KDC Placement Problem in Secure VPLS Networks

Mohammad Borhani
IDA
Linköping University
Linköping, Sweden
0000-0002-3570-3297

Ioannis Avgouleas
IDA
Linköping University
Linköping, Sweden
0000-0001-8960-0544

Madhusanka Liyanage
School of Computer Science
University College Dublin
Dublin, Ireland
0000-0003-4786-030X

Andrei Gurtov
IDA
Linköping University
Linköping, Sweden
0000-0002-9829-9287

***Abstract*—Virtual Private LAN Service (VPLS) is a VPN technology that connects remote client sites with provider networks in a transparent manner. Session key-based HIPLS (S-HIPLS) is a VPLS architecture based on the Host Identity Protocol (HIP) that provides a secure VPLS architecture using a Key Distribution Center (KDC) to implement security mechanisms such as authentication, encryption etc. It exhibits limited scalability though. Using multiple distributed KDCs would offer numerous advantages including reduced workload per KDC, distributed key storage, and improved scalability, while simultaneously eliminating the single point of failure of S-HIPLS. It would also come with the need for optimally placing KDCs in the provider network. In this work, we formulate the KDC placement (KDCP) problem for a secure VPLS network as an Integer Linear Programming (ILP) problem. The latter is NP-hard, thereby suggesting a high computational cost for obtaining exact solutions especially for large deployments. Therefore, we motivate the use of a primal-dual algorithm to efficiently produce near-optimal solutions. Extensive evaluations on large-scale network topologies, such as the random Internet graph, demonstrate our method's time-efficiency as well as its improved scalability and usefulness compared to both HIPLS and S-HIPLS.**

***Index Terms*—VPLS, VPN, HIP, Security, Industrial Internet**

I. Introduction

The industrial world has taken notice of the widespread deployment of the Industrial Internet of Things (IIoT). Cyber-physical systems (CPS) can be used in smart industries to improve the monitoring and control abilities of physical systems using modern information communication technologies (ICT) (e.g., manufacturing equipment and assembly lines) [1]–[3]. The backbone of industrial systems is comprised of a large number of low-cost sensors as well as smart devices.

By 2030, the IIoT may be valued at 7.1 trillion dollars in the United States and over 1.2 trillion dollars in Europe. Poorly secured design in IIoT environments paves the way for malware to launch destructive cyber-attacks, such as the Mirai attack in 2016, in which hundreds of thousands of connected devices were infected. Moreover, Consumer IoT isn't the only target of threats. In reality, industrial environments have been targeted in the past, with catastrophic results (e.g., Industroyer and Stuxnet). As a result, it is clear that IIoT would never be able to reach its full potential without security [4]–[6].

The evolution of communication networks has paved the path towards advanced networking concepts. Among these technologies, the virtual or virtual private network (VPN) concept is considered one such early advancements in communication technologies. The origins of VPNs can be traced back to 1996, when Microsoft employees deployed a point-to-point tunneling protocol over their intranet. Since then, VPNs have become popular and widely used in many use cases, such as allowing remote workers to access internal networks, connecting remote branches with head offices, and logically partitioning internal networks. In general, VPNs offer benefits such as bypassing restrictions enforced by Internet Service Providers (ISPs) and governments, eliminating geographic restrictions, protecting users and user traffic from snooping type attacks, protecting anonymity by hiding the user's actual location, and encrypting all user traffic [7].

Among the various VPN solution, VPLS (Virtual Private LAN Service) is one of the widely used types of VPN. VPLS is a Layer 2 provider-provisioned VPN, and it can provide Ethernet multi-point to multi-point connectivity over an IP or Multiprotocol Label Switching (MPLS) network. Therefore, VPLS allows the Local Area Networks (LAN) to extend across multiple customer sites and appear as a single Ethernet LAN network [8]. Since VPLS shares a single broadcast domain, it offers several benefits, such as low communication latency, low implementation and maintenance cost, the ability to handle legacy protocols in addition to IP, and the ability to build secure and homogeneous networks rapidly. The popularity of VPLS is further supported by the fact that technology giants such as Cisco, Juniper, Samsung, Nokia, and Vodafone are working and conducting training on VPLS and related technologies [9]–[11]. It is estimated that the global market share of VPLS will reach *$2,420 million by 2025*, growing at a *Compound Annual Growth Rate (CAGR) of 19.5%* between 2020 and 2025 [12].

In addition, VPLS has gained immense popularity as a viable VPN solution for securely and transparently interconnecting multiple client LANs in industrial environments and novel CPSs on the Industrial Internet. VPLS is an especially ideal solution for interconnecting SCADA (Supervisory Control and Data Acquisition) industrial control system devices, which sometimes do not support higher layer protocols such as IP. VPLS's salient features such as protocol independence, multi-point to multi-point mesh connectivity, robust security, low operational cost are essential to supporting future CPS applications.

This research was supported by the Excellence Center at Linköping – Lund in Information Technology.

Moreover, the popularity of CPSs brings requirements such as high security, enhanced scalability, and optimal utilization of network resources. Specifically, the increasing number of connected devices and the introduction of new CPS application services will demand improved security. VPLS solutions should accommodate secure connectivity for this expected traffic growth, which is imminent for future CPSs.

Several secure VPLS solutions have been proposed to improve the security and scalability of existing VPLS networks. The first secure VPLS architecture was proposed as Host Identity protocol enabled VPLS, or HIP-based virtual private LAN service (HIPLS) [13]. HIPLS-based VPLS networks are used in many industrial plants, including Boeing's 777 airplane assembly plant [14]. In addition, major SCADA network appliance manufacturing companies are working on HIPLS-based appliances, such as Tempered in USA [15].

Several research works have been proposed to improve the features of HIPLS such as efficient key management [16], [17], scalability [16]–[19] and tunnel establishment procedure [20], [21]. In [16], [17], the authors propose a Session key-based HIPLS (S-HIPLS) architecture which has been used as the basis for other improved HIPLS versions, i.e. [18]–[21]. In S-HIPLS, the authors propose two session keys called the Content Encryption Key (CEK) and the Key Encryption Key (KEK) to improve the efficiency of key management and to improve scalability. These session keys are distributed by a Key Distribution Center (KDC), the placement of which is critical to archive optimal performance in S-HIPLS. However, the authors fail to propose a cost-effective strategy for KDC placement for S-HIPLS in their proposal.

The distributed placement of the KDC in a secure VPLS network is analogous to the placement of the controller in Software Defined Networks (SDNs) [22], which has been widely investigated. The authors of [23] developed a distributed control platform for large-scale networks and concluded that this paradigm brings an abstraction for network-wide management. The authors of [24] originally identified the controller placement problem, with a goal of reducing overall network latency.

Recent advances in the controller placement problem (CPP) motivated the authors in [25] to write a survey paper on the CPP approaches in SDN. They reviewed the current state of the controller placement problem from an optimization perspective and concluded that the location of controllers has a direct impact on network performance. The authors of [26] argued that the CPP and its challenges have not been thoroughly examined. Thus, in SDN use cases, they provide a thorough overview of numerous algorithmic challenges in controller placement problems.

Network configurations in which the data must transit via the hypervisor to reach the corresponding controller are presented in [27]. As a result, the latency of networking devices is determined by the locations of both hypervisors and controllers. The authors used a solver to obtain the optimal solution of the proposed formulation. However, obtaining the optimal solution in a large-scale scenario may require extensive computational power, and solving the problem optimally may require the solver to run for hours or days.

