 3 and interconnecting all zones at a given site of the enterprise network. All the traffic towards either internal network or WAN, therefore, passes through the TP. Note that TPs are the endpoints of our architecture, meaning that minimum changes to the internal network are required, ensuring compatibility with modern enterprise environments.

The main task of TPs is twofold: first, they ensure that traffic adheres to a set of allowed zone transitions. For a packet originating in a zone and destined for another zone, the transition must be explicitly allowed by a policy. Second, TPs enable communication across the WAN without losing previously established security information. To this end, TPs embed a tag with cryptographically secured zone information into packets before they leave the internal network. Furthermore, TPs act as endpoints hiding sensitive information, such as internal addresses and the respective zone binding, from external entities. We also note that, for a case where zones with the same security requirements and functionality are distributed over multiple branch sites (e.g., Zone 1 and 3 in Figure 2), we consider them as the same logical zone (e.g., the same zone identifier). We call this concept zone extension which is not subject to zone authorization.

A logically centralized controller orchestrates the TPs. The controller is managed by a single administrative domain, and thus the operator of the controller manages all the zones behind the TPs. The controller provides its operator an interface in which the outline of zone structure and transition policies can be specified. The explicit network configuration is then distributed to the TPs via a secure control channel to enforce the configuration at the individual premises.

Communication Flow. MONDRIAN enables secure zone transition over WAN as follows:

1) Network administrators first establish an IP-zone map and their transition policies, which represents a virtual network configuration that explicitly specifies reachability, and upload them to the logically centralized controller.
2) Each pair of TPs exchanges symmetric keys to establish a secure channel. The key is being updated regularly through a state-of-art key management and distribution system that protects the secrecy of the zone transition while ensuring high-speed data transmission.
3) The on-site TP inspects all the packets to be transmitted to another zone. The TP acquires the corresponding zone transition policy from the controller and verifies if the transmission is authorized.
4) If a packet is authorized, the TP looks up the respective end-point TP, encrypts the packet along with the corresponding zone transition information, and forwards it across the WAN. Otherwise, the packet is dropped.
5) The remote-site TP then decrypts the packet and forwards it according to the enclosed zone transition information. The receiving TP could also verify the validity of the zone transition.

C. Threat Model

We consider a threat model in which attackers reside either on-premise (i.e., compromised end hosts) or are located outside of the cooperative networks. The goal of attackers is to access unauthorized zones to exfiltrate information assets, or disrupt networks and services. To achieve this goal, attackers can use the following strategies:

Unauthorized Access. Attackers may disguise as authorized entities to blind the security middleboxes and access restricted zones. A more sophisticated attack is to override security systems by directly injecting tampered policies.

Denial-of-Service. Here, the goal is to disrupt the target networks or services. Attackers can sabotage the core network systems, for example, by flooding security middleboxes that perform deep packet inspection. This might then lead to network performance degradation, causing denial-of-service for legitimate clients.

D. Assumptions

Public Key Infrastructure. A given enterprise network has a public key infrastructure (PKI). That is, the enterprise creates a trust model for its network infrastructure, acts as a trusted certificate authority (CA), and issues certificates for the core systems. Entities can retrieve and verify the public keys of the core systems. Open source projects, such as EJBCA [15] and OpenXPKI [46], are available for setting up enterprise-grade PKIs.

Secure Cryptography. Cryptographic primitives we use in MONDRIAN are secure; authenticity and secrecy remain intact unless the cryptographic keys are exposed.

Time Synchronization. Core entities within the cooperative network have loosely synchronized system clocks with a precision of hundreds of milliseconds (e.g., using network time protocol). Time synchronization is mainly used to constrain the validity of cryptographic keys.

Secure MONDRIAN Operation. All MONDRIAN entities including controllers and TPs are operating securely (i.e., not controlled by the adversary), and the security policy is correct.

IV. ARCHITECTURE

In this section, we present the MONDRIAN architecture and the underlying protocols in detail. Later, we describe our key-establishment system that enables rapid key derivation and distribution in large networks.

A. MONDRIAN *Bootstrapping*

The bootstrapping procedure is performed when a new zone or a new remote site (a group of zones along with a TP) joins the network.

Zoning Policy. In MONDRIAN, an IP subnet corresponds to exactly one network zone, such that the zone identifier ($zoneID$) a host belongs to can be identified by the host IP address. Ideally, IP addresses of each host in an enterprise network are unique, ensuring inter-zone (inter-VLAN) routing. Nonetheless, the same private address spaces with network address translation (NAT) devices could be a part of different zones, and they are distinguishable by their translated IP addresses (see §VIII-A). The way zones are described here corresponds to how enterprises typically segment their networks today, upholding backward compatibility and consequently deployability. Furthermore, MONDRIAN does not depend on the specific layer 2 protocols used at each site.

Every zone is under one administration domain, e.g., company A. In case multiple companies want to collaborate

by sharing certain network zones, such as company A allows access to zone A from company B's zone B, it is company A that creates and registers the zones transition policy to A's controller. Company B simply follows the policy.

Zone Transition Policy. To allow network operators to explicitly express their zone transition policies, we consider both denylist and allowlist-based policy definition. This is a mature approach commonly used in modern network management systems, enabling flexible and agile orchestration of complex networking policies. The following depicts the zone transition policy format:

$$< zoneID_{dst}, port_{dst}, zoneID_{src}, port_{src}, proto > \Rightarrow \quad < action > \tag{1}$$

action determines the corresponding action for the given source and destination zone pair, e.g., `forwarding`, `drop`, and `established`. Similar to *iptables* rules, `forwarding` would allow any incoming packets from the source zone whereas `drop` discards all traffic. `established` allows incoming traffic for all established connections.

The five-tuple expression allows network operators to define zone transition rules at application granularity. For example, hosts can reach a Web service running on servers in DMZ (e.g., <DMZ, 443, *, *, https> ⇒ <forwarding>), while another policy restricts access to other services available on the same servers (e.g., <DMZ, *, *, *, *> ⇒ <drop>). In rule conflict, longest tuple matching is performed to select an entry from a policy table; more precisely described rules have priority. The application-level policies maximize the flexibility of the network zone configuration.

Translation Point Initialization. A TP is initialized when a new remote site opens, prior to any communication between zones. The initialization process has mainly two goals: i) bootstrap a secure channel between TP and controller to exchange control-plane messages, and ii) establish tunnels with other TPs for data transmission.

Upon bootstrap, a TP performs a *client-authenticated TLS handshake* with its controller to exchange certificates, and agree on a cipher suite and a symmetric key. The TP finds the corresponding address in its configuration file. This means that, prior to first use, there needs to be an out-of-band setup where the TP is configured. This can either be done by the administrator of the controller shipping a pre-configured machine to the new remote location or by the remote location setting up a machine and granting the administrator management access to the given machine. Bootstraping TPs with multiple controllers will be described in §VIII-C with practical considerations such as controller discovery, TP migration, and policy consistency.

The controller synchronizes with TPs to keep their list of other TPs in the network up-to-date. To this end, the controller frequently pushes a list of relevant TPs with which a TP can communicate. This can be done either regularly (e.g., on a daily basis) or occasionally (e.g., when a new TP joins the network). TPs then establish a secure channel with new TPs. To prevent TPs from using a stale TP list, the shared TP list is associated with a time to live (TTL) value.

We ensure source authenticity of the MONDRIAN entities by leveraging PKI-based identities. The TPs and controller

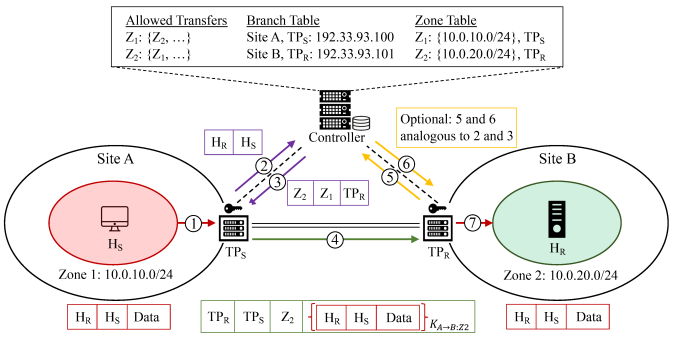

Fig. 3: Protocol details for data forwarding. The controller frequently updates TPs with the latest zone transition policies.

verify each other, preventing: i) a legitimate controller pushing information to an unauthorized TP, and ii) a legitimate TP connecting to a bogus controller or TP under an attacker's control. The main idea behind the source authenticity is that all the TPs and controllers obtain certificates for their IP address from a public key infrastructure (PKI). The administrator of the entities (e.g., an enterprise) runs a certificate authority, and issues the certificates before an entity bootstraps.

B. Protocol Description

We now describe how two end hosts within different remote sites are able to communicate. Figure 3 illustrates a protocol-level design including authorization, forwarding, and verification. A detailed header design follows.

