Deploying Robust Security in Internet of Things

Ruozhou Yu, Guoliang Xue, Vishnu Teja Kilari, Xiang Zhang
Arizona State University, {ruozhouy, xue, vkilari, xzhan229}@asu.edu

***Abstract*—Popularization of the Internet-of-Things (IoT) has brought widespread concerns on IoT security, especially in face of several recent security incidents related to IoT devices. Due to the resource-constrained nature of many IoT devices, security offloading has been proposed to provide good-enough security for IoT with minimum overhead on the devices. In this paper, we investigate the inevitable risk associated with security offloading: the unprotected and unmonitored transmission from IoT devices to the offloaded security mechanisms. An important challenge in modeling the security risk is the dynamic nature of IoT due to demand fluctuations and infrastructure instability. We propose a stochastic model to capture both the expected and worst-case security risks of an IoT system. We then propose a framework to efficiently address the optimal robust deployment of security mechanisms in IoT. We use results from extensive simulations to demonstrate the superb performance and efficiency of our approach compared to several other algorithms.**

***Keywords*—Internet-of-Things, security offloading, robustness, risk measurement, conditional value-at-risk**

I. Introduction

Despite its powerfulness and popularity, the current IoT is facing many challenges, among which a major one is the emerging concern on IoT security and privacy. In regard to security, the massive number of connected smart devices in IoT, which is indeed its biggest strength, is also a huge potential threat. On one hand, providing fine-grained security to a large number of geo-distributed devices is non-trivial, especially given the limited resources on each of these devices. On the other hand, these massive devices, once compromised, can be used as a powerful weapon against the system itself, for example, by launching large-scale distributed denial-of-service (DDoS) attacks. In fact, the digital world has already witnessed the devastating effect of such attacks in several recent incidents [32]. Privacy becomes part of the concern when IoT-connected devices infiltrate various private spaces, including homes, factories, and hospitals.

Enforcing security and privacy in IoT is difficult. The main reason, as aforementioned, is the limited resources (computing resource, memory, battery) on connected devices. Due to this, IoT devices can hardly run conventional cryptographic algorithms. Currently, there are two directions that deal with this issue. The first direction is to develop lightweight yet strong-enough cryptographic algorithms for IoT devices. Unfortunately, current advances along this direction is yet enough to conquer the overhead issue of cryptography in constrained environments [31]. The second direction is to offload security to the cloud [6], [37]. This is based on recent advances in network function virtualization (NFV), which can virtualize security mechanisms as software components to be run on general-purpose platforms [20]. However, cloud-based offloading has several drawbacks. First, cloud-based security leaves data transmission as a major vulnerability. In other words, the traffic from IoT devices to the cloud is unprotected, leaving room for data interception, manipulation and injection attacks, especially when the traffic needs to traverse the Internet before reaching the cloud. Second, cloud-based security cannot prevent saturation attacks at the early stage, such as DDoS attacks from IoT devices. In fact, a DDoS attack may even invalidate the cloud by saturating its bandwidth, rendering the whole security system unusable. Third, the cloud suffers from high latency which increases the probability of device-oriented oppotunistic attacks, for instance, the recently revealed Meltdown and Spectre hardware attacks [12] that can affect an extremely wide spectrum of IoT devices. The cloud is not capable of responding in real-time to incidents on IoT devices. Last but not least, the cloud becomes a single point of failure in the system. An attack compromising the cloud itself can have control over all the devices, including those in private spaces like homes or factories.

Powered by fog computing, a new approach to IoT security emerges. Fog computing offers in-network computing hardware in the edge network, which enables edge offloading of security mechanisms near the end devices. Hence it can largely realize fine-grained and real-time security, while avoiding high latency, high bandwidth usage and a single point of failure. That being said, fog-based security also has its limitations. First of all, the cost of using fog computing is generally higher than that of using the cloud. This is because fog nodes are commonly deployed in areas that are already dense with other devices, and hence will have higher deployment, operation and energy costs. Due to this, fog resources are still limited in each area, though much more abundant than IoT devices. Second, fog-based security still leaves some vulnerabilities in the early stage of data transmission. The unprotected and/or unmonitored traffic can cause various threats to the devices and the system, as we detail in Sec. III-B.

An ideal architecture for IoT security should jointly make use of end device-, cloud- and fog-based security. Among these, fog-based security has the highest flexibility in the trade-off between security and cost, and hence should be carefully optimized by the IoT operator. In this work we focus on modeling and formulating the security offloading problem from the perspective of fog computing. Given a limited cost budget, the operator would want to deploy security functions on distributed fog nodes, in order to minimize the security risk experienced by the end devices. We mathematically formulate the security risk of the system in terms of the distance from each device to the nearest deployed security function, which is general enough to incorporate a wide range of risk measures in practice; see Sec. III-B. A major challenge in the deployment problem is the dynamic nature of IoT, where both infrastructure and demand fluctuations could happen frequently, such as device mobility, maintenance, interference, failures, etc. To address this challenge, we propose a stochastic model to

978-1-5386-4586-4/18/$31.00 ©2018 IEEE

capture the uncertainties caused by dynamics. Leveraging a relevant concept in economics, our model can account for both the expected and the worst-case risks of the system, which is more robust than traditional stochastic models solely based on expectations. We then propose a novel decomposition-based framework, along with an efficiency-enhancement technique, to achieve accurate and efficient optimization of system security risk in the dynamic IoT environment. We evaluated our proposed model and optimization framework in extensive simulation experiments, and the results have shown the superb performance and efficiency of our proposed approach.