A. Contributions

To the best of our knowledge, this is the first study on the optimal placement of KDCs in secure VPLS networks. Since the majority of secure VPLS networks that employ public-key mechanisms use KDC nodes as entities for exchanging the necessary security keys, as well as for serving other security-related features, this work can be regarded as an original model for designing minimal cost VPLS networks from a provider network perspective. The main contributions of this paper are:

- We first motivate the study of the KDC placement (KDCP) problem in a secure VPLS network by demonstrating how network operating expenses behave in large-scale network deployments. We formulate the KDCP problem by considering the KDCs' activation costs and the cost of providing secure communications for each VPN, which accounts for exchanging the necessary security keys between KDCs and PEs.
- We formulate the KDCP problem as an integer linear program (ILP) problem and use an exhaustive search to determine the optimal placement of KDCs inside the provider network. The exact solution of the latter is used as a benchmark for the performance of our proposed algorithm.
- We develop a primal-dual approximation algorithm that offers a polynomial-time algorithmic solution to the KDCP problem since the ILP problem is an NP-hard optimization problem i.e., unless P=NP, a very difficult problem to efficiently solve for large-scale network topologies. Besides being fast in practice, the proposed solution offers a guaranteed approximation factor to the KDCP.
- We extensively evaluate the performance of our approximation algorithm using several large-scale network topologies, e.g., the Random Internet graph, Erdős–Rényi graph, etc. Our results validate that our algorithm considerably outperforms exact solutions in terms of time-efficiency. Fast solutions to the KDCP problem offer network providers the ability to recalculate the KDC placement on-the-fly in case of addition/deletion of PE routers .
- Finally, we demonstrate that our proposed architecture scales better than S-HIPLS and HIPLS with regard to keys storage at KDCs in real-world scenarios.

B. Article Organization

The rest of the paper is organized as follows. Section II considers the VPLS, S-HIPLS, and components of the VPLS networks. An overview of the problem in the S-HIPLS network, as well as a real-world scenario, are given in Section III. The mathematical problem formulation and the proposed algorithm are considered in Section IV. Section V contains the simulation result and discussion. Finally, Section VI concludes the paper.

II. Background

This section provides background information about VPLS and S-HIPLS.

A. Virtual Private LAN Service (VPLS)

VPLS is a Layer 2 provider-provisioned VPN type that can provide multi-point to multi-point connectivity between remote customer sites over a provider network.

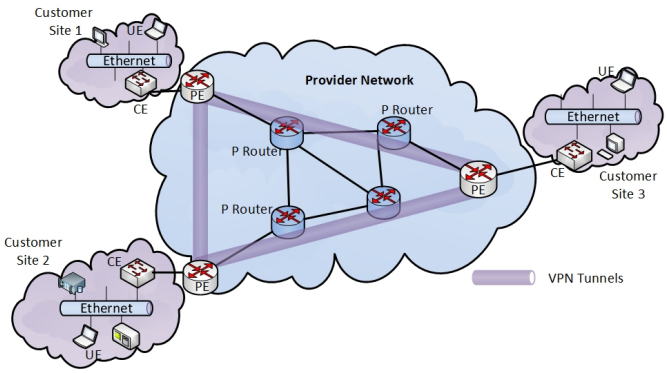

Fig. 1. The VPLS Architecture

A typical VPLS network contains several key components, as depicted in Fig. 1, such as

- **Customer Network:** This is the user of the VPLS network. It consists of several sites that are geographically distributed and fully controlled by the user.
- **Provider Network:** This is the core/underlay network of the VPLS, which facilitates VPN tunnels or Pseudowires (PWs). Provider networks generally operate as Layer 3 networks using common network protocols such as MPLS or IP. E.g., Internet, Telecommunication Network, Wi-Fi Network.
- **Customer Edge Equipment (CE):** CEs are the intermediate devices that interconnect the customer and provider networks. CEs typically belong to the customer and reside on the customer's premises. A CE can be connected to a single or multiple Provider Edge Equipment (PE).
- **Provider Edge Equipment (PE):** PEs are the gateways for customer network traffic, and they reside at the edge of the provider network. PEs also hold complete knowledge about the VPLS network. Pseudowires (PWs) are built between PEs and these PEs owned by the service provider.
- **VPN Tunnels/PWs:** A VPN tunnel/PWs is an encrypted or encapsulated link between two PEs. VPLS generally utilizes protocols such as MPLS and IPsec for encapsulation/encryption.
- **Provider Router (P Router):** This is a transit router belonging to the provider network. A P router is typically connected to one or more PE Routers. It is not aware of the VPLS and is mainly responsible for traffic routing within the provider network.

B. Session key based HIPLS (S-HIPLS)

Early VPLS architecture does not support security. HIPLS is the first secure VPLS architecture, and it has introduced a logical security plane for management of the security services related to VPLS. However, the scalability of the HIPLS security plane is limited due to the complexity of key storage. In HIPLS, every PE has to store $O(m)$ keys, where m is the number of PEs in the VPLS network. To overcome this issue, Session-key-based HIPLS (S-HIPLS) was introduced [17]. S-HIPLS overcomes the scalability issue of HIPLS by proposing a novel session key-based security mechanism instead of per-tunnel key management procedure of HIPLS. In S-HIPLS, the number of keys stored on PEs is reduced from $O(m)$ to $O(v+1)$, where v is the number of VPN types and m is the number of PEs in the VPLS network. Thus, the number of keys stored on each PE is independent of the number of PEs in S-HIPLS based VPLS networks. In addition, S-HIPLS proposes an efficient broadcast mechanism to increase the forwarding plane scalability of HIPLS networks.

Two types of session keys are used in S-HIPLS:

- **Content encryption key (CEK)**: This is a unique symmetric key for each VPN that is used to encrypt and decrypt all user traffic in a single VPN network.
- **Key Encryption Key (KEK)**: This is a unique symmetric key for each PE that is used to encrypt and decrypt the corresponding CEKs.

S-HIPLS proposes the use of a Key Distribution Center (KDC) to manage the key distribution process. Each PE shares a unique KEK with a KDC, and each KDC periodically distributes the CEK to PEs encrypted with unique KEKs. In addition, the KDC is responsible for PE subscription, PE life-cycle management, and PE authentication. For large-scale networks, S-HIPLS should utilize a distributed KDC structure to obtain a balanced workload, reduce the complexity of key distribution, and avoid single failure points.

III. Overview

A. KDC placement in Secure VPLS

The KDC and each PE share a unique symmetric key, which is utilized as the KEK for that PE. The CEK distribution is under the control of the KDC. It sends the CEK to PEs on a regular basis. The KEK of each PE is used to encrypt these CEKs. As a result, an eavesdropper will be unable to retrieve the CEK. To refresh the CEK/KEK, each PE and KDC run HIP BEX instances on a constant schedule (i.e., every 10 seconds) [28]. The HIP architecture implements this operation, which is known as rekeying. Thus, to protect the VPLS communication session, each KDC produces fresh CEKs. Even if an intruder manages to capture a CEK, it will be invalid after the rekeying timeout.

Secure VPLS is normally used in large-scale networks (e.g., ISP and WAN networks, Long Term Evolution (LTE) backhauls, etc.). Thus, the VPLS provider networks' performance has a direct impact on the customer's active VPN communications. Furthermore, some emerging industry 4.0

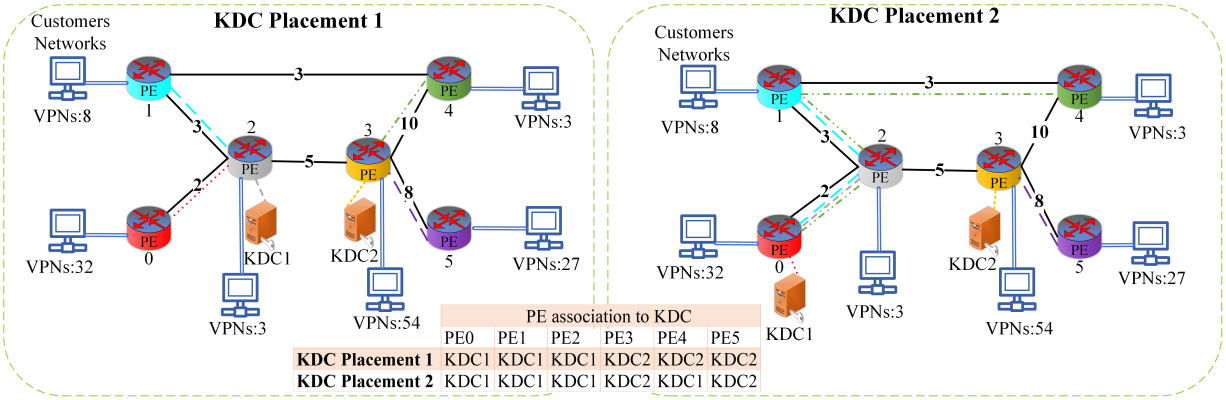

Fig. 2. Example of cost-effective KDC placement in VPLS: The left figure depicts placing the KDCs on the PEs with the highest connectivity degrees, which is not optimal in terms of cost. The right figure illustrates the optimal placing of KDCs with minimal cost. Path to KDC is also shown for each PE.

network use cases may be extremely reliant on the provider network's latency.