Zone Transition Authorization. End hosts behind a TP operate as they normally would when they reside in a local network connected to a commodity gateway. That is, without any acknowledgment on network changes, a sender H_S sends a packet to the receiver H_R. If H_S and H_R are residents of the same subnet (namely the same zone), the packet will be directly steered to the destination by the local forwarding devices. If H_R is in a different subnet (or a remote site) however, the packet is first delivered to TP_S since TP_S is the gateway of H_S (① in Figure 3).

To determine if the packet is allowed to be forwarded to the given destination zone B, TP_S needs an explicit zone transition policy for the given source and destination pair. Ideally, the policy is cached in the TP_S's zone transition table. In case the cache misses, TP_S acquires the policy from the controller C as follows:

1) TP_S requests a zone transition policy from C (②):

$$TP_S \rightarrow C : H_R \mid H_S \tag{2}$$

2) C replies with the zone transition policy (③):

$$C \rightarrow TP_S : Z_R \mid Z_S \mid rule \mid TP_R \mid ExpTime \tag{3}$$

The controller consults the zone transition policy to see if the packet is allowed to be forwarded to H_R, more specifically to destination zone Z_B. To this end, the controller first checks the corresponding zone information, and matches it with the zone transition rules (Equation 1). The authorization result is then delivered to the requesting TP along with the corresponding source and destination zone identifiers (Z_R and

Z_S, respectively), the destination TP address (TP_R), and the expiration time ($ExpTime$) for the policy. $ExpTime$ can be an arbitrary number, but we consider it to be a small number used for policy freshness.

Data Forwarding. The TP discards the packet (Host Unreachable) if $rule = \texttt{drop}$. Otherwise, the TP looks up its routing table and transmits the packet*. There exist two types of zone transition cases: one for local (same-site) zone transition and another for remote zone transition as described in §II-A. For the local zone transition, the TP simply rewrites the Ethernet header by following the local layer 2 protocol, and forwards it through a corresponding network interface. Since the local network is assumed to be trustworthy, no additional packet processing is necessary apart from the authorization.

For the remote zone transition, the TP is responsible for the secure transmission of the packet towards the destination TP (④). Recall that it is important for the inter-domain zone transition packet to keep confidentiality and integrity in transmission. We therefore leverage secure tunneling, i.e., the IPSec tunnel mode [31], [32], meaning that the original packet is wrapped, encrypted, authenticated, and attached to a new IP header. The new packet layout is formed as follows:

$$EIP = \{H_R \mid H_S \mid payload\}_K, \tag{4a}$$

$$AT = MAC_K(Z_R \mid EIP), \tag{4b}$$

$$TP_S \rightarrow TP_R : TP_R \mid TP_S \mid Z_R \mid AT \mid EIP. \tag{4c}$$

The field name, Encrypted original IP payload (EIP), is self-explanatory. The original packet including IP header and payload is encrypted with a secret key K pre-shared between TP_S and TP_R. By encrypting the original packet and encapsulating it into the new IP datagram, we ensure confidentiality on the original payload as well as the host identities. We also introduce an Authentication Token (AT) which is placed in front of the EIP and contains a message authentication code (MAC) covering EIP and the destination zone identifiers. AT provides integrity over the entire packet except the outer IP header field which could be modified in transit.

The main difference to the Encapsulating Security Payload (ESP) in IPSec tunnel mode is that, rather than having site-to-site symmetric keys, we use site-zone pairwise keys. That is, the keys used for every triplet of $\{TP_{src} \mid TP_{dst} \mid zoneID_{dst}\}$ differs, providing a variety of unique symmetric keys even for the same pair of TPs. In addition, by conveying only $zoneID_{dst}$ in the header, zone pair information, which could lead to the potential disclosure of the zone structure and their transition rules, is not exposed.

Verification. The destination TP performs two steps of verification upon packet arrival: authentication and authorization. By extracting the quartet information from the header, TP_R first derives the corresponding symmetric key and recalculates AT to see whether the MAC matches the original AT value. This step is used to verify packet integrity as well as authenticity since only the two parties can derive the same key. If the match fails, either the packet integrity is compromised or source authentication failed, causing the packet to be discarded.

*Note that the TP is not involved in the routing decision; the TP is an add-on application running on top of the legacy gateway, performing the zone tranfer policy check and tunneling between the gateways. Packet routing and forwarding is independently done by layer-2 and 3 protocols.

To further verify authorization, TP_R obtains H_S and H_R by decrypting EIP, and verifies if H_S is authorized for the zone transition towards H_R. Similar to TP_S, TP_R might send a request to its controller (⑤) to acquire the authorization policy (⑥) when the policy is missing in its zone transition table (Equations 2 and 3).

In principle, MONDRIAN is constructed under a single administrative domain such that all the core entities, i.e., TPs and controllers, are trustworthy. One of the main advantages of this trust model is that the authorization check performed by the sender side TP is also trusted by the receiver-side TP, therefore not requiring verification of the zone transition authorization. Upon receiving a packet, TP_{dst} checks the authenticity of the packet, decrypts EIP, and forwards the original IP packet to the destination host. This trust model simplifies the entire verification process significantly by omitting the authorization step which requires an additional challenge-response protocol to the controller, improving practicality for TPs running at small branches limited in operational resources.

C. Key Management

In order for the TPs to create and verify authenticators based on symmetric cryptography we need a scheme to distribute keys amongst them. Ideally, the keys used for every triplet of $\{TP_{src} \mid TP_{dst} \mid zoneID_{dst}\}$ should be different. Additionally, ease of key management is a major concern as key distribution mechanisms in today's Internet, such as IKE [39], [22], [30], are complicated and introduce management overhead; the number of symmetric key pairs increases quadratically with the number of zones, which hampers scalability. To alleviate these problems we propose a key management system based on PISKES [56].

Major modifications to PISKES for MONDRIAN key management stem from the following requirements: first, in the context of network zoning, we require a high degree of confidentiality on top of authenticity to protect sensitive information. PISKES mainly targets authenticity for network entities, not confidentiality. Second, it is unlikely that an AS wants to have a dedicated key-establishment service for each enterprise, while PISKES's key establishment relies on the asymmetric key pair issued to each AS. The AS-driven key establishment approach would require additional management overhead and deployment for ASes. Third, MONDRIAN requires key management at zone granularity, while PISKES is intended to support key exchanges at a finer granularity (e.g., per host or application). Thus, design simplification also simplifies the architecture, reducing functional complexity and enhancing management scalability.

Driven by this, we redesign the PISKES's key-derivation architecture removing the AS dependency (i.e., the AS keys and the dedicated key servers at each AS), meaning that an enterprise has full control over distributed network zones and does not need to trust other ASes for inter-zone networking. Additionally, we simplify the key derivation to work at zone granularity, supporting faster key derivation while providing the same level of security.

Key Hierarchy. PISKES introduces a key hierarchy that allows services to efficiently derive symmetric keys. We adapt the

concept to support key derivation in the context of network zoning. The key hierarchy is as follows:

- 0th-level key: S_{TP} is the secret value generated by each TP individually.
- 1st-level key: a TP derives different symmetric keys for other TPs from the local secret value S_{TP}. The derived symmetric keys are called first-level keys and are calculated as

$$K_{A\to B} = PRF_{S_{TP_B}}(A), \tag{5}$$

  where B stands for the receiving TP address and PRF is a secure pseudo-random function. Since only one of the two parties can derive this key, it is necessary for the other party, in this case A, to fetch the shared symmetric key by contacting B[†].
- 2nd-level key: from the first-level keys, second-level keys are derived to provide diverse symmetric keys for each zone within the same source and destination TP pairs. The second-level keys are calculated as

$$K_{A\to B:Z} = PRF_{K_{A\to B}}(Z). \tag{6}$$

  Z is the zone ID of the target zone where the destination host resides.

This hierarchical key structure is beneficial in multiple aspects: first, it delivers key diversity. Since a second-level key is bound to a destined zone, it enables each zone to have different keys even for the same pair of source and destination TPs. Second, it is easy for TPs to efficiently derive the symmetric keys as all the required inputs to derive the key (i.e., local and remote TP address, and destination zone) are contained in the packet header. In particular, a remote TP thus can derive the key directly from the packet header without a memory lookup. Finally, since all second-level keys are derived directly from first-level keys, the system scales linearly with the number of TPs, not the number of zones, achieving scalability.

Bootstrapping Keys. Each TP randomly generates a local secret value S_{TP}, the root of the TP-specific key hierarchy. Since the first and second-level keys are derived from the secret value recursively, they inherit the randomness and secrecy of S_{TP}. We suggest using a true random number generator. The randomly generated secret value never leaves TP premises and is frequently renewed, e.g., on a daily basis, to achieve perfect forward secrecy [26], [55].