To summarize, our main contributions are as follows:

- To the best of our knowledge, we are the first to study, quantitatively model, and optimize the security risk in fog computing-based IoT security offloading. The problem is of both theoretical and practical importance.
- We propose a stochastic model to account for both the expected and the worst-case security risks of the IoT system with uncertainties, which is more robust than traditional stochastic models solely based on expectations, and is more suitable for security-related use cases.
- We propose a decomposition-based optimization framework, along with an efficiency-enhancement method that addresses the large overhead of stochastic programming.
- We conducted extensive simulations which show the superb performance of our approach compared to several other approaches.

II. Backbround and Related Work

A. IoT Security Challenges and Approaches

IoT security has yet attracted a lot of attentions, at least not until several recent attacks based on IoT devices [32]. It has then been recognized that security breaches in IoT can be as dominating and devastating as, if not more so than, any other known type of security incidents. Since then, more efforts have been put into addressing security challenges in IoT, which can be viewed roughly in two perspectives.

The first perspective is on protecting the IoT devices. The main challenge here is the limited resources on IoT devices, including computing power, memory, power supply, etc. Currently, there are two approaches to address the resource issue. The first one is lightweight cryptography and security, which aims to develop mechanisms that can provide good-enough security on resource-constrained devices, such as [13], [19]. Unfortunately, the current lightweight methods still can only be deployed on devices like smartphones or tablets, but can hardly be used on smaller devices like radio frequency identification (RFID) tags, largely due to the stringent power constraint on these devices [31]. Another approach is to offload security mechanisms to other platforms, including in-network fog nodes or the cloud [1], [6], [14], [34]. This approach addresses the overhead issue, but inevitably introduces some risk associated with the transmission to the offloaded security mechanisms, as detailed in the next section.

The second perspective is to protect the broader IoT system rather than only the end devices. An IoT system includes the end devices, network infrastructures in different levels, applications and their hosting fog or cloud nodes, and users of the IoT services. There are several challenges in this perspective, including the huge number of IoT devices, the heterogeneous and dynamic network environment, various requirements and characteristics of applications, etc. The *de facto* approach here is cloud-based security [27], which again has several issues. The cloud is generally far away from the IoT edge network, and hence is difficult to adapt to the fast changing environment of IoT; it is also vulnerable to saturation attacks like DDoS [30]. NFV [29] is a promising approach to addressing this issue, by enabling implementation of security mechanisms as in-network software components to resolve threats before they reach the cloud, other devices or the users.

It can be seen that both perspectives call for deployment of security mechanisms in the network. In both cases, security can be largely guaranteed when traffic has passed certain security functions, while the part of transmission before is not protected/monitored. Therefore, what we address in this paper is to minimize the risk associated with the transmission before the security functions, by deploying these functions at optimal locations. A recent survey on IoT security is in [36].

B. Risk Management in Network Security

Traditionally, network security is mainly a 0–1 problem: a system is either secure or insecure. Recently, however, there has been a transition to providing best-effort network security, due to the enormous number and variety of attacks and the impractical amount of resources to prevent them all. The goal is either to make the difficulty or cost of launching an attack unacceptable to the attackers, or to make the probability or potential loss of undergoing an attack acceptable to the system.

Our work follows a number of existing works along the second direction. A major series of works have focused on modeling and optimizing network security risk using Attack Graphs (AG) [2], [23], [28]. An AG is a graph representation of all the possible attack paths into the system, and is used to derive various security risk measures via Bayesian theory [23], node ranking [28], or other mathematical tools [2]. The derived measures are then used to guide the deployment of security functions to harden the system [8], [22]. A main drawback of the AG representation is its scalability. The size of the AG can grow exponentially with the size of the network, and is further increased if each node have multiple vulnerabilities. Hence it is most suitable for simple and small-size environments like home networks, but is hardly applicable to IoT networks with even a few hundred devices. Furthermore, AG cannot handle fast dynamics in the network: it needs to be modified each time a new device joins or an old device leaves.

In face of these issues, some researchers have been seeking for simpler and more practical security measures for large-scale and dynamic environments like IoT. Rullo, Serra, Bertino, and Lobo [26] have proposed a novel model for security risk when monitoring geo-distributed IoT devices in an area. They considered the robustness of the system when facing dynamic user densities, an approach similar to what we use to address demand and topology fluctuations in the IoT network. In this paper, we propose a security risk measure based on distances in the network, as well as an efficient optimization framework to minimize the security risk by flexibly deploying security functions on fog nodes.

C. Other Related Areas

Mobile Offloading (MO): The idea of offloading dates back to mobile cloud computing, where users offload their mobile applications to the cloud [16]. Yet security offloading differs from MO in two ways. First, MO is mostly ad hoc where each user makes its own decisions; however, security offloading is centrally controlled by the operator, who optimally coordinates the offloading for all users. Second, MO does not consider the network, since the destination is commonly the cloud. Security offloading needs to consider the edge network topology, and to minimize the risk associated with the offloading decisions.

NFV and Service Function Chaining (SFC): SFC is arisen in the context of NFV. SFC considers the interplay between different network services or security functions, modeling them as a chain of virtual functions. Existing work has focused on embedding service chains for each traffic flow [4], [17], [25]. Unfortunately, this method is neither scalable for the massive IoT devices, nor robust to the high dynamics. A good method should account for both the expected scenarios, and possible unfavorable scenarios due to system fluctuations.

Security with Robustness: Our work uses a stochastic approach to ensure system robustness. Similar methods have also been used in other security scenarios [26], [33]. Another potential approach is robust optimization, which deterministically optimizes the worst possible performance of the system, such as [5]. This approach has two drawbacks. First, it is hard to represent all unfavorable scenarios deterministically. Second, it only plans for the worst possible performance of the solution, which is commonly an overkill and wastes precious resources. A good approach should be able to balance the expected and worst-case performances, and give operator the flexibility to specify the desired objective.