A single KDC will obviously not be able to meet the demands of supporting the security requirements of each PE in a large network with hundreds or thousands of PEs. As discussed in [17], a distributed KDC design has numerous advantages, including reduced workload, dispersed key storage complexity, and no single point of failure. However, in a distributed KDC design in a secure VPLS network, placement of the KDCs in the underlying network can drastically affect network performance. As a result, optimal KDC placement can considerably reduce the measurement overhead in the control plane.

We present an example in Fig. 2 to illustrate the KDC placement problem in a secure VPLS network. Fig. 2 represents a simple network topology with six PEs along with the routing path and the link cost. As VPLS networks are commonly used in WAN networks, the link cost can be defined as the price of using the links for each PE. Other metrics such as link delay also could be considered as link cost. Each PE serves VPN communication for many customers. The number of customers is marked in the figure. The complexity of the rekeying mechanism is highly dependent on the number of VPNs supported by each PE.

Let's start with the assumption that the best location to place two KDCs in this network is on nodes with high degrees of connectivity (i.e., PE2 and PE3). We link each PE to its directly connected KDC since each PE has to connect to a KDC to ensure network security features. The total network cost can be calculated as the sum of all communication costs between KDCs and PEs plus the cost of placing KDCs in specific locations on the provider network. The network administrator can manually modify this cost. Furthermore, in a secure VPLS network, communication costs are directly related to the number supported VPNs by PE. Hence, it can be stated that `Communication Cost = Number of VPNs × Cost of connecting PE to associated KDC`.

Assuming that installing a KDC on each location in this example costs 50, the total network cost of this deployment is $(32\times2+8\times3+0+0+3\times10+27\times8)+(50+50)=434$. This assignment scenario is depicted in KDC placement 1 in Fig. 2. The connection of PEs with KDCs is depicted using dashed lines. Although the logic behind this assignment appears to be reasonable, its network cost is higher than the optimal assignment, which has a cost of 386 and is shown in KDC placement 2 in the figure.

For the sake of demonstration, we began with a very low number of PEs and supported VPNs. In a real-world scenario, however, a network topology may include hundreds or thousands of PEs, each serving the needs of hundreds of customers. Distributed KDC placement is essential in networks with massive amounts of nodes. By proposing an algorithm to place KDCs optimally, the aim for a given secure VPLS network is to minimize the total provider network cost. This algorithm should achieve three key objectives:

- Optimal network cost: the solution generated by the algorithm should impose a minimum network cost that is adaptable to a wide range of network topologies. It also needs to report the guarantee of optimality.
- Fast execution: due to some critical use cases of VPLS, the algorithm should be executed in a way that does not impose any additional delay on the network.
- Abstraction: The algorithm should be transparent to upper-layer applications. Also, the execution of the algorithm should not require any changes to customers' networks or VPLS configurations.

B. Other Applications

Applications of our formulation, which we discuss shortly, are not restricted to the primary area discussed in this paper. It can be applied to various domains as the service placement problem (SPP). For the sake of brevity, we will outline some of them as follows:

- Software-defined networking (SDN): Due to limited computational power and the geographical dispersion of the switches, a single controller cannot handle thousands of flows in real-world SDN implementations. The placement

of controllers influences the network's performance and its ability to react to network events [29]. By changing the KDC to SDN controller, PEs to SDN-enabled switches, and the number of VPNs to the number of flows, Fig. 2 can represent the controller placement problem in SDN.

- Wireless Sensor Networks (WSN): Data acquisition and transmission are the primary roles of resource-constrained nodes in a WSN. In addition, the location of the base stations (BS) and the association of sensors to base stations significantly impact the network's lifespan. Sending data to a base station far from the source sensor increases latency, energy consumption, and packet loss rate [30]. To maximize the network lifetime and reduce the cost of base station deployment, the base station placement problem can be represented in Fig. 2 by substituting KDCs for base stations, PEs for sensors, the weight of the links for the distance between nodes, and the number of VPNs for the amount of data to send and receive between a sensor and its base station.
- Content Delivery Network (CDN): Replica server placement is a key design problem for content delivery networks (CDNs), in which large distributed infrastructures of replica servers are deployed in predetermined locations. It seeks to minimize deployment costs and assure end-user QoS satisfaction [31] (e.g., content delivered to end-users with reduced latency). In Fig. 2, let a replica server replace the KDC. Moreover, end users need to connect to a replica server (instead of PE connecting to a KDC). By setting the deployment cost of replica servers on each location and removing the number of VPN parameters, solutions to the KDC problem can be used in replica server replacement in CDN.

IV. The KDCP Problem

In this section, we formulate the KDC placement problem as a linear integer programming one. First, we consider an algorithm for determining the optimal placement using an exhaustive search for all possible combinations of KDCs inside the provider network. Secondly, we present an approximation algorithm to efficiently acquire a near-optimal solution due to the high computation cost of finding the exact solution for networks with high numbers of PEs.

A. Problem Formulation

The provider network is modeled as an undirected graph $G = (V,E)$, where $V = \{PE_1, PE_2, ..., PE_m\}$ is the set of PEs in the network, and the set of links connecting the PEs is denoted by E. V also denotes the candidate locations for placing a KDC. Let x_{ij} denote a binary variable indicating whether the j-th PE is connected to the i-th KDC for $i = 1, ..., n$ and $j = 1, 2, ..., m$. Additionally, $y_i = 1$ denotes that KDC i is open for serving requests. The cost of serving PE_j by KDC_i is denoted by d_{ij} and w_i denotes the opening cost of KDC_i. The problem of KDC Placement in a Provider network (KDCP) can be formulated as follows:

Table I
Summary of notations

Notation	Description
$G = (V,E)$	undirected graph where V is the set of PEs and E is the set of links
V	candidate locations of placement of KDCs
n	number of KDCs in a network
m	number of PEs in a network
v	number of VPNs in a network
x_{ij}	variable indicating whether the j-th PE is connected to the i-th KDC
y_i	variable indicating KDC i is open
d_{ij}	cost of serving PE_j by KDC_i
w_i	opening cost of KDC_i
u_j	dual variable associated with constraint (2b)
v_{ij}	dual variable associated with constraint (2c)
(u^*, v^*)	maximal dual solution
$N(j)$	set of neighbors of PEj
$j \in adj(i)$	PE j is directly connected to KDCi
i_{cl}	KDC center of a cluster
d_{VPLS}	VPLS network diameter

$$(P) \text{ minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{m} d_{ij}x_{ij} + \sum_{i=1}^{n} w_i y_i \tag{1a}$$

$$\text{subject to} \quad \sum_{i=1}^{n} x_{ij} \geq 1,\ j = 1,\ldots,m \tag{1b}$$

$$x_{ij} \leq y_i,\ j = 1,\ldots,m \text{ and } i = 1,\ldots,n \tag{1c}$$

$$x_{ij} \in \{0,1\},\ j = 1,\ldots,m \text{ and } i = 1,\ldots,n \tag{1d}$$

$$y_i \in \{0,1\},\ i = 1,\ldots,n \tag{1e}$$

The term $\sum_{i=1}^{n}\sum_{j=1}^{m} d_{ij}x_{ij}$ in the objective function calculates the communication cost between PE_j and KDC_i, and is called *service cost*; the second term i.e., $\sum_{i=1}^{n} w_i y_i$, captures the cost of activating specific KDCs from V, and evaluates the *KDC activation cost*. Constraint (1b) states that each PE is assigned to at least one KDC. The restriction of PEs assignments to only active KDCs is represented by constraint (1c). Finally, constraint (1d) and (1e) define the domain of the binary decision variables. The notations used in KDCP are listed in Table I.

B. Algorithms for solving KDCP

Optimal Solution: Problem formulation of (P) is a variation of the Facility Location Problem (FLP), which has been shown to be NP-hard [32]. The process of proving the NP-hardness of the problem is constructed by reduction from the set cover problem. First, we acquire an exact solution for KDCP using Gurobi solver. The KDCP's optimal solution can be found by searching exhaustively for all feasible KDCs placement combinations. Gurobi [33] uses exact algorithms to solve the KDCP. Exact algorithms are guaranteed to discover the best solution but may need practically prohibitive amounts of time to provide an optimal solution.