Key Establishment. Key establishment precedes the first data transmission. To establish a first-level key, the source TP initializes the key exchange protocol by sending a key exchange request:

$$req = A \mid B \mid ValTime, \tag{7a}$$

$$TP_A \to TP_B : req \mid \{H(req)\}_{K_A^-}, \tag{7b}$$

where $ValTime$ represents the validity period of the request. The hash value of the request is signed with the requesting TP's private key K_A^-, such that the receiving TP can verify the authenticity of the request packet. Recall that, for authenticity of MONDRIAN entities, each TP verifies the public key based on a certificate issued by the trusted CA, i.e., the controller.

Upon receiving the key exchange request, TP_B verifies the source authenticity and checks the validity of $ValTime$. If the request is valid TP_B derives a first-level key from the local secret value S_{TP_B} and replies back to the requester. The reply packet is formed as follows:

$$K_{A\to B} = PRF_{S_{TP_B}}(A), \tag{8a}$$

$$rep = \{B \mid K_{A\to B} \mid ExpTime\}_{K_A^+}, \tag{8b}$$

$$TP_B \to TP_A : rep \mid \{H(rep)\}_{K_B^-}, \tag{8c}$$

where $ExpTime$ denotes the expiration time of the first-level key, K_A^+ is the TP_A's public key used for encryption, and K_B^- is the TP_B's private key to sign the reply packet. Finally, the requesting TP verifies the validity of the reply packet and caches $K_{A\to B}$ until it expires.

Ideally, a TP prefetches all the first-level keys for other TPs it wishes to communicate with. The TP acquires the list of active TPs from its controller and initiates the key exchange protocol for these TPs in advance. This is feasible because the number of TPs for an enterprise is surely limited; for example, the total number of branches that the Bank of America has in 2019 is approximately 4.6k [62]. Each branch would need at least one TP, which means that a TP needs to prefetch more than 4.6k first-level keys. Nonetheless, on-demand key fetching is also possible. In particular, when the current first-level key expires in the middle of on-going data transmission or a new TP joins, a key exchange is initiated.

TPs are also responsible for second-level key establishment. However, this does not require any key exchange protocol. Upon data transmission, source and destination TPs are able to dynamically derive the same second-level key for the destined zone from the shared first-level key as shown in Equation 6.

V. IMPLEMENTATION

We now describe the implementation details of each component in MONDRIAN. We implemented a prototype that comprises a software-based gateway and controller. The main development language is golang 1.14.1 [20], and we used SQLite3 [61] for the database. To secure control-plane channels, we also leveraged TLS 1.3 [55]. The prototype is publicly available [52].

Our implementation builds on top of the SCION architecture [49]. The implementation decision has been driven by the following reasons: i) SCION provides network programmability along with the separation of control and data plane, ii) SCION comes with an embedded PKI system that can be utilized for our key management system, and iii) the opensource version of the PISKES system [51] as well as a software-based gateway working with SCION are available, thus enabling rapid prototyping.

A. Translation Point

To implement a prototype of TP, we extend the SCION-IP Gateway (SIG) [59]. The main functionality of the SIG is to encapsulate legacy IP packets into SCION packets and vice

[†]We note that, in contrast to PISKES, the arrow direction in the notation indicates the communication direction for which the key is used.

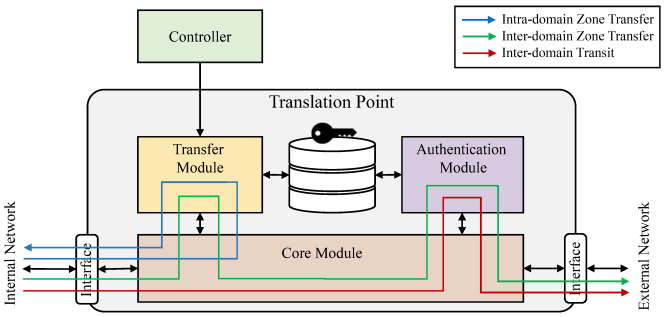

Fig. 4: An overview of the modularized TP implementation. Major usecases are also indicated with colored arrows.

versa. In this context, a SIG acts as a gateway between an internal (legacy) network and an external (SCION) network. Since TP is designed as a gateway that bridges LAN traffic over WAN—the underlying inter-domain routing protocol is not relevant here—the functional aspects of TP meets with what SIG provides. To be integrated with SIG, TP mediates between the UNIX socket and SIG socket, and performs zone transition authorization and verification for all incoming/outgoing packets. Figure 4 illustrates the implementation details of the modularized TP design that consists of three main modules: i) core module, ii) transition module, and iii) authentication module.

Core Module. The core module is the main loop of TP. It reads packets from the UNIX socket and redistributes them to the corresponding interfaces. More precisely, when receiving packets from the internal network, it retrieves metadata (further illustrated in Appendix A) from the raw packet, and hands over to the transition module. If the zone transition is authorized ($return = 1$), the packet is then either forwarded back to the internal network or, in case the given destination is in a remote zone, once again handed over to the authentication module to be prepared for secure transmission. For packets coming from the external network, TP first calls the authentication module for verification of the conveyed authentication token. Packets with invalid tokens are simply discarded.

Transition Module. The main objective of this module is to check the zone transition rules. The transition module communicates with its controller to maintain a list of up-to-date zone transition policies. To this end, it establishes a TLS channel with the controller, downloads policies, and populates the database. We implemented the transition module to support different drop-in options using APIs.

- No-Op: This is for a setup in which no inter-domain zone transition is required, but only inter-domain zone extensions.
- Standard: This mode would perform an authorization check for the requested zone transition based on the source and destination IP addresses.
- Firewall: If needed, the module could be instantiated as a full-fledged firewall.

Authentication Module. For inter-domain packet transmissions, the authentication module issues an authentication token for the packet. It (ideally) caches the first-level keys prefetched from other TPs and derive a second-level key to generate the authentication token. Inversely, for packets from other TPs, it derives the corresponding 2nd-level key and verifies the delivered authentication token.

Database. The TP's database consists of three tables: *Zone Table*, *Zone Transition Table*, and *Key Table*. The zone table is used to map hosts to their corresponding zones. For a fast table lookup, we leverage Radix Trees (also known as a trie or compact prefix tree), `cidranger` [8] in particular. The zone transition table is a database in which the zone transition rules are stored. The two tables are populated with the information acquired from the controller. The key table is where the shared first-level keys are stored.

Policy Miss. The current TP's implementation keeps sessions associated with a certain remote site. Packets from the interface are added to the ring buffer of their corresponding session. Each session keeps a set of worker threads handling its packets. If there is a policy miss in lookup, the worker handling the packet will initiate a policy fetch from the controller while the packet is kept in memory. It could happen that all workers of a given session are busy handling packets of a stream for which the policy is missing. This would stall communication with the corresponding remote entity until the policy is installed locally. Other sessions remain unaffected.

B. Controller

We implemented the controller as a Web server written in golang with an SQLite database storing the zone information and transition policies. The controller offers an API which allows TPs to fetch zoning information via HTTPS GET requests.

APIs. The endpoints of interest are: i) */api/get-subnets* and ii) */api/get-transitions*. Using these endpoints TPs fetch IP subnet and zone transition rules. Important to note is that the controller only hands out the subset of the full set of rules which is required for the requesting TP to be operational. This minimizes the size of data transmissions and also improves security by not disclosing the full network view to every TP. For every call to the API the controller first verifies the authenticity of the caller before the request is forwarded to the corresponding handler. The handlers then load the requested data from the database and send it to the caller as JSON-formatted bytes.

Database. The database consists of four tables (Zones, Sites, Subnets, Transitions), each describing one of the core elements of the architecture. The database schema is listed in Appendix B. An abstraction layer written in golang allows the controller to interface with the database using high-level calls. The abstraction layer makes use of transactional queries to ensure consistency even in the event of errors. Furthermore, the abstraction uses prepared statements for insertions, deletions and retrievals of data. This protects against SQL-injections and improves the speed of queries.

C. Authentication Token

The MONDRIAN packet format follows the IP tunneling conventions of encapsulating the original packet with a new outer IP header that indicates the two tunnel endpoints as the

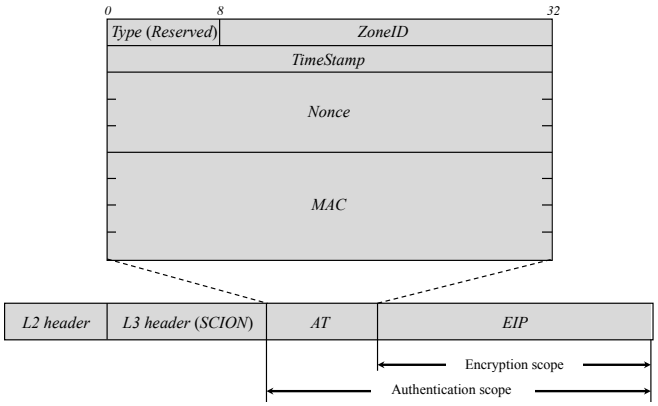

Fig. 5: MONDRIAN packet format for secure tunneling.

new source and destination. The original packet is encrypted and then authenticated along with the new packet header fields. Figure 5 shows the detailed packet structure and coverage of the confidentiality and integrity mechanisms.