III. Problem Description and Formulation

A. System Model

The IoT network is modeled as a directed graph $G = (V, E)$. V is the set of network nodes, including access points (APs), switches, routers, gateways, and various fog nodes. We use $F \subseteq V$ to denote the set of fog nodes, each being able to host security functions. E is the set of links between nodes.

The IoT faces two types of uncertainties. The first one is demand uncertainty, which comes from dynamics such as device deployment, maintenance, user load variation, device and user mobility, etc. Let $A \subseteq V$ be the set of APs of the IoT. To model demand uncertainty, each AP $a \in A$ is associated with a random variable $d_a \in \mathbb{R}^*$ ($\mathbb{R}^*$ is the non-negative real number set), denoting the amount of demand at a. We use $D = \{d_a \,|\, a \in A\}$ to denote the demand uncertainty set, and $d_{\text{sum}} = \sum_{a \in A} d_a$ to denote the total demand in the IoT. In practice, the demand can refer to (but is not limited to) the number of devices, number of flows, traffic volumes, etc.

The second type is topology uncertainty due to unexpected failures, maintenance, interference, etc. To model it, we associate each link $e \in E$ also with a random variable $y_e \in \{0, 1\}$, where $y_e = 1$ means e is operational, and $y_e = 0$ means e is down. This model is also able to account for node dynamics: if a node fails, all its adjacent links have $y_e = 0$. We use $Y = \{y_e \,|\, e \in E\}$ to denote the topology uncertainty set.

Let $\overline{D} = \{\overline{d}_a\}$ and $\overline{Y} = \{\overline{y}_e\}$ be a specific realization of the demand and topology uncertainty sets, respectively. We define a realization of the system (called a *scenario*) as $\Pi = (\overline{D}, \overline{Y})$.

B. Threat Model and Defense Mechanism

In an IoT network with security offloading, data is first transmitted from IoT device to the offloaded security function before heading to its final destination. Below, we analyze the threat associated with such transmission in several scenarios:

- *Highly constrained devices:* these devices cannot run any cryptographic procedure, hence offload all procedures to the network. All transmitted data is unprotected before passing the security functions, which is subject to leakage, manipulation, injection and other types of attacks.
- *Moderately constrained devices:* these devices can run lightweight cryptographic procedures such as secure computation outsourcing [6], [14], which only offloads computation-intensive operations but keeps the data private. Yet in this case, the data stays unprotected in device memory until encryption is complete, during which it can be compromised by opportunistic attacks leveraging vulnerabilities of the devices themselves, such as the recent Meltdown and Spectre hardware attacks [12]. Probability of an attack is determined by the latency between device and the offloaded location, as the process can involve multiple rounds of back-and-force messaging [14].
- *IoT network:* the operator may deploy security functions (*e.g.*, intrusion detection, firewall, deep packet inspection) to protect the system from malicious/compromised devices. However, traffic not processed by certain security functions can be a threat to the network. For example, a DDoS attack may still overwhelm some of the nodes before being tackled by a detection and resolution function.

It can be seen that in these various scenarios, it is essential that the operator can flexibly deploy security functions to minimize the threats brought by the unprotected/unmonitored transmission of offloading. Unfortunately, the operator may have a tight budget for deploying the functions. We next define the operator's deployment plan to optimize such threats.

We consider the deployment of a single type of security function to prevent a specific type of attacks. Formally, we use binary variable set $X = \{x_v \,|\, v \in F\}$ to denote a *deployment plan*, where $x_v = 1$ means a security function is deployed at node v and 0 otherwise. Let $c_v \in \mathbb{R}^+$ be the deployment cost at node $v \in F$. A deployment plan is said to be *feasible*, iff it satisfies the operator's budget b:

$$\sum\nolimits_{v \in F} c_v x_v \leq b. \tag{1}$$

Definition 1 (Security risk). *Given network $G = (V, E)$, the **security risk** of a deployment plan X is the average distance that a unit of demand has to traverse before reaching the nearest security function from its AP, denoted as $R(X, D, Y)$. In other words, the security risk is a function of the deployment plan X and the uncertainty sets D and Y.* □

In Definition 1, the distance is a cumulative measure that can be selected based on different use cases. For instance, it can be as simple as the number of hops (vulnerable links or nodes) or transmission latency (time window of possible

attacks), or as complex as the negative logarithm of the "safe" probability of traversed links or nodes (the "safe" probability is one minus the probability that a node/link/path is attacked; it is cumulated multiplicatively along the transmission path but is to be maximized instead of minimized). We do not assume a specific distance measure for sake of generality. We focus on the average distance of demands for ease of illustration, although our approach can be trivially extended to minimizing the maximum distance of demands. Note that the average distance is the average over demands from all APs in a fixed scenario, rather than over all possible scenarios of the system. For measurement over scenarios, we use both the expectation and a worst case-oriented metric, as detailed next.

C. Measuring Security Risk with Uncertainty

Given a fixed scenario Π, measuring the security risk of deployment plan X is as simple as finding shortest paths between AP-fog node pairs; minimizing it with flexible deployment plans, though, is NP-hard due to a reduction from the facility location problem [11]. Furthermore, the IoT environment is rather volatile, with uncertainties in both the demands and the topology. One approach is to measure and minimize the *expected risk*, which reflects the average level of threat of the system. However, such an approach is not robust enough when applied in security, as the system may experience arbitrarily high security risk in unfortunate scenarios.