Although KDCP is very simple in formulation, it is a very difficult problem to efficiently solve (unless P = NP) since it is an NP-hard optimization problem. Therefore, its exact solution is prohibitively time-consuming [34]. Consequently, we present a primal-dual approximation schema [35], since it yields algorithms with good running times in practice, as opposed to methods based on branch-and-bound that are, in general, more computationally expensive and more complex to implement. Our intention is to develop a polynomial-time solution to the KDCP with a guaranteed approximation factor. To do so, we substitute a simpler optimization problem with an optimal value at least as small as (P) for a hard problem formulated as KDCP. The Linear Programming relaxation (LP-relaxation) of KDCP is:

$$(P^{LP}) \text{ minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{m} d_{ij}x_{ij} + \sum_{i=1}^{n} w_i y_i \tag{2a}$$

$$\text{subject to} \quad \sum_{i=1}^{n} x_{ij} \geq 1,\ j = 1,\ldots,m \tag{2b}$$

$$x_{ij} \leq y_i,\ j = 1,\ldots,m \text{ and } i = 1,\ldots,n \tag{2c}$$

$$x_{ij} \geq 0,\ j = 1,\ldots,m \text{ and } i = 1,\ldots,n \tag{2d}$$

$$y_i \geq 0,\ i = 1,\ldots,n \tag{2e}$$

We should note that as P^{LP} is relaxation of KDCP problem, then $P^{LP} \leq P$.

C. Approximation Algorithm for KDCP Problem

We previously defined an integer formulation for the KDCP problem, and devised the linear programming relaxation. The dual program of (P^{LP}) associates two sets of decision variables with these two constraints, i.e., dual variables u_j and v_{ij} are associated with constraints (2b) and (2c), respectively and is given by:

$$(D) \text{ maximize} \quad \sum_{j=1}^{m} u_j \tag{3a}$$

$$\text{subject to} \quad u_j - v_{ij} \leq d_{ij},\ j = 1,\ldots,m \text{ and } i = 1,\ldots,n \tag{3b}$$

$$\sum_{j=1}^{m} v_{ij} \leq w_i,\ i = 1,\ldots,n \tag{3c}$$

$$u_j \geq 0,\ j = 1,\ldots,m \tag{3d}$$

$$v_{ij} \geq 0,\ j = 1,\ldots,m \text{ and } i = 1,\ldots,n \tag{3e}$$

Let's consider the dual program. By setting all KDC activation costs to zero, we obtain a trivial lower bound for the KDCP. If we set $w_i = 0$ for each KDC i, then the solution is to activate all KDCs and allocate each PE to the closeset KDC. Also, by setting $u_j = min_i(d_{ij})$, the lower bound is $\sum_{j=1}^{m} u_j$. Setting all $w_i = 0$ obtains a trivial lower bound. However, since placing each KDC on a network will incur extra cost for the network provider, we need to account for nonzero w_i costs into consideration as well.

Let's assume that the activation cost of each KDC (i.e., w_i) is divided into v_{ij} shares and distributed to the PEs such that $\sum_{j=1}^{m} v_{ij} = w_i$. Hence, v_{ij} can be seen as the price that each PE needs to pay to use a secure VPLS network—activating a KDC on the provider network incurs costs for service providers. Each node using the secure network requires to pay for the secure transmission of the data.

The j-th PE is only required to pay this charge if it makes use of the KDC i. Moreover, each PE j is willing to pay the lower price for its connection cost to the KDC and a share of the KDC activation cost across all KDCs i.e., $u_j = min_i(d_{ij} + v_{ij})$. A PE in the provider network may need to traverse multiple links with different delay parameters to connect to an associated KDC. Moreover, since d_{ij} encompasses the cost of transmission CEKs of a PE on the path connecting the PE to its associated KDC, each PE seeks to minimize the latency of exchanging the (CEK/KEK) keys and to pay its share of building a secure network. Likewise, $\sum_{j=1}^{m} u_j$ yields a lower bound for the problem. Furthermore, by setting $(u_j - v_{ij} \leq d_{ij})$ and $max(\sum_{j=1}^{m} u_j)$, we obtain the maximum value of the lower bound. We should elaborate that the dual constraint $u_j - v_{ij} \leq d_{ij}$ considers the variable x_{ij} in the primal, while the dual constraint $\sum_{j=1}^{m} v_{ij} \leq w_i$ corresponds to the y_i.

Definition 1 We define (u^*, v^*) as the maximal dual solution when no u_j^* can be raised by a positive value so as to construct a set of v_{ij}^* that leads to a dual feasible solution.

Definition 2 Given (u^*, v^*), we denote the set of neighbors of PE j by $N(j)$ in which, if $N(j)$ includes KDC i, i.e., KDC i neighbors PE j, then $u_j^* \geq d_{ij}$.

Algorithm 1 is proposed to solve the KDCP problem using the primal-dual method. The primal and dual program will not be exactly solved by Algorithm 1. Instead, we begin with zero solutions of dual program and increase them until we obtain a feasible primal solution. More specifically, by increasing the dual variable u_j (see (3a)), we seek to obtain as large a dual solution as possible. We uniformly increase all u_j, which we seek to maximize, in each time step of running the algorithm until they become blocked (blocked variables are those variables in the dual that we can not increase anymore). The notion of blocked variables paves the way for generating the maximal dual solution.

Definition 3 We denote that KDC i is *blocked* if $\sum_{j=1}^{k} v_{ij} = w_i$.

Definition 4 We denote that PE j is *blocked* if PE j is a neighbor of a blocked KDC i. By Definition 2, PE j is blocked if $u_j \geq d_{ij}$ for some blocked KDC i.

Furthermore, if we witness $u_j = d_{ij}$ for some KDC i, then we start increasing v_{ij} in parallel to u_j. By considering constraint (3b) of the dual program, we can keep raising u_j. As a result, u_j will keep rising, and v_{ij} will increase with it as well, unless some blocking condition of $\sum_{j=1}^{m} v_{ij} = w_i$ (see 3c)

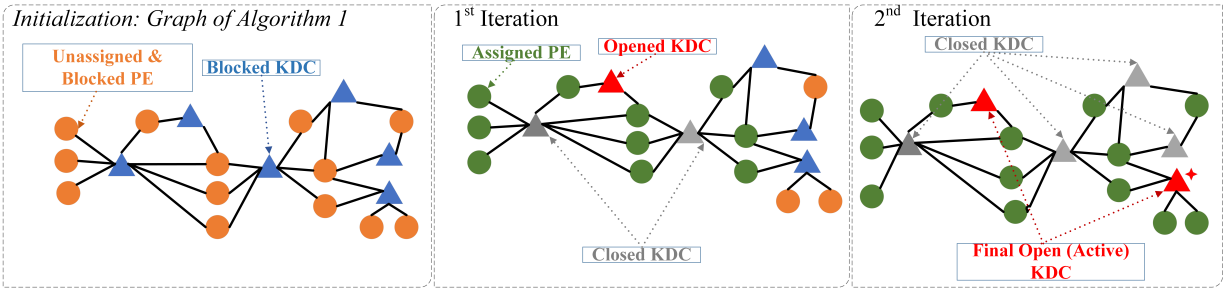

Fig. 3. Initialization and two iterations of Algorithm 1

stops it. Moreover, at this point, v_{ij} is blocked, which results in the blocking of u_j (see constraint (3b)). Lines 1-8 execute the first part of the algorithm after which we cannot anymore raise u_j. However, we can now derive a set of v_{ij} that leads to a dual feasible solution.