The authentication token starts with one byte of reserved space for a *Type* field. While currently unused this will be useful in the future for distinguishing variations of the authentication token. *ZoneID* depicts the 3 byte long zone identifier of the destination zone. It is used by the receiving TP to derive the correct key for MAC verification and decryption. The next 4 bytes are occupied by a *TimeStamp* which is added by the sending TP. It is the Unix time (unsigned 32-bit `time_t`) at the point of sending the packet. The receiving TP uses this timestamp to reject replayed packets. A *Nonce* (12 bytes) follows the timestamp.

The nonce as well as the previous three token fields and the data to be encrypted (*EIP*) serve as input to a *Galois Counter Mode* (GCM) algorithm with an underlying *AES-128* block cipher as cryptographic primitive. This mode of operation is widely adopted for its performance and to achieve *authenticated encryption with associated data* (AEAD). Here it provides authenticity over the header fields (*Type*, *ZoneID*, *TimeStamp*) and the data in *EIP* while *EIP* additionally also gets encrypted. The 16 byte *MAC* generated by GCM is the last field in the authentication token. Both the nonce and the MAC sizes follow the guidelines recommended by NIST SP 800-38D [14].

VI. EVALUATION

A. System Benchmarks

We first conduct microbenchmark tests to evaluate the performance of TP including the key derivation, packet authentication, and authorization. For reproducible evaluation, we leverage the standard benchmark library `testing` officially supported by golang. The benchmarks are conducted on commodity machines equipped with an Intel i7 2.9 GHz CPU, 16 GB memory, and a 1 GbE NIC.

Authorization. From a technical perspective, the zone transition authorization is a database lookup consisting of three tree searches; upon receiving the packet metadata from the core module, the transition module first looks up the corresponding zone identifiers for the source and destination addresses, and then compares them to the zone transition policies. The authorization performance is therefore dependent on the lookup time of the policy database.

Table I shows the benchmark results of database lookups for different quantities of policies. Each benchmark ran over two million iterations and kept the mean value. The authorization check takes approximately 300 to $500\,\mathrm{ns}$ per packet, which is a notable result considering: i) a lookup consists of three tree searches, ii) the result is from a high-level language implementation, and iii) the size of the test set is increased by a factor of 1000. Interestingly, a lookup failure is commonly 24 to 31 % faster than a successful lookup. This implies that abnormal packets with invalid zone transition requests can be quickly discarded.

Key Derivation. We investigate the key derivation performance. Recall that the key derivation proceeds differently for sender and receiver (See §IV-C). The receiver directly derives the first and second-level key from the local secret, whereas the sender-side key derivation comprises two steps: fetching the first-level key from the key table and deriving the second-level key from the first-level key. From a scalability perspective, we increase the number of stored first-level keys up to $100\,\mathrm{K}$.

Table II shows the average key derivation time for the benchmark tests (each number reported represents the average of over a million trials). The time duration to obtain the first-level key at the source TP requires $154 \sim 161\,\mathrm{ns}$ (key lookup), while at the destination TP it requires $188 \sim 197\,\mathrm{ns}$ (key derivation). While key derivation can be optimized to require only tens of nanoseconds [56], the high-level golang implementation achieved a lower performance. The time duration to derive the second-level key from the first-level key is the same at source and destination TP, and it amounts to $104\,\mathrm{ns}$. Since the first-level key also requires a time validity check, the second-level key derivation is slightly faster. We observe that for pratical network sizes, the key derivations can be achieved in less than $300\,\mathrm{ns}$ with unoptimized code.

Authentication. The additional processing time for packet encryption and decryption is shown in Table III. In summary, it requires approximately 1.5 to $2.5\,\mu\mathrm{s}$ to authenticate various sizes of packets. We note that the processing overhead occurs for all the tunneling technologies that provide confidentiality for data transmission. The processing time can be minimized with implementations using the Data Plane Development Kit (DPDK) [13] or by leveraging hardware dedicated to cryptographic operations.

B. Network Benchmarks

So far, we evaluated the performance of MONDRIAN operations. Since a different set of operations needs to be applied depending on the zone transition usecase, it is also important to investigate the overall network performance for handling different types of zone transition packets. We now benchmark the actual network performance for both intra-/inter-domain zone transition cases.

Latency Inflation. Figure 6 illustrates the network benchmark results for the intra-domain zone transition where the source and destination zones are within the same local network, such that the TP only performs zone transition authorization.

TABLE I: Benchmark results for the zone transition policy lookup (ns).

# of Policies	Valid	Invalid
100	307	236
1 K	405	264
10 K	423	293
100 K	497	375

TABLE II: Key derivation times for different network sizes (ns).

# of Keys	1st key (src)	1st key (dst)	2nd key
100	154	188	104
1 K	155	188	104
10 K	161	197	103
100 K	157	188	104

TABLE III: Processing times for the encryption/decryption (ns).

Packet Size (byte)	Encryption	Decryption
64	801	623
512	1006	701
1024	1279	796
1500	1557	950

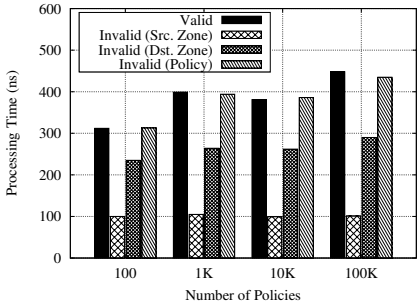

Fig. 6: Processing time for intra-domain zone transition.

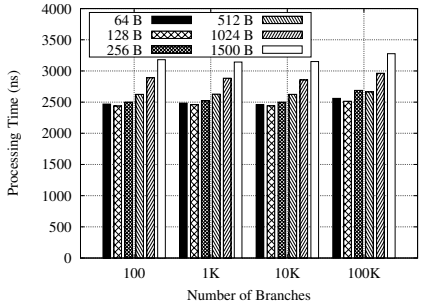

Fig. 7: Processing time on TP_S for inter-domain zone transition.

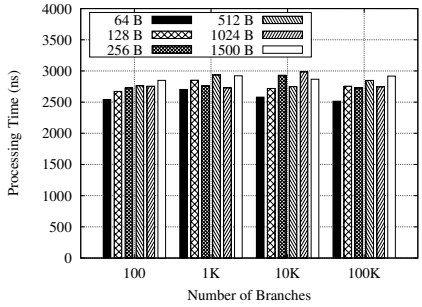

Fig. 8: Processing time on TP_R for inter-domain zone transition.

Since no cryptographic operations are involved, the additional latency is negligible (as a legitimate zone transition takes $300\,\mathrm{ns}$ to $450\,\mathrm{ns}$). There might be an authorization abort due to a lookup failure that could be caused by the following three reasons: no source zone ID, destination zone ID, or zone transition policy. In our prototype, the lookups are performed sequentially and thus there are different processing overheads ($100\,\mathrm{ns}$ to $600\,\mathrm{ns}$) depending on when a lookup failure occurs. Nevertheless, if there exists no valid zone transition policy, the packet will be simply dropped and therefore no additional latency is caused.

For inter-domain zone transition cases, we benchmark the overall network latency increase during TP operations including packet parsing, key derivation, authorization, and authentication. Figures 7 and 8 depict the processing delay from sender-side TP and receiver-side TP, respectively. From the results, we make the following observations: first, the overall latency inflation that MONDRIAN introduces is insignificant ($\sim 3\,\mu$s). Second, MONDRIAN scales well with the size of the network, i.e., the number of branches. We do not see any notable performance degradation ($\leq 200\,\mathrm{ns}$) for realistic values. Third, the size of a packet is the primary factor for the latency increase as expected for all data-plane devices. The packet size has a small latency increase factor of 1.28 (i.e, $2.4\,\mu$s to $3.8\,\mu$s). Lastly, we observe no significant bias in network performance between sender- and receiver-side TPs.

Forwarding Performance. We further investigate the actual forwarding performance for various packet sizes including a representative mixture of Internet traffic (iMIX) [43], [60]. Figure 9 shows the results. The baseline is the forwarding performance without TP operations. The other bars represent the forwarding performance for intra-domain zone transition (with authorization only) and inter-domain zone transition (with authorization and authentication) respectively.

For 64-byte packets which demonstrates the highest packet rate, and thus requiring the most extreme packet processing, the intra-domain zone transition exhibits a throughput degradation of only $9\,\%$. It also achieves 96 to $100\,\%$ of throughput for other packet sizes. These results are expected because the TP performs only authorization for the intra-domain zone transition packets, which would increase the processing delay by less than $500\,\mathrm{ns}$; considering that a typical intra-domain packet transmission incurs a few milliseconds of latency, the additional delay is negligible.