To account for the worst-case performance, we adopt the concept of *Conditional Value-at-Risk* (CVaR), a risk measure widely used in economics and finance. The concept has been applied in several security-related use cases [26], [33]. Here "risk" refers to the investment risk an investor encounters when facing market variations, *i.e.*, potential loss due to unfavorable market trends. Formally, let random variable R be the loss of an investment. The following terms are defined:

$$\text{VaR}_\alpha(R) = \min\{c \,|\, P(R \le c) \ge \alpha\}, \tag{2}$$

$$\text{CVaR}_\alpha(R) = \mathbb{E}[R \,|\, R \ge \text{VaR}_\alpha(R)]. \tag{3}$$

Here $\mathbb{E}[\cdot]$ is the expected value of a random variable. $\text{VaR}_\alpha(R)$ (Value-at-Risk, also called α-VaR) is the minimum value such that the actual loss will not exceed it with α confidence, while $\text{CVaR}_\alpha(R)$ (also called α-CVaR) is the expected loss of all scenarios where the loss can actually exceed $\text{VaR}_\alpha(R)$. In other words, $\text{CVaR}_\alpha(R)$ denotes the expected loss of the worst $(1-\alpha)$ percent scenarios in terms of investment loss.

Given uncertainty sets D and Y, $R(X, D, Y)$ is the security risk of deployment plan X, which is also a random variable. We use R to denote $R(X, D, Y)$ if no ambiguity is introduced. Next, we formally model and minimize the security risk of security offloading, utilizing the CVaR definition.

IV. Security Deployment with Uncertainty

A. Problem Description and Formulation

Given the network and a limited budget, the IoT operator wants to ensure system security to the best extent. This includes both the overall system security in expectation, as well as the potential security risk in the $(1-\alpha)$ percent most unfavorable scenarios. We model these two goals as a multi-objective optimization problem as follows:

$$\min_{X \in \mathcal{X}} \quad \mathbb{E}[R],\ \text{CVaR}_\alpha(R), \tag{4}$$

where $\mathcal{X}$ is the feasible deployment plan set satisfying Eq. (1).

Optimizing two objectives can be hard, as they may conflict with each other in their own optimality points respectively. A common technique is to *scalarize* the multiple objective functions into a single objective function. We scalarize Program (4) as the following single-objective program:

$$\min_{X \in \mathcal{X}} \quad \mathbb{E}[R] + \rho \cdot \text{CVaR}_\alpha(R). \tag{5}$$

where ρ is a chosen balancing parameter.

In Eq. (3), the formulation of α-CVaR requires the computation of α-VaR beforehand, which is hard to incorporate in the above program. Fortunately, Rockafellar and Uryasev proved in [24] that the α-CVaR can be computed as follows without knowing the α-VaR beforehand:

$$\text{CVaR}_\alpha(R) = \min_{c \in \mathbb{R}} \left\{ c + \frac{1}{1-\alpha} \mathbb{E}\left[(R-c)^+\right] \right\}, \tag{6}$$

where $\mathbb{R}$ is the real number set, and $(z)^+ = \max\{z, 0\}$. Therefore, Program (5) can be re-written, by incorporating Eq. (6), as the following program:

$$\min_{\substack{X \in \mathcal{X} \\ c \in \mathbb{R}}} \ \mathbb{E}[R] + \rho \left(c + \frac{1}{1-\alpha} \mathbb{E}\left[(R-c)^+\right] \right). \tag{7}$$

The random variable R is a function of the deployment plan X and random variable sets D and Y. Unfortunately, writing it as a closed-form equation is also a difficult task. Instead, we formulate it as the following program:

$R(X, D, Y) =$

$$\min_{t} \quad \frac{1}{d_{\text{sum}}} \sum_{a \in A} d_a \sum_{v \in F} \text{dist}_a(v) t_a(v) \tag{8a}$$

$$\text{s.t.} \quad \sum_{v} t_a(v) = 1, \quad \forall a; \tag{8b}$$

$$t_a(v) \le x_v, \qquad \forall a, v; \tag{8c}$$

$$t_a(v) \in [0, 1], \qquad \forall a, v. \tag{8d}$$

In Program (8), $\text{dist}_a(v)$ is a random variable, denoting the distance between AP a and node v. It is determined by the distance metric used (number of hops, latency, safe probability, etc.), as well as the topology uncertainty set Y. Objective (8a) is to minimize the demand-weighted average security risk (distance) of all APs. Constraint (8b) bounds the node selection variable $t_a(v)$. Constraint (8c) defines the relationship between the deployment variable x_v and the node selection variable $t_a(v)$. Note that $t_a(v)$, which can be viewed as the probability of selecting v for AP a, is a continuous variable in $[0, 1]$; its upper bound 1 is explicitly expressed in (8d) for clarity, but can be omitted due to Constraint (8b) when solving the program. If multiple nodes have the same minimum distance from a, the node selection can be arbitrarily split among them without affecting the objective value.

B. Scenario-based Stochastic Optimization

Program (7) is a stochastic optimization problem. Even with the characteristic functions of D and Y, the problem is still hard to solve since $R(X, D, Y)$ cannot be written in a closed form (and is not even convex as Y is discrete). A common approach is to approximate the expectations using sampling. Specifically, N sample scenarios are obtained from the underlying distributions of D and Y, denoted as $\mathbf{\Pi} =$

$\{\Pi_1, \ldots, \Pi_N\}$. Let $\overline{R}_i \triangleq R(X, \overline{D}^i, \overline{Y}^i)$ be the security risk in scenario Π_i, which can still be expressed by Program (8) with the random variables replaced by the deterministic values in Π_i. The expected security risk $\mathbb{E}[R]$ is then approximated by the sample average function $\frac{1}{N}\sum_{i=1}^{N}\overline{R}_i$, while the α-CVaR is approximated by $\min_{c\in\mathbb{R}}\left\{c + \frac{1}{1-\alpha}\frac{1}{N}\sum_{i=1}^{N}(\overline{R}_i - c)^+\right\}$.