Algorithm 1: Primal-Dual Algorithm for KDCP

```
1  u ← 0, v ← 0, T ← 0 ;
2  while (All u_j is not in block state) do in parallel
3      if (u_j ≠ blocked) and (v_ij ≠ blocked) then
4          Increase u_j by each time step T;
5          Increase v_ij such that u_j ≥ d_ij for some u_j;
6          T = T + 1
7      end
8  end
   /* End of Part 1 - Start of Part 2  */
   /* KDCs are neither open nor closed.
      PEs are still not assigned       */
9  S = set of blocked KDCs part 1;
10 while (unassigned PEs ≠ 0) do
11     i_cl = choose the first blocked KDC from set S;
12     i_cl.open();
13     C = set of blocked KDCs within distance 2 of i_cl
         in the graph of the algorithm;
14     i_cl.close(C);
15     i_cl.assign(any unassigned PE within distance 3 of
         the i_cl in the graph of the algorithm);
16     S = S − {i_cl ∪ KDC ∈ C};
17 end
```

Lemma 1 *The execution of the first part of Algorithm 1 yields the dual solution* (u^*, v^*) *which associates every PE to some KDC set* $M = \{i \in KDC : w_i = \sum_{j=1}^{m} v_{ij}^*\}$.

Proof. We need to investigate the properties of the maximal dual solution in this proof. As discussed earlier, each PE j is willing to construct $u_j^* = min_i(v_{ij}^* + d_{ij})$ in the maximal solution. Moreover, we have claimed that there exists some KDC in set M if the solution is maximal. By Definition 2, the designated KDC i will have $u_j^* \geq d_{ij}$. Let's consider the case that $u_j^* < min_i(v_{ij}^* + d_{ij})$. In this case, as we would be able to increase u_j^*, we did not obtain the maximal solution. Moreover, if $u_j^* = min_i(v_{ij}^* + d_{ij})$, and no KDC i exists in set M, then we are be able to increase u_j^* and v_{ij}^*, as $\sum_{k=1}^{m} v_{ik}^* < w_i$, and therefore we did not obtain the maximal solution. ■

In order to obtain the primal solution, we execute the part 2 of the primal-dual algorithm. For the sake of demonstration, we employ a graph of Algorithm 1 execution connecting KDCs and PEs. Fig. 3 depicts the graph in which the circles represent PEs, while triangles denote KDCs. We need to note that in this graph, there exists an edge between two vertices like PE j and KDC i if $u_j \geq d_{ij}$ (See Definition 2). Before the execution of part 2, all PEs are unassigned, i.e., PEs have yet to be connected to one or more KDCs, and in the blocked state.

In the first iteration of the second part of the algorithm, the first KDC that has been blocked in part 1 is opened. For instance, the KDC depicted in red in Fig. 3 (1st iteration) is the one that is opened first. Then, the algorithm shuts down all KDCs within distance 2 of the first opened KDC (line 14 in Algorithm 1). These closed KDCs are marked with grey in the figure (1st Iteration). The next step in Iteration 1 concerns the assignment of PEs to the recently opened KDC. All PEs that are adjacent to the opened KDC will be connected to it. Moreover, PEs that are within distance 3 of the opened KDC also connect to it (Line 15 in Algorithm 1).

The second blocked KDC is selected from the list of remaining blocked KDCs in the next iteration. There exist three blocked KDCs, as shown in the blue in the figure (1st Iteration), that require a decision to be opened or closed following the completion of Iteration one.

Let's assume that in Iteration 2, the KDC with a small star marker in the figure is chosen to be opened. Furthermore, the algorithm shuts down all KDCs within distance 2 of the newly opened KDC, likewise Iteration 1. The two KDCs above the newly opened one will be closed. Hence, the closed KDCs are illustrated in grey in the 2nd Iteration in the figure. Now, for the assignment phase (line 15), the same procedure of iteration 1 is executed until all PEs are assigned to KDCs. Finally, since no unassigned PEs or blocked KDC exist, the algorithm will finish executing.

Theorem 2 *Algorithm 1 is a 3-approximation algorithm for the KDCP problem.*

Proof. The analysis to prove the theorem concerns the cost of solution provided by our primal-dual algorithm for KDCP.

In Algorithm 1, the opened KDC i_{cl}, and its associated PEs can form a cluster (lines 11 to 15 in Algorithm 1). Moreover, the graph representing the execution of the algorithm can be partitioned into multiple clusters cl. The cost of solution can be written as $cost = \sum_{cl}(\sum_{j \in cl} d_{i_{cl}j} + w_{i_{cl}})$. The $cost$ is the sum of two terms; the total cost of opening KDC i_{cl} for each cluster, as well as the total cost of exchanging security keys between PE j and the KDC i_{cl} for each PE j in that cluster. Considering the fact that KDCs are blocked, the cost of activating KDC i_{cl} is derived as $w_{i_{cl}} = \sum_{j \in adj(i_{cl})} v_{i_{cl}}j$, in which PE $j \in adj(i_{cl})$ means that PE j is adjacent to KDC i_{cl} (the PE j is directly connected to KDC i_{cl} in the graph for Algorithm 1). Moreover, we can take the $\sum_{j \in cl} d_{i_{cl}j}$ and divide the sum into two parts. One concerns all the PEs that are adjacent to KDC i_{cl}, while the other part includes all the PEs that are not adjacent to KDC i_{cl} or PE $j \notin adj(i_{cl})$. We prove the below lemma for PEs that are adjacent to i_{cl}.

Lemma 3 *The cost of solution for adjacent PEs to KDC i_{cl} is $\sum_{j \in adj(i_{cl})} u_j$*

Proof. If there exists an edge between PE j and KDC i in the graph of Algorithm 1, then we have $u_j = v_{ij} + d_{ij}$, Thus, we can write (substituting the value for $w_{i_{cl}}$):

$$\sum_{j \in adj(i_{cl})} d_{i_{cl}j} + w_{i_{cl}} = \sum_{j \in adj(i_{cl})} v_{i_{cl}j} + d_{i_{cl}j} = \sum_{j \in adj(i_{cl})} u_j \quad \blacksquare \tag{4}$$

Now, the only remaining part of the analysis concerns the PEs that are indirectly connected to KDC i_{cl} (i.e., not adjacent to KDC i_{cl} or PEs that are located in distance 3 of i_{cl}). The lemma below considers these PEs.

Lemma 4 *The cost of solution for indirectly connected PEs to KDC i_{cl} is $d_{i_{cl}j} \leq 3u_j$ for PE $j \notin adj(i_{cl})$.*

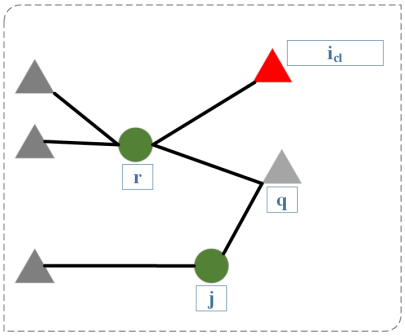

Fig. 4. Indirectly connected PE

Proof. Let's consider the indirectly connected PEs by illustrative example. As can be seen in Fig. 4, PE j is at distance 3 from KDC i_{cl} (PE j connects to KDC i_{cl} indirectly). Based on our observation regarding the existence of an edge in the graph of Algorithm 1, we have $u_j \geq d_{qj}$, $u_r \geq d_{qr}$, and $u_r \geq d_{i_{cl}r}$. Moreover, we have $d_{i_{cl}j} \leq d_{qj} + d_{qr} + d_{i_{cl}r}$. Since we witnessed that KDC i_{cl} was blocked prior to KDC q, and uniform increasing of the dual variable was considered, we have $u_r \leq u_j$. Hence, we can obtain that $d_{i_{cl}j} \leq 3u_j$. ■

By combining Lemmas 3 and 4, the total cost of solution provided by Algorithm 1 is $cost(solution) \leq \sum_{cl} \sum_{j \in cl} 3u_j \leq 3\sum_j u_j \leq 3OPT$, where OPT denotes the value of the optimal solution to the instance of the KDCP. ■

V. Evaluation and Discussion

A. Evaluation Setup

To evaluate the proposed algorithm's performance, we used the Gurobi optimization solver via a Python script to assess the running time and efficiency of our algorithm, as well as to compare the objective value of the KDCP problem. Experiments were performed using a PC running Windows 10 on a 4-core 2.60 GHz Intel Core i7-6700 HQ CPU, equipped with 8GB of RAM. Moreover, throughout the evaluation, we use the words *approximation* and *Primal-Dual algorithm* interchangeably to refer to the proposed Algorithm 1.