On the other hand, the inter-domain zone transition degrades the throughput by $30\,\%$ for the smallest packets. Although the degradation diminishes as the packet size increases, the performance still degrades by $28\,\%$ for iMIX traffic. To investigate the main degradation factor, we compare the amount of transmitted data (goodput) and the total bits transmitted including all the network headers (throughputs) as shown in Figures 10 and 11. From the comparison, we observe the following: i) the inter-domain zone transition achieves a similar throughput to the baseline if the extra headers are considered, ii) the performance degradation is caused by not only the additional processing delay but also transmission delay of the extra headers, and therefore iii) MONDRIAN performs similar to today's tunneling applications while providing the security policy enforcement for network zoning.

C. Comparison to the Current Practice

We finally compare MONDRIAN with IPsec (ESP-tunnel mode) to demonstrate its space and time overhead. Table IV shows the comparison results.

Time Overhead. We estimate the processing time of cryptographic operations for 64 to 1024-byte packets. For fairness, we consider the same cipher suite for both approaches, along with the same size of the cryptographic key (i.e., 128-bit AES-GCM). Note that the performance of the cryptographic operations are empowered by AES New Instructions (AES-NI [57]), a hardware acceleration technology integrated into many processors nowadays. We make the following observations: first, MONDRIAN is 68 to 287 ns slower than IPsec in encryption and decryption. Second, with respect to the packet size increase, IPsec shows a significant increase in

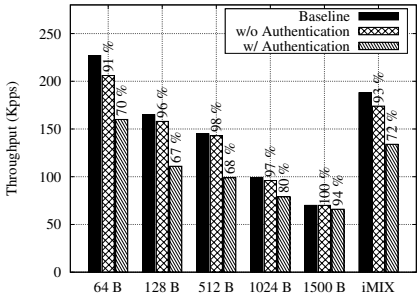

Fig. 9: Forwarding performance of TP for various size of packets.

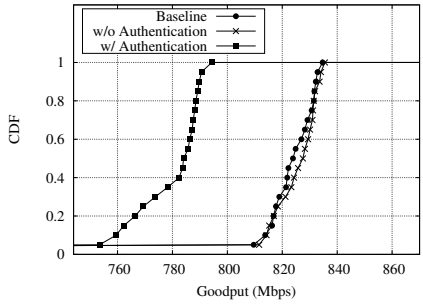

Fig. 10: CDF of goodput for 1400-bytes of maximum segment size (MSS).

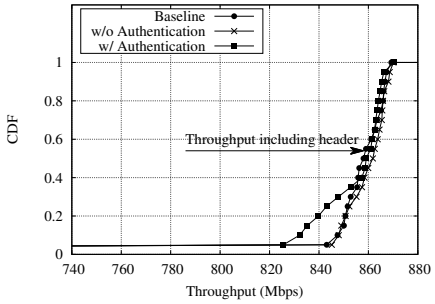

Fig. 11: CDF of throughput including extra header fields.

TABLE IV: Time and space complexity in comparison. n denotes the number of TPs and m represents the number of zones, where $m \geq n$.

	Cipher Operation ($64 \sim 1024$ Byte)		Extra Header (Byte)	Memory (Byte)
	Encryption (ns)	Decryption (ns)		
MONDRIAN	$142 \sim 513$	$114 \sim 395$	56	$16 \times n^2$
IPSec	$46 \sim 226$	$46 \sim 213$	58	$16 \times m^2$

the processing time, i.e., up to $400\,\%$, while MONDRIAN shows an increase of $200\,\%$. This is a reasonable performance considering that our current implementation relies on an open-source cryptography library.

Space Overhead. IPsec ESP-tunnel mode requires a minimum of 58 bytes of extra headers, including 20 bytes of an outer IP header and 38 bytes of ESP header, while MONDRIAN introduces 56 bytes of extra headers (i.e., 20 bytes of an outer IP header and 36 bytes of AT header) for tunneling. Since both mechanisms could have additional padding depending on the size of the original payload, the bandwidth overhead is hardly distinguishable. In contrast, we notice a significant disparity in memory requirments. To establish secure channels between m zones, IPsec requires a total of m^2 keys, whereas MONDRIAN only needs n^2 keys where n represents the number of TPs. As typically $m \gg n$, MONDRIAN outperforms IPsec in real-world networks.

VII. SECURITY ANALYSIS

We analyze the security properties that MONDRIAN provides, considering the threat model introduced in §III-C. The attack classes described here are twofold: i) attacks to infiltrate restricted zones without a proper permission, and ii) disruption attacks that prevent availability.

A. Infiltration without a Permission

Man-In-The-Middle Attack. To interpose "in the middle", an attacker could initiate independent communication channels with two TPs and relay messages between them. By using the attacker's public key for the channel establishment, the attacker may attempt to generate valid packets to bypass the TP's authentication check.

MONDRIAN's PKI design prevents such MITM attacks. A MITM attack can succeed only if the attacker convinces each TP that they are talking to each other. In the process of secure channel establishment, however, TPs authenticate each other using the certificates issued by the mutually trusted CA (e.g., enterprise). Since the attacker's public key cannot be certified by a valid certificate issued by the CA, a MITM will fail.

Packet Replay. Attackers can observe valid MONDRIAN packets and then attempt to reuse them to transmit attack traffic. Nevertheless, the validity of the reused packet header will be lost once the payload is changed—recall that the authentication scope covers the entire packet including AT and EIP as shown in Figure 5—and therefore attackers cannot successfully pass the authentication check at the recipient TP.

Brute-force Attack. Finally, the attacker could attempt to brute-force the key used for authentication or the MAC. As we are using 128-bit cryptographic keys and MACs, such attacks are currently infeasible. To achieve resilience to quantum computers, the key size would need to be doubled, however.

B. Denial-of-Service Attacks

Exhaustion Attack. In principle, flooding the TP's authentication process could be an effective attack vector. Attackers may attempt to forward a large number of packets to the target TP in order to exhaust the TP's resources. Even if the attack packets contain invalid authentication tokens, the TP still needs to verify these tokens, therefore wasting resources, preventing legitimate packets from getting through.

To be resilient to such attacks, we suggest operating multiple TPs at the entry points of cooperative networks. Multiple TPs enable network operators to load balance and switch over to another TP in case of a link (or a TP) failure (see §VIII-D). In fact, many data centers and large enterprise networks already employ equal-cost multipathing (ECMP) [65], [25] along with multiple gateways and ToRs (Top-of-Rack switches) to provide reliable intra-networking services. In addition, to mitigate possible DoS attacks from the public Internet, Microsoft implemented a global ECMP infrastructure [42]. Path-aware networking along with multi-path communication also enables active switching to different entry points, if some fail or are under DDoS attack. [11], [66].

VIII. PRACTICAL CONSIDERATIONS

In this section, we discuss some practical considerations including functional and management aspects, along with various deployment scenarios describing how MONDRIAN can be realized on today's enterprise infrastructure.

A. NAT Devices

Multiple hosts connected through a NAT device appear as a single host to the TP. *Internal* NAT devices hardly pose a problem since the hosts under that NAT device are all subject to the same subnet and therefore belong to a single network zone. Network operators establish a security policy on the translated IP address, such that TPs are able to authenticate the zone transit requests from/to a host under the internal NAT device.

However, an *external* NAT device located in an external network, e.g., carrier grade NAT, could affect the TP's secure tunneling ability. The translated TP's IP address would cause a MAC verification failure—recall that each symmetric key binds to the triplet including TPs' IP addresses as described in §IV-C. This, however, can be addressed by enforcing TPs to use their public IP addresses to derive the symmetric keys. The keys are still secure since the first-level key from which the pairwise keys are derived is exchanged with the CA-certified public keys. Discovering the translated TP address is also not a problem thanks to the controller informing senders about the recipient TP's address (see, Protocol ③ in Figure 3).

A potential operational failure is where multiple TPs are behind the same NAT device. This would lead to TPs having identical keys from the view of remote TPs. One possible solution would be to use a unique TP identifier instead. Since all such TPs would be under one administrative domain, assigning unique TP identifiers upon bootstrapping is feasible. Then, the TPs convey their identifiers in the AT header field alongside the destination zone ID. This might slightly increase the size of the header, but does not degrade the security of the underlying authentication.

B. Tunneling Granularity

Secure tunneling can be realized in different granularities: i) site-to-site tunneling, ii) zone-to-zone tunneling, and iii) site-to-zone tunneling. In the following we motivate our design choice of site-to-zone tunneling in MONDRIAN.

Similar to IPSec VPN, site-to-site tunneling provides strong guarantees on communication security and privacy for two tunnel endpoints. However, from a flexibility and manageability standpoint, having a site-to-site tunneling architecture is not ideal. Every tunnel endpoint needs to share a key with every other endpoint with which it wishes to exchange data. This adds state to the endpoints that needs to be updated and synchronized. Adding a new site requires an update on all the other sites that wish to communicate with the new site. Then, yet another layer of security middleboxes (e.g., firewalls) are required to perform zone transition authentication since keys are do not designate a specific zone.