Program (7) is then approximated by the following:

$$\min_{\substack{X\in\mathcal{X}\\ c\in\mathbb{R}}} \frac{1}{N}\sum_{i=1}^{N}\overline{R}_i + \rho\left(c + \frac{1}{1-\alpha}\frac{1}{N}\sum_{i=1}^{N}\left(\overline{R}_i - c\right)^+\right). \quad (9)$$

We next re-write Program (9) to resolve $(\cdot)^+$, $\overline{R}_i$ and $\mathcal{X}$. To resolve $(\cdot)^+$, we introduce additional variable $z_i \in \mathbb{R}^*$ for $i = 1 \ldots N$, and constrain the inner term of $(\cdot)^+$ using z_i. To resolve $\overline{R}_i$, note that both Program (8) and Program (9) are minimization programs, and hence can be merged. $\mathcal{X}$ is resolved by bringing Eq. (1) into the program. We then arrive at the following Mixed Integer Linear Program (MILP):

$$\min \quad \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\overline{d}^i_{\text{sum}}}\sum_{a\in A}\overline{d}^i_a\sum_{v\in F}\text{dist}^i_a(v)t^i_a(v) + \rho\left(c + \frac{1}{1-\alpha}\frac{1}{N}\sum_{i=1}^{N}z_i\right) \quad (10a)$$

$$\text{s.t.} \quad \frac{1}{\overline{d}^i_{\text{sum}}}\sum_{a\in A}\overline{d}^i_a\sum_{v\in F}\text{dist}^i_a(v)t^i_a(v) - c \le z_i, \quad \forall i; \quad (10b)$$

$$\sum_{v} t^i_a(v) = 1, \quad \forall i, a; \quad (10c)$$

$$t^i_a(v) \le x_v, \qquad \forall i, a, v; \quad (10d)$$

$$\sum_{v\in F} c_v x_v \le b; \quad (10e)$$

$$x_v \in \{0,1\}, t^i_a(v) \in [0,1], z_i \ge 0, c \in \mathbb{R}, \ \forall i, a, v. \quad (10f)$$

In Program (10), note that the random variable distances $\text{dist}_a(v)$ are also instantiated by the deterministic distances $\text{dist}^i_a(v)$, which can be computed beforehand for all scenarios. There may be scenarios where an AP cannot reach all the fog nodes due to disconnectivity. In this case, we set the distance to the disconnected nodes to a large value, in order to prefer security function deployment at other fog nodes instead.

Program (10) can be solved using optimization solvers such as Gurobi [10]. Yet, there are two reasons that Program (10) is extremely hard to solve in practice. First, the program is non-convex due to integer variables x_v. Second, the program size is linear to N. To get a good approximation of the distributions of D and Y, the number of samples needed is commonly at least thousands. An MILP of such size is largely unsolvable in practice. Therefore, we resort to some optimization techniques, which can drastically reduce the complexity of solving Program (10) to a practical level.

C. Two-stage Optimization with Benders' Decomposition

At a closer look, Program (10) can be viewed as a typical two-stage optimization problem, with a small number of master variables but a huge number of slave variables. In the first stage, the program seeks to fix deployment plan X, which is called the *master problem*. In the second stage, given fixed deployment plan, it then computes the security risk of all scenarios by selecting the optimal fog nodes that actually deploy the security functions; this stage is called the *slave problem*. A difficulty in solving a two-stage program is that when fixing the first-stage master solution, there is no clue on how such decisions will affect the second-stage slave solution, until the slave problem is actually solved. A natural choice is thus an iterative algorithm that progressively solves both the master and the slave, until an optimal solution is found. In this subsection, we apply a well-known algorithm of such kind: the *Benders' decomposition* due to Benders [3].

Formally, Program (10) can be re-written as follows:

$$\min_{X,c} \quad \rho\cdot c + Q(X,c) \quad (11a)$$

$$\text{s.t.} \quad (10e),$$

$$x_v \in \{0,1\}, c \in \mathbb{R}, \qquad \forall v. \quad (11b)$$

where the slave problem $Q(X,c)$ is given by

$$Q(X,c) = \min_{z,t} \quad \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{\overline{d}^i_{\text{sum}}}\sum_{a\in A}\overline{d}^i_a\sum_{v\in F}\text{dist}^i_a(v)t^i_a(v) + \frac{\rho}{1-\alpha}z_i\right) \quad (12a)$$

$$\text{s.t} \quad (10b), (10c), (10d),$$

$$t^i_a(v) \in [0,1], z_i \ge 0, \quad \forall i, a, v. \quad (12b)$$

An intriguing property of the slave is that it is further decomposable regarding each scenario Π_i, as there are no coupling variables or constraints over i. Hence the slave problem can also be written as $Q(X,c) = \frac{1}{N}\sum_{i=1}^{N}Q_i(X,c)$, where $Q_i(X,c)$ is the slave subproblem for each scenario Π_i, defined by the inner term of the objective function and all the constraints for the specific i in Program (12).

To employ Benders' decomposition, we further need to study the dual program of $Q_i(X,c)$, defined as follows:

$$\Delta_i(x,c) = \max_{\lambda,\phi,\mu} \quad \sum_{a\in A}\left(\phi_i(a) - \sum_{v\in F}x_v\mu_i(a,v)\right) - c\cdot\lambda_i \quad (13a)$$

$$\text{s.t.} \quad \lambda_i \le \frac{\rho}{1-\alpha}; \quad (13b)$$

$$\phi_i(a) - \mu_i(a,v) \le \frac{\overline{d}^i_a\text{dist}^i_a(v)}{\overline{d}^i_{\text{sum}}}(1+\lambda_i), \forall a, v; \quad (13c)$$

$$\lambda_i, \mu_i(a,v) \ge 0, \phi_i(a) \text{ unbounded}, \quad \forall a, v. \quad (13d)$$

In Program (13), dual variables λ, ϕ and μ correspond to primal constraints (10b), (10c) and (10d), respectively.