We created different types of provider network topologies for performance evaluation purposes. The following network topologies were used:

- Random Internet Graph: since a VPLS is mainly used in large-scale networks and there exists a demand to employ VPLS between different Autonomous Systems (AS), it is highly relevant for our evaluation to use the graph representing Internet communications. The Python script generator for Random Internet Graph yields an undirected graph that resembles the Internet AS network, employing Elmokashfi's approach as in [36].
- Erdős–Rényi model: The Python script generator for this model outputs a random graph in which the degree of every given vertex has a binomial distribution. This random graph of the Erdős–Rényi model is commonly used for evaluating a random network [37].
- Random Graph: network topologies using random graph models exhibit features of a complete graph, such as high interconnectivity among network nodes, while randomly some edges are removed such that the graph is still connected.

The choice of the above network topologies for the performance evaluation of the proposed algorithm is necessitated by the fact that the majority of secure VPLS deployments suffer as they scale up to large-scale networks with a massive number of nodes (such as the ones we are assessing) thereby making the assignment of multiple KDCs to PEs an intractable problem to solve exactly (as we argue in the previous sections). Consequently, the use of mostly huge networks in our assessments is well justified.

In practice, each PE will be called to serve a random number of customer VPNs. Thus, in our assessments, we randomly assign a number of VPNs to each PE. Moreover, in session-key-based VPLS networks, packets containing security keys need to be transferred between each KDC and PE to maintain the security schema. The number of VPNs acquired by each PE will be multiplied by the number of required packets in

Table II
EXECUTION TIME FOR LARGE-SCALE NETWORKS

Topology	Number of PEs (Nodes)	Execution Time
Random Internet Graph (AS networks)	1000	Approx: 0.11 s Exact: 95.98 s
	1500	Approx: 0.27 s Exact: 114.45 s
	2000	Approx: 0.55 s Exact: 144.45 s
Random Graph	500	Approx: 0.12 s Exact: 39.47 s
	800	Approx: 0.19 s Exact: 182.65 s
	1000	Approx: 0.29 s Exact: 558.47 s

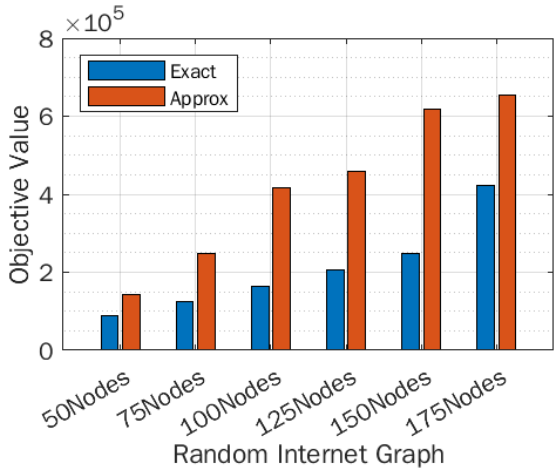

Fig. 5. Total cost for Random Internet Graphs

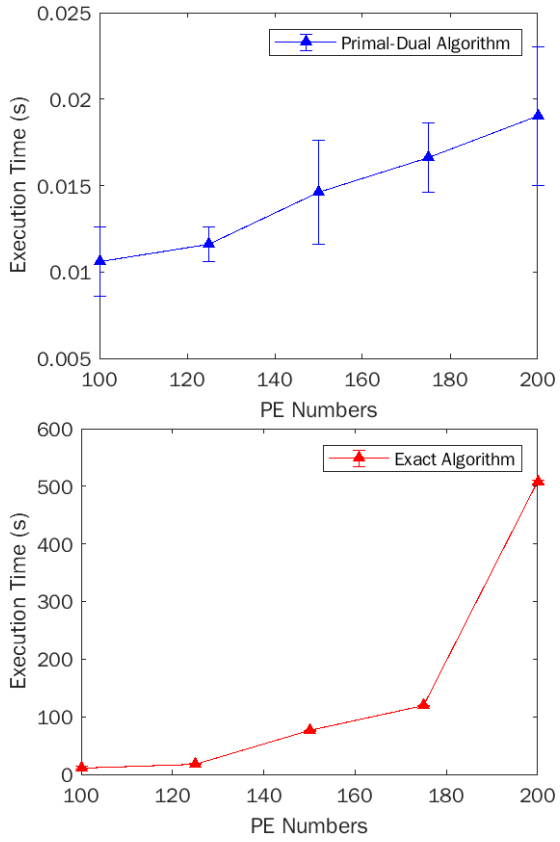

Fig. 6. Execution Time for Erdős–Rényi graph

the secure architecture. This cost will then be incurred on the shortest path between the designated PE and associated KDC. Thus, the cost produced in the previous stage will be multiplied by the weight of the shortest path between KDC and PE. The final value will be denoted as d_{ij}.

The cost of placing each KDC in the network graph can be set by network administrators. If n denotes the number of KDCs, the upper bound on the total synchronization cost between KDCs (e.g., each KDC sends a sync message to other KDCs) can be calculated as $(n-1) \times n \times d_{VPLS}$, where d_{VPLS} denotes the network diameter. The cost is then divided across n KDCs, with each KDC receiving a $((n-1) \times d_{VPLS})$ opening cost.

B. Results and Performance

Table II reports the execution time of the exact algorithm (using the Gurobi solver) versus our primal-dual approximation algorithm for two different topologies i.e., the Random Internet Graph and the Random Graph. To determine the optimal placement of KDCs, Gurobi explores the best solution using the Branch-and-Bound technique. However, due to its high computational cost, this method does not scale well with the number of PEs.

The computation time for optimally placing the KDCs in a large-scale network (i.e., a network with a high number of PEs) appears to be critical. Furthermore, each addition and/or deletion of PEs necessitates recomputing the optimal location for KDCs, possibly contributing to network delays. The execution of KDC placement algorithms is directly related to this delay. In Table II, we start with 1000 PEs in the network for the Random Internet Graph topology. When the network scales up in size, our approximation algorithm proves to be more considerably faster in providing solutions.

We observe that when the number of PEs is increased, the rate of growth in execution time for our algorithm is significantly slower compared to the exact solution. For example, increasing the number of PEs in an AS network from 1000 to 2000 incurs 0.44 seconds more to yield a solution from our approximation algorithms compared to 48.47 seconds for the exact solution. Since the proposed primal-dual algorithm does not need to solve the dual program directly, the primal-dual method seems more applicable than e.g., LP-rounding or Branch-and-Bound (BB), in which the number of nodes in the BB Trees will expand as the number of PEs grows for the exact solution.

As mentioned in Theorem 2, one important concern when designing the approximation algorithm is the guarantee of the produced solutions. It seems that knowing the cost of each solution (i.e., the placement of KDCs in a provider network) is critical for provider networks, since lower network costs result in lower operational costs. Fig. 5 compares the objective

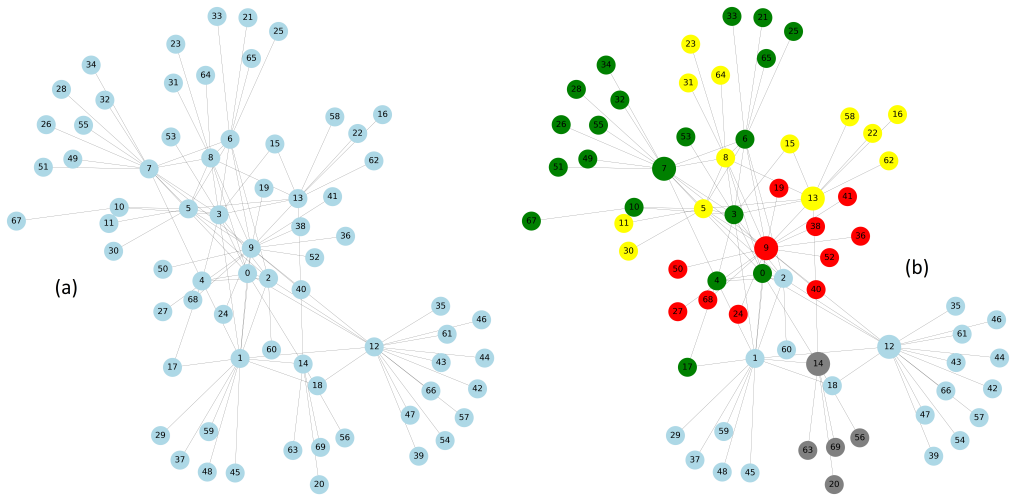

Fig. 7. Exact solution for the Random Internet Graph with 70 PEs

values for the KDCP problem, i.e., the total cost, in both exact and approximation solutions for different network sizes.