An alternative way of providing authentication is to use one key per zone. In this model, the TP would sign the data on behalf of the zones. In case of a zone transition, the TP would perform the transition and then use the key of the source/receiver zone pair. This approach has the benefit that sender and receiver are decoupled, as in principle any site that contains a given zone is able to decrypt data destined for that zone. Adding a new site would be as easy as fetching the right keys for the zones used in this site. This process is independent of all the other sites. However, a receiver TP needs to be able to fetch the right keys for the zones, which means the zone transition information must be visible, and thus an attacker could potentially learn the zone structure of the observed network. Also, a separate key per zone pair does not scale well as the number of zones can grow over 1000 in large networks.

Driven by these considerations, we designed the new concept of site-to-zone tunneling, which represents a middle ground combining the advantages of the two approaches, the notion of secure tunneling and zone transition authentication. The symmetric keys are distinguishable depending on the destination zone, while at the same time the zone-to-zone security policies are not being exposed. Thanks to the flexible and scalable key derivation scheme introduced by PISKES [56], the key establishment does not expand state, while still providing unique symmetric keys per zone.

C. Distributed Controllers

To avoid *single-point-of-failure*, logically centralized control-planes built on physically distributed instances are commonly being used. The most common approaches to realize distributed controllers can be broadly categorized into horizontal distribution [5], [41] and hierarchical distribution [23], [68]. Independent of which distribution architecture is used, we discuss location, coordination, and migration aspects of distributed controllers.

Location. The notion of a logically centralized control-plane offers flexibility in network design and management. A key design choice is placement of the (distributed) controllers, which could impact performance, reliability, and management scalability of a given network. There is comprehensive research on the controller placement problem considering practical issues from control latency to reliability, from cost-optimization to load balancing, etc [10], [24], [71]. Among those, we are mainly interested in the latency performance indicator; that is the latency between a controller and regional forwarding devices.

The best latency is achieved when each branch site has its own controller. By a placement near local TPs, the controller minimizes the TP-controller latency for the zone transition authorization protocol, allowing instant feedback for packet forwarding—we note that inter-controller communication for global coordination is commonly not latency sensitive. For the sake of control-plane security, the controller resides in a restricted zone to which only the local TPs and remote controllers can access. Although the per-site controller offers the best performance regarding policy enforcement for the data-plane, there might be a cost-efficiency problem for a large-scale network with thousands of branches.

Alternatively, we consider a sparse distribution model, e.g., on edge-cloud systems. Similar to today's cloud services, network operators running geographically distributed data centers can instantiate multiple controllers at the central point of regional branches. The control-plane latency overhead would be relatively high compared to a dense deployment model—if the data center edges are geographically diverse, the overhead could be minimized—but, in terms of cost-optimization and management scalability, this would be a more viable approach.

Coordination. It is important to keep consistency across the distributed controllers. Inconsistency in security policy might grant hosts with a low security clearance unauthorized access to restricted zones, resulting in leakage. With this in mind, we consider a consensus algorithm with strong consistency guarantees [47], [50], [58], where the security policy is dynamically shared/replicated across the distributed controller instances, ensuring consistent policy enforcement toward the data-plane devices. There are numerous open-source projects, such as Consul [9], Apache ZooKeeper [2], and ETCD [17] available.

TP Migration. To benefit from the distributed controller environment, a dynamic controller discovery process also becomes important. That is, TPs should be able to search a cluster of best candidates, diagnose the performances in terms of control latency, and seamlessly migrate to the best controller. To this end, we consider a two-step migration process: i) TP-driven control channel initialization, and ii) controller-driven TP migration.

TPs are responsible for establishing the first control plane channel with a controller. For example, a new TP (TP_{new}) has been configured to contact an initial controller acting as a first rendezvous point. The initial information contains the controller's IP address (C), the corresponding zone ID (Z_C), and the TP's IP address behind which the controller resides (TP_C). If the controller is located in a remote site (i.e., $TP_{new} \neq TP_C$), TP_{new} should connect with C through TP_C. Otherwise, e.g., C is within the same LAN or in a public network, TP_{new} can directly send C a request for control-plane channel establishment.

Once the TP joined the network, the controller then initiates a migration process to find the best controller (C_{best}) for TP_{new}. Upon a migration request broadcast by C, other controllers measure the possible latency to TP_{new} and reply back the results. Then, C elects C_{best} considering the latency measurements and the current load balance, and sends TP_{new} a `RoleChange()` request containing C_{best}, $Z_{C_{best}}$, and $TP_{C_{best}}$. Finally, TP_{new} swaps the best controller by establishing a new channel with C_{best}. The migration process is also applied when changes in the network are detected.

D. Distributed TPs

Network zoning with a TP may create another *single-point-of-failure*. From a reliability perspective, flattening and connecting all the zones with a TP could potentially eliminate redundancy for connectivity between zones. From a security perspective, the centralized TP—instead of several distributed security middelboxes—is a clearer target for adversaries.

However, both these concerns can be addressed by operating multiple TPs with advanced multipath-enabled layer-2 protocols (e.g., SPB and TRILL). This network design provides load balancing and enhanced resilience against a TP or link failure. If a TP is unable to continue data transmission, the underlying protocol redirects flows to another TP, ensuring continuous communication for end hosts [6]. TPs do not keep state, and therefore the TP conversion can be seamless and no state migration is required.

For defense in depth, zones can be nested, resulting in a hierarchical structure. MONDRIAN enables a hierarchical zone structure by employing nested TPs that are only accessible from upper-layer TPs, ensuring access control at multiple levels. In addition, MONDRIAN can coexist with other security middleboxes: for high-security zones, network operators can consider positioning additional security solutions behind a TP, facilitating a multitude of defense options.

E. Nonce Reset

The same nonce must never be used twice with the same key, otherwise the security of the cipher significantly decreases. In theory it is easy to create nonces that fulfill this requirement. One can simply use a counter which is increased for every invocation of the AEAD algorithm. In real systems this is not so easy to achieve since machines can crash and lose their state, specifically their nonce counter. We consider the following techniques to approach the problem.

- *Purely random nonce*: All bits of the nonce are used for randomness. Using this technique the probability of a nonce clash is very low. Still, there are no guarantees even if the system does not crash.
- *Counter paired with random sequence*: This technique divides the nonce into a counter and a randomized part. The random part is initialized after every restart and the counter is increased for every packet that is sent. After a crash, the counter starts from zero with the random part being initialized to a new random value. In case of a nonce clash, all subsequent nonces clash as well.
- *Reset points*: This technique uses all bits of the nonce for a counter and simultaneously defines specific reset points which are stored on non-volatile memory (NV-memory). The counter is incremented in memory and the next reset point is written to the NV-memory once the current reset point is passed. If a crash occurs, the counter restarts from the latest reset point.

F. Incremental Deployability

A new networking technology must satisfy the following requirements to be incrementally deployable: First, the new technology should require minimum changes on the network stack, especially of the end hosts. Except for security-concerned users, most users are not interested in updating their system, hampering incremental deployability. Second, it should provide early adopters an instantaneous incentive, and third, the incentive should be valid even in partial deployment. MONDRIAN satisfies these requirements, providing seamless incremental deployment capabilities. We outline here three incremental deployment strategies: a gateway deployment, a middlebox deployment, and software deployment.

Gateway Deployment. We consider a gateway deployment scenario that does not require changes from end hosts nor the local network infrastructure. The network operator deploys MONDRIAN using a gateway, which performs the required packet authentication and authorization operations. MONDRIAN takes a supportive role by complementing already installed lines of defense such as firewalls, IPS, and IDS. For instance, traffic can be pre-filtered by TPs before it reaches firewalls located deeper inside the network. This deployment

scenario leaves the enterprise's network intact while providing the security properties of MONDRIAN. The simple deployment already provides early adopters a clear incentive; secure and strong policy enforcement of zone translation with a flexible zone migration and dynamic access control.

Middlebox Deployment. MONDRIAN can also be used as a single, all-in-one solution providing packet filtering, tunneling, and routing within one middlebox device. We envision the middlebox to be positioned alongside the routers inside the network, serving a few hundred hosts. Such a deployment is especially interesting for small branch sites with much simpler network layouts. Here, MONDRIAN can drastically reduce the number of devices that need to be maintained.

Software Deployment. Given that *working-from-home* becomes a new normal in modern society, enterprises should allow their employees secure access from home to their network. We envision a software-based TP as a viable deployment option for such home users. Similar to VPN tools, the TP runs on the user machine and acts as a virtual gateway, performing secure tunneling and zone translation to interconnect with information systems in the enterprise network.