Given the above, the key idea of Benders' decomposition is to, instead of considering all constraints as a whole, progressively add constraints (also called *cuts*) that may help in approaching the optimal solution in an iterative manner, thus avoiding to consider the large number of constraints together. The algorithm starts with a feasible master solution (*e.g.*, with all $x_v = 0$ and $c = 0$ in our problem), a pair of lower and upper bounds LB $= -\infty$ and UB $= \infty$, and the master problem that contains only the master constraints (in our case, Constraint (10e) is the only master constraint). In each iteration, the algorithm first solves the dual slave problem given the current master solution. It then updates the master problem by adding cuts based on the dual slave problem solution. The updated master problem is then solved

to optimality to obtain a new master solution. Note that although the master problem is still an MILP, it has a much smaller size compared to the original, and hence can be efficiently solved using a standard solver. The LB and UB are updated based on the master and slave solutions, respectively. The whole algorithm is shown in Algorithm 1.

Algorithm 1: Benders' Decomposition for Program (10)

Input: Network G, scenarios $\mathbf{\Pi}$, quantile α, tolerance ϵ
Output: Deployment plan X

1 $X \leftarrow \{x_v = 0 | \, v \in F\}$, $c \leftarrow 0$, LB $\leftarrow -\infty$, UB $\leftarrow \infty$;
2 **while** *UB* $-$ *LB* $> \epsilon$ **do**
3 Solve dual slave problem $\Delta_i(x, c)$ for $\forall \Pi_i$;
4 **if** $\Delta_i(x, c)$ *is unbounded* **then**
5 Get unbounded ray $(\tilde{\lambda}, \tilde{\phi}, \tilde{\mu})$;
6 Add *feasibility cut* to the master problem:
$$\sum_{i=1}^{N} \left(\sum_{a \in A} \left(\tilde{\phi}_i(a) - \sum_{v \in F} x_v \tilde{\mu}_i(a, v) \right) - c\tilde{\lambda}_i \right) \leq 0;$$
7 **else**
8 Get optimal point $(\lambda^*, \phi^*, \mu^*)$;
9 UB $\leftarrow \min\{\text{UB}, \Delta(x, c)\}$;
10 Add *optimality cut* to the master problem:
$$\sigma \geq \sum_{i=1}^{N} \left(\sum_{a \in A} \left(\phi_i^*(a) - \sum_{v \in F} x_v \mu_i^*(a, v) \right) - c\lambda_i^* \right);$$
11 Solve master problem $\min\{\sigma + \rho \cdot c \,|\, \text{cuts}, x \in \mathcal{X}\}$;
12 LB $\leftarrow \sigma + \rho \cdot c$;
13 **return** *X*.

In the algorithm, an unbounded ray $(\tilde{\lambda}, \tilde{\phi}, \tilde{\mu})$ in Line 5 is essentially a direction (in other words, a solution vector) to which the dual objective value goes to infinity. If the dual slave problem is unbounded, the primal slave problem is infeasible, and hence a *feasibility cut* is added in Line 6 to drive the master problem back into the feasible domain. If the dual slave has an optimal solution, the optimal dual point is incorporated into the master by adding an *optimality cut*, which drives the algorithm to search for master solutions with higher objective values. By adding cuts progressively, the algorithm avoids considering all the slave constraints together, but instead only considers those "promising" constraints that are likely to be active in the optimal solution. This can drastically reduce the overhead, especially for scenario-based optimization where a huge number of scenarios are considered. The UB is updated as the best feasible solution ever found, while the LB is updated when a new master solution is obtained. Optimality can be claimed when LB and UB converge.

D. Speeding-up Per-iteration Optimization

Even with the decomposition technique, Algorithm 1 is not efficient enough. The main reason is that it needs to solve a large number of dual slave linear programs (LPs) in each iteration, which is a slow process due to the cubical complexity for solving LPs [35]. In this subsection, we revisit the dual slave subproblems $\Delta_i(x, c)$ in (13), and show that they can be solved analytically due to their special structure.

Without loss of generality, we assume that the dual slave problem is feasible and bounded, *i.e.*, the corresponding primal slave problem is also feasible and bounded. Note that the primal problem is infeasible (hence the dual is unbounded) only when some AP is disconnected from any fog node in a certain scenario, which can be detected in the shortest path pre-computation phase. In the disconnected case, the *security risk* is ill-defined, as the AP cannot even communicate with the Internet. To tackle this, one way is to simply assign a uniform distance for all fog nodes, meaning that in this scenario it has no effect on the node selection. The primal problem cannot be unbounded (security risk is lower bounded by 0).

For scenario Π_i and AP a, let $v_a^i[1]$ and $v_a^i[2]$ be the two fog nodes with $x_v = 1$ and with the minimum distances from a, and let $\text{dist}_a^i[1]$ and $\text{dist}_a^i[2]$ be the corresponding distances respectively (with $\text{dist}_a^i[1] \leq \text{dist}_a^i[2]$). Also let $\delta_a^i = \overline{d}_a^i / \overline{d}_{\text{sum}}^i$. We then have the following optimal solution:

$$\lambda_i = \begin{cases} \dfrac{\rho}{1-\alpha} & \text{if } \sum\limits_a \delta_a^i \text{dist}_a^i[1] \geq c \\ 0 & \text{otherwise} \end{cases} \tag{14a}$$

$$\phi_i(a) = \delta_a^i \text{dist}_a^i[2](1 + \lambda_i) \tag{14b}$$

$$\mu_i(a, v) = \begin{cases} \delta_a^i (\text{dist}_a^i[2] - \text{dist}_a^i(v))(1 + \lambda_i) & \text{if } v = v_a^i[1] \\ & \text{or } x_v = 0 \\ 0 & \text{otherwise} \end{cases} \tag{14c}$$

The following theorem states the optimality of the above solution, whose proof is delegated to the appendix.