For this configuration, a random number of VPNs between 1 and 1000 was chosen for each PE, as well as a random activation cost. Fig. 5 depicts the objective value for both the exact and approximation algorithms. For all networks, the approximation algorithm yielded a higher objective value. However, the ratio of approximation objective value to exact value (i.e., the value of $\frac{\texttt{Objective Value (Approximation)}}{\texttt{Objective Value(Exact)}}$) demonstrates the constant-factor approximation proved in Theorem 2. With [50, 75, 100, 125, 150, 175] PEs, ratio is [1.61, 1.96, 2.52, 2.22, 2.48, 1.55].

In the next experiment, the Erdős–Rényi model is used as the network graph. Fig. 6 depicts the execution time in seconds versus the number of PEs. When the primal-dual algorithm is employed for a network with 100 to 200 PEs, the blue and the red curves illustrate the execution time of the approximation and the exact algorithm, respectively. The execution time generally increases with the number of PEs and ranges around 0.01 seconds for the primal-dual approximation. Conversely, it greatly increases with the number of PEs for the exact solution. When the number of PEs increases from 100 to 200, the exact algorithm begins with a modest rise in execution time. However, for a large number of PEs in the network, the slope of execution time growth becomes considerable. This experiment suggests that, even though the exact algorithm produces the optimal KDC placement, it is not applicable for large-scale network deployments. Fig. 7 shows an example output for an exact algorithm running on the Random Internet Graph.

Fig. 7(a) illustrates the Random Internet Graph, which has 70 PE routers (depicted as the vertex nodes in the figure) connected by edge links. A random number of VPNs from client sites must be supported by each PE. The number of VPNs assigned to each PE was chosen at random from a range of 1 to 300 VPNs. The Dijkstra algorithm was used

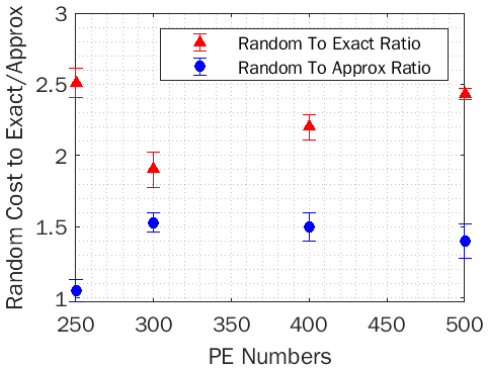

Fig. 8. Ratio of random cost

to calculate the shortest distance between two graph nodes.

The network administrator randomly determined the weight of each link in this graph. According to the exact solution of KDCP, shown in Fig.7(b), five KDCs will be opened in positions 7,9,12,13, and 14. Each KDC, along with its corresponding PEs (clients), has been assigned a particular color to demonstrate the assignment of each PE to its corresponding KDC (i.e., 7:green, 9: red, 12:light blue, 13:yellow, and 14:grey).

In the next experiment, we randomly place KDCs in the Random Internet Graph with 250, 300, 400, and 500 PEs to demonstrate that the exact and the approximate solution offer a better cost than randomly placing KDCs. We also compute the cost of the solution generated by random placement, which we denote as random cost. We divide the random cost by the costs of the solutions by exact algorithm and approximation algorithm, respectively. In Fig. 8, the blue circles indicate the ratio of dividing the random cost by the cost of the approximate solution, while the red circles depict the random to exact cost ratio. For all PE numbers, the ratio of random to exact cost is greater than the ratio of random to approximate

cost, since the exact solution is optimal and, hence, lower than the approximate solution in general. Furthermore, for all PE numbers, the ratio of random to approximate cost (blue circles) is greater than 1, suggesting that algorithm 1 gives a better solution than random KDC placement.

To create an efficient VPLS network, several VPLS architectures have been considered in literature. HIPLS is one of the most popular VPLS schemes for providing a secure VPLS architecture. It is also the first VPLS design to present a public key authentication structure as well as a data traffic encryption mechanism. A representative indicator of the scalability of each protocol is the key storage requirement at each component of the VPLS. Herein, we compare the key storage requirements of our proposed method, in which multiple KDCs are deployed to secure the network, with S-HIPLS and HIPLS.

Fig. 9 represents the key stores in the KDC versus the number of PEs in the network. HIPLS employs an authentication server (AS), and our architecture contains a KDC that distributes keys (CEK and KEK) to PEs. We set the number of VPNs to $v = 5$, and the number of PEs increases from 1 to 100. We observe that a similar linear increment of the number of keys can be seen in both HIPLS and S-HIPLS architectures, in which S-HIPLS requires a higher number of keys for its operation.

The main reason is that the complexity of key storage at AS/KDC of HIPLS proved to be $O(m)$, where m is the number of PEs. On the other hand, the complexity of S-HIPLS is $O(m+v)$, where v is the number of VPNs [17]. Moreover, our proposed method employs the multiple KDC in the VPLS network. Thus, each KDC will be responsible for a portion of all PEs, and the complexity of our proposed architecture is $O(\frac{m+v}{n})$, where n is the number of KDCs calculated in Algorithm 1. Thus, the proposed architecture reduces the key complexity at the KDC level and is more efficient than S-HIPLS and HIPLS.

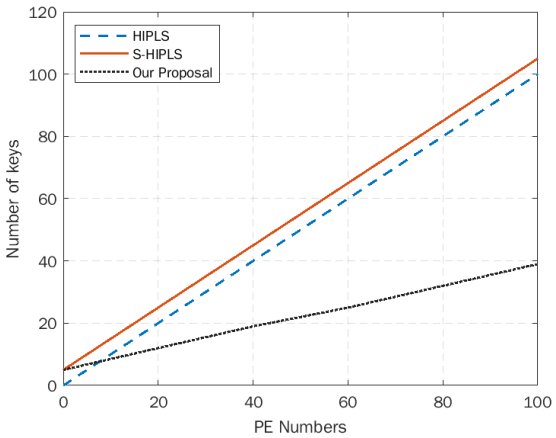

Fig. 9. Number of required keys versus number of PEs for the operation of a KDC/AS network with 5 VPNs

Fig. 10 illustrates how the key stores scale with the number of VPNs at AS/KDC while serving 100 PEs in the network. For our architecture, Fig. 10 shows a linear increase in

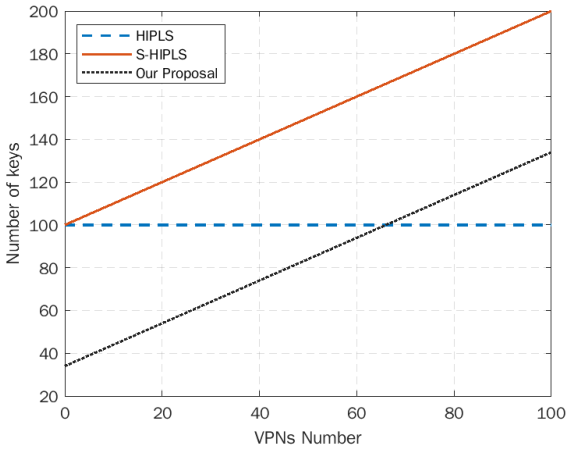

Fig. 10. Number of required keys versus number of VPNs for the operation of a KDC/AS network with 100 PEs

the number of keys stored at the KDC, but for the HIPLS architecture, it remains constant. As a result, the number of HIPLS keys at the AS is unaffected by the number of VPNs in the provider network. Our proposed architecture shows less key storage compared to S-HIPLS. Comparison between our proposed architecture and HIPLS reflects that the difference in key storage complexity is proportional to the number of VPNs in the network, and the difference is less significant as long as the number of VPNs is restricted. To conclude, since in real-world scenarios $m >> v$ i.e., the number of PEs far outweigh the number of VPNs, our method seems to offer better scalability compared to both HIPLS and S-HIPLS for the majority of real-world deployments.

VI. Conclusion

We proposed a distributed session-key HIPLS architecture by finding the optimal and near-optimal placement of multiple KDCs in a provider network. We optimized for the cost of activating KDCs in the provider network and the cost of providing secure communication for each VPN as a result of transferring keys between PEs and KDCs.