IX. RELATED WORK

The majority of literature in network zoning has focused on security enforcement architecture using middleboxes such as firewalls, IPS, and IDS. Conventional security middleboxes define restricted zones and filter unwanted traffic at the entry points of protected zones [7]. As information systems and the corresponding network functions get more complicated, the notion of distributed security systems has been introduced in the late 1990's [4]. Early approaches to protect only internal information systems from external threats have further evolved to mitigate sophisticated threats, for example insider attacks, rule tampering, application-level proxies, and denial-of-service attacks [38]. Later, with emerging network virtualization technologies and cloud computing environments, virtual firewalls and collaborative security enforcement kept getting attention from both academia and the industry [34], [69].

Secure network design with advanced technologies complicates network configuration and management. Notable efforts to simplifying the complexity involve automation tools enabling "top-down" network provisioning [64], [63]. For instance, PRESTO [16] constructs a router-native configuration by using configlets, configuration snippets that encode a high-level service description into the device-vendor-specific language. Later, SDN further simplified the network configuration and management via logically centralized control and network programmability [35], [44].

Despite the numerous research efforts, network zoning with security middleboxes has challenges with respect to performance—the iMIX throughput degrades by 40 - 75% on commodity products [29]—and misconfigurations [18], [67], [70]. With MONDRIAN, we address the challenges by leveraging a cryptographic-based policy enforcement and centralized policy orchestration, achieving scalable, effective, and cost-efficient network zoning.

Network isolation through network segmentation is another essential element of network zoning. To logically segment the physical network, network virtualization technologies are heavily used in today's Internet, in particular large enterprise networks and cloud computing environments. VLAN [28] is the most frequently used network segmentation technique. It logically segments a physical LAN into up to 4094 virtual LANs by tagging the layer 2 header with a unique VLAN identifier (VID). Later, Virtual eXtensible LAN (VXLAN) [36] has been introduced for better scalability; it expands the number of virtual LANs to up to 16 million by leveraging a 24-bit identifier. SPB [27] and Trill [1], [48] are layer 2 routing protocols enabling multi-path communication among virtual LANs within the same physical LAN. Although security is an important property in network segmentation, unfortunately it has not been treated as a major concern—without protection in membership or access control. To isolate each segment, the use of a large number of security middleboxes is necessary, thus increasing operational costs and management complexity.

As the closest related work, SVLAN [33] is an architecture enhancing security in network isolation by enforcing the receiver's consent towards incoming traffic. MONDRIAN provides several significant advantages. First, MONDRIAN is compatible with various Internet architectures, achieving practicality and deployability. Second, MONDRIAN provides enhanced security along with source authenticity, data confidentiality, and data integrity. Lastly, MONDRIAN improves manageability. The simplified zoning structure, advanced key establishment system, and centralized policy management simplify the network management.

X. CONCLUSION

Network zoning has long been recognized as the cornerstone of secure network operation and management. In the current practice, operators realize network zones with network segmentation technologies and security middleboxes. As information systems become more dynamic from a topological, operational, and functional perspective, however, the conventional network-zoning architectures face new challenges in terms of scalability and flexibility. In this paper, we have shown that lightweight policy enforcement for inter-zone communication is achievable. Following a constructive approach with a cryptographic foundation, it is possible to create a proactive alternative to the mostly reactive systems presently used in network zoning. In conjunction with MONDRIAN, verification based on firewalls becomes simpler because firewalls would only process a limited amount of (filtered) traffic. MONDRIAN consequently reduces the number of management points of distributed networks while retaining a high degree of security.

ACKNOWLEDGMENT

We would like to thank our shepherd, Patrick Traynor, and the anonymous reviewers for their insightful feedback and suggestions. We thank to Markus Legner and Giacomo Giuliari for helpful discussions that improved this research. We gratefully acknowledge support from ETH Zurich, and from the Zurich Information Security and Privacy Center (ZISC). This work was also supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-01697, Development of Automated Vulnerability Discovery Technologies for Blockchain Platform Security).

REFERENCES

[1] D. E. 3rd, T. Senevirathne, A. Ghanwani, D. Dutt, and A. Banerjee, "Transparent Interconnection of Lots of Links (TRILL) Use of IS-IS," RFC 7176, IETF, 2014.

[2] Apache ZooKeeper, https://zookeeper.apache.org/.

[3] J. N. Bakker, I. Welch, and W. K. Seah, "Network-wide Virtual Firewall using SDN/OpenFlow," in *Proceedings of IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2016.

[4] S. M. Bellovin, "Distributed Firewalls," *Login Magazine, Special Issue on Security*, 1999.

[5] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow *et al.*, "Onos: towards an open, distributed sdn os," in *Proceedings of the Workshop on Hot Topics in Software Defined Networking (HOTSDN)*, 2014.

[6] Y. Cai, L. Wei, H. Ou, V. Arya, and S. Jethwani, "Protocol Independent Multicast Equal-Cost Multipath (ECMP) Redirect," RFC 6754, IETF, 2012.

[7] W. R. Cheswick, S. M. Bellovin, and A. D. Rubin, *Firewalls and Internet security: repelling the wily hacker*. Addison-Wesley, 2003.

[8] Cidranger, https://github.com/yl2chen/cidranger.

[9] Consul, https://github.com/hashicorp/consul.

[10] T. Das, V. Sridharan, and M. Gurusamy, "A Survey on Controller Placement in SDN," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 472–503, 2019.

[11] S. Dawkins, "Path Aware Networking: A Bestiary of Roads Not Taken," IETF Internet Draft, 2018.

[12] J. Deng, H. Hu, H. Li, Z. Pan, K.-C. Wang, G.-J. Ahn, J. Bi, and Y. Park, "VNGuard: An NFV/SDN Combination Framework for Provisioning and Managing Virtual Firewalls," in *Proceedings of IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, 2015.

[13] DPDK Project, "Data Plane Development Kit," https://dpdk.org.

[14] M. Dworkin, "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC," NIST Special Publication 800-38D, 2007.

[15] EJBCA, https://svn.cesecore.eu/svn/ejbca/trunk/ejbca/.

[16] W. Enck, P. McDaniel, S. Sen, P. Sebos, S. Spoerel, A. Greenberg, S. Rao, and W. Aiello, "Configuration management at massive scale: System design and experience," in *Proceedings of the USENIX Annual Technical Conference (USENIX ATC)*, 2007.

[17] ETCD, https://github.com/etcd-io/etcd.

[18] S. K. Fayaz, T. Yu, Y. Tobioka, S. Chaki, and V. Sekar, "BUZZ: Testing Context-Dependent Policies in Stateful Networks," in *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2016.

[19] D. Felsch, M. Grothe, J. Schwenk, A. Czubak, and M. Szymanek, "The Dangers of Key Reuse: Practical Attacks on IPsec IKE," in *Proceedings of the USENIX Security Symposium*, 2018.

[20] Go 1.14, https://golang.org/doc/go1.14.

[21] A. Gontarczyk, P. McMillan, and C. Pavlovski, "Blueprint for Cyber Security Zone Modeling," *Information Technology in Industry*, vol. 3, no. 2, pp. 38–45, 2015.

[22] D. Harkins and D. Carrel, "The Internet Key Exchange (IKE)," RFC 2409, IETF, 1998.

[23] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications," in *Proceedings of the workshop on Hot Topics in Software Defined Networks (HOTSDN)*, 2012.

[24] M. He, A. Varasteh, and W. Kellerer, "Toward a Flexible Design of SDN Dynamic Control Plane: An Online Optimization Approach," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1694–1708, 2019.

[25] C. Hopps, "Analysis of an Equal-Cost Multi-Path Algorithm," RFC 2992 (Informational), IETF, 2000.

[26] IEEE Std., "1363-2000: Ieee standard specifications for public-key cryptography," 2000.

[27] IEEE Std., "802.1aq-2012: Local and metropolitan area networks-shortest path bridging," 2012.

[28] IEEE Std., "802.1q-2018: Local and metropolitan area networks-bridges and bridged networks," 2018.

[29] Juniper, "Security Products Comparison Chart," https://www.juniper.net/us/en/local/pdf/datasheets/1000265-en.pdf.

[30] C. Kaufman, "Internet Key Exchange (IKEv2) Protocol," RFC 4306, IETF, 2005.

[31] S. Kent, "IP Encapsulating Security Payload (ESP)," RFC 4303, IETF, 2005.

[32] S. Kent and K. Seo, "Security Architecture for the Internet Protocol," RFC 4301, IETF, 2005.

[33] J. Kwon, T. Lee, C. Hähni, and A. Perrig, "SVLAN: Secure & Scalable Network Virtualization," in *Proceedings of the Symposium on Network and Distributed System Security (NDSS)*, 2020.

[34] A. X. Liu and F. Chen, "Collaborative Enforcement of Firewall Policies in Virtual Private Networks," in *Proceedings of the ACM Symposium on Principles of Distributed Computing*, 2008.