Theorem 1. *If the dual slave problem has bounded optimal value, Eq.* (14) *is an optimal solution to Program* (13)*, and can be computed in linear time.* □

V. Performance Evaluation

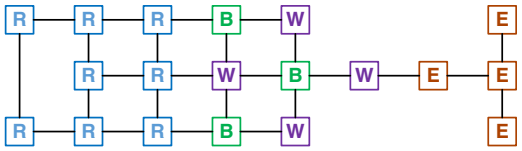
A. CVaR vs. Expectation

In this subsection, we show simulation results on comparing the expected case and the worst case up to a small tail probability. Our topology is in Fig. 1(a), which is based on the IoT framework in [21]. The topology had four types of nodes representing different street blocks: residential (R), work (W), business (B) and entertainment (E). Each node was both an AP and a fog node with uniform cost. The operator had a 30% budget over the sum of fog node costs. The scenarios were generated from a 3-month time period sliced into 15-minute intervals. Each link had an independent reliability of 99%. Depending on time and day of week, demand at each node was drawn from a Gamma distribution $\Gamma(a, b)$ where a is the shape and b is the scale, as shown in Table I.

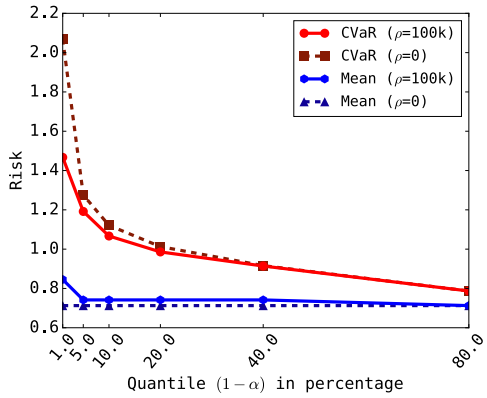
	Mon. – Fri.	Sat. & Sun.		Other (Sleep)
	8am–6pm	12pm–6pm	6pm–10pm	
R	$\Gamma(0.5, 1.5)$	$\Gamma(0.5, 1.5)$	$\Gamma(0.5, 0.5)$	$\Gamma(0.5, 3.0)$
W	$\Gamma(0.5, 2.0)$	$\Gamma(0.5, 0.2)$	$\Gamma(0.5, 0.2)$	$\Gamma(0.5, 0.8)$
B	$\Gamma(0.5, 0.5)$	$\Gamma(0.5, 2.0)$	$\Gamma(0.5, 1.0)$	$\Gamma(0.5, 0.3)$
E	$\Gamma(0.5, 0.2)$	$\Gamma(0.5, 0.5)$	$\Gamma(0.5, 2.5)$	$\Gamma(0.5, 0.1)$

TABLE I: Probability Distribution of Demands

We compared between using Algorithm 1 with a large enough ρ value (where only CVaR is considered) and with $\rho = 0$ (where only the expectation of risk is considered). We varied the quantile α to see the CVaR of different confidence levels. For example, a large quantile $\alpha = 99\%$ means we only look at the 1% most unfavorable scenarios, while $\alpha = 0\%$ means that we are looking at all the scenarios in CVaR, which by definition is equivalent to the expectation itself. To solve the MILP master problem, we used the Gurobi solver [10].



(a) Simulation topology



(b) CVaR and expectation with $\rho = 100$k (min CVaR) and $\rho = 0$ (min expectation) respectively

Fig. 1: Simulation on designated topology.

Fig. 1(b) shows the optimal solutions (with error tolerance of $\epsilon = 10^{-5}$, same hereinafter) obtained by Benders' decomposition. The quantile $(1-\alpha)$ is the percentage of scenarios included in the computation of CVaR. We can see that optimizing CVaR and optimizing expectation indeed achieves different solutions in most of the cases. With increase in $(1-\alpha)$, more scenarios are included in the CVaR, hence the CVaR value approaches the mean. However, with $(1-\alpha) = 1\%$, there is a great difference in minimizing CVaR ($\rho = 100$k) and minimizing expectation ($\rho = 0$), where the CVaR of security risk can be almost $1.5\times$ larger in the latter case than in the former. This suggests that in security systems where a few worst cases need to be carefully tackled, it is important to apply CVaR-aware optimization to account for the worst-case performance rather than for the average case alone.

B. Optimality and Efficiency

Next, we used randomly generated network to show how our proposed algorithm was able to achieve vast speed-up compared to the brute-force approach. Each topology had 30 nodes, and was generated using the Waxman model [9] with parameters $\alpha = \beta = 0.3$. We randomly picked half of the nodes to be APs, and ψ of the nodes to be fog nodes where $\psi \in [0, 1]$ is the *fog node ratio*. Deployment costs were generated uniformly from $[10, 100]$, while the operator had a budget of 50% of the sum of costs. In total 10k scenarios where generated for each experiment. Link reliability was uniformly 99%. The demands at APs were generated from an Erlang$(1, 2)$ distribution. We set quantile $\alpha = 95\%$. To average out noise of topology randomization, we generated 20 different topologies for each experiment setting, and took the average over all runs. Simulations were conducted on a Linux PC with 3.4GHz Quad-Core CPU and 16GB memory.