The high computational cost of obtaining optimal solutions renders exact solutions impractical for large-scale network deployments, even when using sophisticated optimization software such as Gurobi. Therefore, we propose the use of a constant-factor approximation algorithm to effectively get near-optimal solutions, which additionally inherits from S-HIPLS: mutual authentication, PE authorization, data and control frame encryption, privacy protection, secure control protocol, and resistance to various attacks.

Our evaluation results indicate that the approximation approach may be used for large networks with a variety of real-world topologies. Furthermore, our proposal to apply near-optimal placement of distributed KDCs in the provider network requires storing fewer keys compared to both HIPLS and S-HIPLS. Consequently, our approach offers more security features than other secure VPLS architectures with the benefit

of improved scalability and, hence, usefulness in real-world scenarios.

A future direction of this work will investigate the effect of including additional constraints in the problem formulation. For instance, a restriction on the capacity for each KDC could be added so that each KDC would be able to meet the demands of a limited number of PEs.

REFERENCES

[1] A. Alipour-Fanid, M. Dabaghchian, N. Wang, L. Jiao, and K. Zeng, "Online-Learning-Based Defense Against Jamming Attacks in Multi-channel Wireless CPS," *IEEE Internet of Things Journal*, pp. 13 278–13 290, 2021.

[2] M. Conti and F. Turrin, *Cyber Forensics for CPS*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2019, pp. 1–3.

[3] S. Kim, K.-J. Park, and C. Lu, "A Survey on Network Security for Cyber–Physical Systems: From Threats to Resilient Design," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 3, pp. 1534–1573, 2022.

[4] K. Tange, M. De Donno, X. Fafoutis, and N. Dragoni, "A Systematic Survey of Industrial Internet of Things Security: Requirements and Fog Computing Opportunities," *IEEE Communications Surveys Tutorials*, vol. 22, no. 4, pp. 2489–2520, 2020.

[5] C. Boudagdigue, A. Benslimane, A. Kobbane, and J. Liu, "Trust Management in Industrial Internet of Things," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3667–3682, 2020.

[6] L. Li, Y. Luo, J. Yang, and L. Pu, "Reinforcement Learning Enabled Intelligent Energy Attack in Green IoT Networks," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 644–658, 2022.

[7] A. Alshalan, S. Pisharody, and D. Huang, "A survey of mobile VPN technologies," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1177–1196, 2015.

[8] K. Gaur, A. Kalla, J. Grover, M. Borhani, A. Gurtov, and M. Liyanage, "A survey of Virtual Private LAN Services (VPLS): Past, present and future," *Computer Networks*, p. 108245, 2021.

[9] "Cisco VPLS Project," Website, 2019. [Online]. Available: https://www.cisco.com/c/en/us/products/ios-nx-os-software/virtual-private-lan-services-vpls

[10] "Juniper Networks-VPLS," Website, 2019. [Online]. Available: https://www.juniper.net/documentation/en_US/junos/topics/concept/vpls-security-overview.html

[11] "Vodafone VPN Services," Website, 2019. [Online]. Available: https://www.vodafone.co.uk/business/business-connectivity/data-connectivity/Ethernet-services/vpn

[12] "VPLS Estimation," Website, 2019. [Online]. Available: https://www.theexpresswire.com/pressrelease/Virtual-Private-LAN-Service-Market-is-Estimated-to-Reach-2420-Million-by-2025_11133956

[13] T. Henderson, S. Venema, and D. Mattes, "HIP-based virtual private LAN service (HIPLS)," *Internet Draft, IETF*, 2011.

[14] T. Henderson, "Boeing HIP Secure Mobile Architecture," *Tech. Rep.*, 2008.

[15] "Whitepaper IDN," Website. [Online]. Available: https://www.tempered.io/

[16] M. Liyanage and A. Gurtov, "A scalable and secure VPLS architecture for provider provisioned networks," in *2013 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2013, pp. 1115–1120.

[17] M. Liyanage and A. Gurtov, "Securing Virtual Private LAN Service by Efficient Key Management," *Security and Communication Networks*, vol. 7, no. 1, pp. 1–13, 2014.

[18] M. Liyanage, M. Ylianttila, and A. Gurtov, "Secure Hierarchical Virtual Private LAN Services for Provider Provisioned Networks," in *2013 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2013, pp. 233–241.

[19] M. Liyanage, M. Ylianttila, and A. Gurtov, "Secure Hierarchical VPLS Architecture for Provider Provisioned Networks," *IEEE Access*, vol. 3, pp. 967–984, 2015.

[20] M. Liyanage, M. Ylianttila, and A. Gurtov, "Fast Transmission Mechanism for Secure VPLS Architectures," in *2017 IEEE International Conference on Computer and Information Technology (CIT)*. IEEE, 2017, pp. 192–196.

[21] M. Liyanage, M. Ylianttila, and A. Gurtov, "Software defined VPLS architectures: Opportunities and challenges," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. IEEE, 2017, pp. 1–7.

[22] H. Cui, G. O. Karame, F. Klaedtke, and R. Bifulco, "On the Fingerprinting of Software-Defined Networks," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 10, pp. 2160–2173, 2016.

[23] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: A distributed control platform for large-scale production networks," in *9th USENIX Symposium on Operating Systems Design and Implementation (OSDI 10)*. USENIX Association, 2010.

[24] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 7–12. [Online]. Available: https://doi.org/10.1145/2342441.2342444

[25] J. Lu, Z. Zhang, T. Hu, P. Yi, and J. Lan, "A Survey of Controller Placement Problem in Software-Defined Networking," *IEEE Access*, vol. 7, pp. 24 290–24 307, 2019.

[26] S.-K. Yoon, Z. Khalib, N. Yaakob, and A. Amir, "Controller Placement Algorithms in Software Defined Network - A Review of Trends and Challenges," *MATEC Web of Conferences*, vol. 140, p. 01014, 2017. [Online]. Available: http://www.matec-conferences.org/10.1051/matecconf/201714001014

[27] B. P. R. Killi and S. V. Rao, "On Placement of Hypervisors and Controllers in Virtualized Software Defined Network," *IEEE Transactions on Network and Service Management*, vol. 15, no. 2, pp. 840–853, Jun. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8331922/

[28] R. Moskowitz, P. Nikander, and T. Henderson, "Host Identity Protocol," 2008.

[29] V. Varadharajan, K. Karmakar, U. Tupakula, and M. Hitchens, "A Policy-Based Security Architecture for Software-Defined Networks," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 897–912, 2019.

[30] K. Zeng, K. Ren, W. Lou, and P. J. Moran, "Energy-Aware Geographic Routing in Lossy Wireless Sensor Networks with Environmental Energy Supply," in *Proceedings of the 3rd International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks*. New York, NY, USA: Association for Computing Machinery, 2006.

[31] Conti, Mauro and Vinod, P. and Tricomi, Pier Paolo, "Secure Static Content Delivery for CDN Using Blockchain Technology," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Cham: Springer International Publishing, 2022, pp. 301–309.

[32] K. Jain, M. Mahdian, and A. Saberi, "A New Greedy Approach for Facility Location Problems," in *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, ser. STOC '02. New York, NY, USA: Association for Computing Machinery, 2002, p. 731–740. [Online]. Available: https://doi.org/10.1145/509907.510012

[33] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2021. [Online]. Available: https://www.gurobi.com

[34] V. V. Vazirani, *Approximation algorithms*. Springer, 2001. [Online]. Available: http://www.springer.com/computer/theoretical+computer+science/book/978-3-540-65367-7

[35] K. Jain and V. V. Vazirani, "Approximation Algorithms for Metric Facility Location and k-Median Problems Using the Primal-Dual Schema and Lagrangian Relaxation," *J. ACM*, vol. 48, no. 2, p. 274–296, mar 2001. [Online]. Available: https://doi.org/10.1145/375827.375845

[36] A. Elmokashfi, A. Kvalbein, and C. Dovrolis, "On the Scalability of BGP: The Role of Topology Growth," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 8, pp. 1250–1261, 2010.

[37] B. Bollobas, "The Evolution of Random Graphs," *Transactions of the American Mathematical Society*, vol. 286, no. 1, pp. 257–274, 1984. [Online]. Available: http://www.jstor.org/stable/1999405