[35] G. Lospoto, M. Rimondini, B. G. Vignoli, and G. Di Battista, "Rethinking virtual private networks in the software-defined era," in *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015.

[36] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks," RFC 7348, IETF, 2014.

[37] Q. Mahmoud, *Cognitive networks: towards self-aware networks*. John Wiley & Sons, 2007.

[38] T. Markham and C. Payne, "Security at the Network Edge: A Distributed Firewall Architecture," in *Proceedings of DARPA Information Survivability Conference and Exposition II. (DISCEX'01)*, 2001.

[39] D. Maughan, M. Schertler, M. Schneider, and J. Turner, "Internet Security Association and Key Management Protocol (ISAKMP)," RFC 2408, IETF, 1998.

[40] A. Mayer, A. Wool, and E. Ziskind, "Fang: A Firewall Analysis Engine," in *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2000.

[41] J. Medved, R. Varga, A. Tkacik, and K. Gray, "Opendaylight: Towards a Model-driven SDN Aontroller Architecture," in *Proceedings of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2014.

[42] Microsoft, "Microsoft 365 Denial-of-Service Defense Strategy," https://docs.microsoft.com/en-us/office365/enterprise/office-365-microsoft-dos-defense-strategy.

[43] A. Morton, "IMIX Genome: Specification of Variable Packet Sizes for Additional Testing," RFC 6985, IETF, 2013.

[44] M. Mousa, A. M. Bahaa-Eldin, and M. A. Sobh, "Autonomic management of mpls backbone networks using sdns," in *Proceedings of the International Conference on Computer Engineering and Systems (ICCES)*, 2017.

[45] L. Obregon, "Infrastructure Security Architecture for Effective Security Monitoring," *SANS Institute*, vol. 2, 2015.

[46] OpenXPKI, https://github.com/openxpki/openxpki/.

[47] A. Panda, C. Scott, A. Ghodsi, T. Koponen, and S. Shenker, "Cap for Networks," in *Proceedings of the ACM Workshop on Hot Topics in Software Defined Networking (HotSDN)*, 2013.

[48] R. Perlman, D. E. 3rd, D. Dutt, S. Gai, and A. Ghanwani, "Routing Bridges (RBridges): Base Protocol Specification," RFC 6325, IETF, 2011.

[49] A. Perrig, P. Szalachowski, R. M. Reischuk, and L. Chuat, *SCION: A Secure Internet Architecture*. Springer Verlag, 2017.

[50] K. Phemius, M. Bouet, and J. Leguay, "Disco: Distributed Multi-domain SDN Controllers," in *Proceedings of the IEEE Network Operations and Management Symposium (NOMS)*, 2014.

[51] PISKES Implementation, https://github.com/netsec-ethz/scion/tree/scionlab/go/lib/drkey.

[52] Prototype Implementation, https://github.com/chaehni/scion/tree/zoning/go/sig.

[53] Y. Qi, B. Yang, B. Xu, and J. Li, "Towards System-level Optimization for High Performance Unified Threat Management," in *Proceedings of the International Conference on Networking and Services (ICNS)*, 2007.

[54] H. V. Ramasamy, C.-L. Tsao, B. Pfitzmann, N. Joukov, and J. W. Murray, "Towards Automated Identification of Security Zone Classification in Enterprise Networks," in *Hot-ICE*, 2011.

[55] E. Rescorla, "The transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, IETF, 2018.

[56] B. Rothenberger, D. Roos, M. Legner, and A. Perrig, "PISKES: Pragmatic Internet-Scale Key-Establishment System," in *Proceedings of the ACM Asia Conference on Computer and Communications Security (ASIACCS)*, 2020.

[57] S. Gueron, *Intel® Advanced Encryption Standard (AES) New Instructions Set*. White Paper, 2012.

[58] X. Shi, H. Lin, H. Jin, B. B. Zhou, Z. Yin, S. Di, and S. Wu, "GIRAFFE: A Scalable Distributed Coordination Service for Large-scale Systems," in *Proceedings of IEEE International Conference on Cluster Computing (CLUSTER)*, 2014.

[59] SIG (Scion-IP Gateway), https://github.com/scionproto/scion/tree/master/go/posix-gateway.

[60] Spirent, *IMIX (Internet Mix) Journal*. Test Methodology Journal, 2006.

[61] SQLite 3.32.0, https://www.sqlite.org/releaselog/3_32_0.html.

[62] Statista, "Leading banks in the United States in 2019, by number of branches," https://www.statista.com/statistics/935643/banks-with-the-most-branches-usa/.

[63] X. Sun, Y. Sung, S. D. Krothapalli, and S. G. Rao, "A systematic approach for evolving vlan designs," in *Proceedings of the Annual IEEE Conference on Computer Communications (INFOCOM)*, 2010.

[64] Y. Sung, S. G. Rao, G. G. Xie, and D. A. Maltz, "Towards systematic design of enterprise networks," in *Proceedings of the ACM CoNEXT*, 2008.

[65] D. Thaler and C. Hopps, "Multipath Issues in Unicast and Multicast Next-Hop Selection," RFC 2991 (Informational), IETF, 2000.

[66] B. Trammell, J. Smith, and A. Perrig, "Adding Path Awareness to the Internet Architecture," *IEEE Internet Computing*, vol. 22, no. 2, 2018.

[67] B. Tschaen, Y. Zhang, T. Benson, S. Banerjee, J. Lee, and J.-M. Kang, "SFC-Checker: Checking the Correct Forwarding Behavior of Service Function Chaining," in *Proceedings of IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2016.

[68] K. Yap, M. Motiwala, J. Rahe, S. Padgett, M. Holliman, G. Baldus, M. Hines, T. Kim, A. Narayanan, A. Jain *et al.*, "Taking the Edge off with Espresso: Scale, Reliability and Programmability for Flobal Internet Peering," in *Proceedings of the ACM SIGCOMM*, 2017.

[69] T. Yu, S. K. Fayaz, M. P. Collins, V. Sekar, and S. Seshan, "PSI: Precise Security Instrumentation for Enterprise Networks," in *Proceedings of the Symposium on Network and Distributed System Security (NDSS)*, 2017.

[70] Y. Yuan, S. Moon, S. Uppal, L. Jia, and V. Sekar, "NetSMC: A Custom Symbolic Model Checker for Stateful Network Verification," in *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2020.

[71] T. Zhang, P. Giaccone, A. Bianco, and S. De Domenico, "The Role of the Inter-controller Consensus in the Placement of Distributed SDN Controllers," *Computer Communications*, vol. 113, pp. 1–13, 2017.

APPENDIX A
PACKET METADATA

```
type Packet struct {
        Ingress         bool
        SrcHost         net.IP
        DstHost         net.IP
        RemoteTP        string
        DstZone         uint32
        RawPacket       common.RawBytes
}
```

The packet metadata is the abstract object passed between modules, describing an IP packet. It accumulates information about the raw IP packet (`RawPacket`). The `Ingress` field identifies a packet as either an ingress packet, coming from the WAN, or an egress packet that originated in the local network. `SrcHost` and `DstHost` reflect the source and destination IP addresses of the packet. `RemoteTP` designates the remote TP. For an ingress packet that is the source TP from which the packet was received, for an egress packet it is the TP to which the packet needs to be forwarded to. `DstZone` is the Zone ID of the zone to which `DstHost` belongs.

APPENDIX B
CONTROLLER DATABASE

```
CREATE TABLE Zones(
id INTEGER NOT NULL,
name TEXT,
PRIMARY KEY(id)
);

CREATE TABLE Sites(
tp_address TEXT NOT NULL,
name TEXT,
PRIMARY KEY(tp_address)
);

CREATE TABLE Subnets(
net_ip BLOB NOT NULL,
net_mask BLOB NOT NULL,
zone INTEGER NOT NULL,
tp_address TEXT NOT NULL,
PRIMARY KEY (net_ip, net_mask),
FOREIGN KEY (zone) REFERENCES Zones(id) ON DELETE CASCADE,
FOREIGN KEY (tp_address) REFERENCES Sites(tp_address) ON DELETE CASCADE
);

CREATE TABLE Transitions(
src INTEGER NOT NULL,
dest INTEGER NOT NULL,
PRIMARY KEY (src, dest) ON CONFLICT REPLACE,
FOREIGN KEY (src) REFERENCES Zones(id) ON DELETE CASCADE,
FOREIGN KEY (dest) REFERENCES Zones(id) ON DELETE CASCADE
)
```

The controller database consists of 4 tables: `Zones`, `Sites`, `Subnets`, and `Transitions`. The `Zones` table contains all network zones known to the controller, identified by zone IDs. Additionally, a human readable description is attached. The `Sites` table holds all known branch sites with the addresses of the corresponding TPs and a textual description. The `Subnets` table describes the configured IP subnets together with their zone membership and the TP behind which they are located. Finally, the `Transitions` table reflects the zone transition matrix of allowed zone transitions.