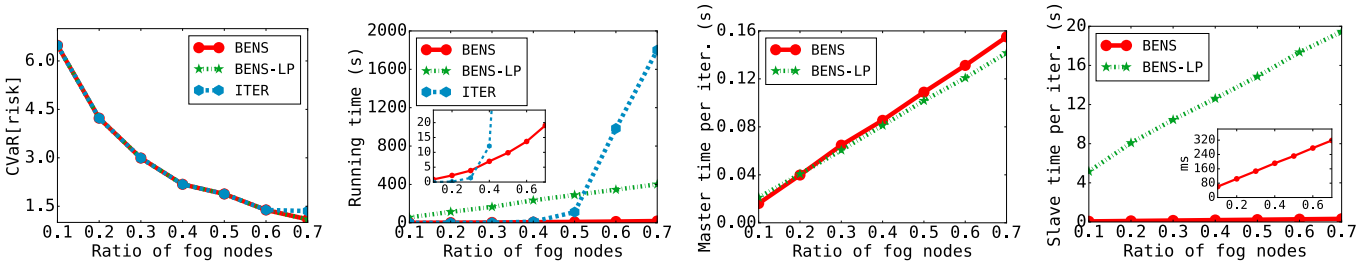
Fig. 2 shows the results of our algorithm over a brute-force algorithm. BENS is our proposed algorithm with our analytical model, while BENS-LP is our proposed algorithm with dual slave LPs solved by the Gurobi solver [10]. Brute-force algorithm ITER iterates over all possible deployment plans $X \in \mathcal{X}$, and returns the best solution found after all iterations or after a running time limit of 1800 seconds; ITER *returns the optimal solution unless exceeding the time limit*. Fig. 2(a) shows the optimality of our algorithm. Note that when the fog node ratio exceeds 0.7, ITER could not iterate over all possible solutions in the time limit, and hence returns suboptimal solutions. Fig. 2(b) shows the overall running times. Our BENS algorithm is the fastest in most cases. ITER is faster than BENS when the fog nodes are few, but then its time grows and quickly tops-up the pre-set time limit of 1800 seconds, due to the exponential increase in the solution space. Comparing BENS and BENS-LP, the former achieves a drastic speed-up. We further plot the average master and slave solving times per iteration in Figs. 2(c) and 2(d), respectively. It can be seen that BENS and BENS-LP do not differ much in master solving time. However, BENS achieves several orders of speed-up in slave solving time, further validating the effectiveness of our analytical model. Finally, an interesting finding is that, although the master problem is an MILP, it actually solves faster than the slave problem, due to its small size compared to the large number of scenarios.

C. Comparison with Fast Baseline Heuristics

In the last set of experiments, we show the performance of our algorithm compared to fast heuristics. For this comparison, we used a synthesized dataset derived from real IoT device traces collected in the Dartmouth College [15]. The original dataset contained the location data of over 600 APs and the connectivity data of over 13000 user devices for more than 5 years. We synthesized the dataset as follows. First, we aggregated APs based on buildings, and regarded each building as a single AP. Second, we further divided buildings into street blocks based on a publicly available map of the campus. Due to lack of campus network topology, we adopted a topology where each building's AP is directly connected to a central node within each block, and all blocks are connected in a ring topology. All block nodes and 10% of the buildings were selected as fog nodes, with deployment costs of 100 and 10, respectively. Finally, since the entire data set contains many intervals that have unstable and erroneous measurements, we only used a one-year subset of data (09/2002–08/2003) that is relatively stable according to the collectors [15].

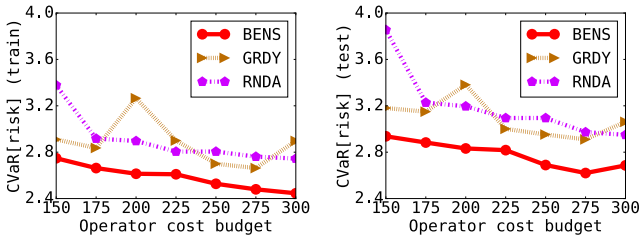
To characterize the demand distribution, we regarded the user connectivity data as samples from the underlying realistic distribution. We used 4 months of data (09/2002–12/2002) as training set, and the next 8 months of data (01/2003–08/2003) as the test set, both sliced into intervals of 15-minute length. We used the training data to deploy the security functions, and then calculated the security risks of the deployed security functions on both the training set and the test set. The number of devices were aggregated and averaged over every 15-minute interval for each building AP. All links had 99% reliability. We again set ρ to a large value, and let $\alpha = 95\%$.

Fig. 3 shows the training and testing security risks using the synthesized data. We compared our proposed algorithm to two heuristics: RNDA and GRDY. RNDA is a random algorithm which randomly picks fog nodes to deploy the security function, until the budget is exceeded. GRDY is a greedy algorithm which iteratively selects fog nodes that result



(a) CVaR vs. fog nodes (b) Running time vs. fog nodes (c) Master time vs. fog nodes (d) Slave time vs. fog nodes

Fig. 2: Simulation results with varying number of fog nodes over total nodes. ITER (optimal algorithm) exceeds the time limit (1800 s) when the fog node ratio is 0.7, and hence is terminated before finishing all iterations.



(a) Training set performance (b) Test set performance

Fig. 3: CVaR of risk on synthesized Dartmouth data.

in the maximum reduction in the objective function, until the budget is exceeded. From the figure, we can see that our algorithm outperforms both GRDY and RNDA. The greedy heuristic generally achieves better performance than RNDA, but may result in poor performance with certain budget values such as a budget of 200. Also, it is interesting to see that an optimal training solution may not be optimal on the test set. For example, when the budget increases from 275 to 300, the training CVaR of BENS (the optimal) decreases, but the test CVaR increases. This is commonly due to changes in the underlying distribution, and should be tackled by periodically re-optimizing the deployment decisions based on new inputs.

VI. Discussions and Future Work

Advanced deployment and risk measurement: Our current distance-based risk measure can be extended to more complex cases. For example, each link may have a routing capacity and each security function may have a processing capacity in practice. In this case, demands may be split over multiple paths and/or security functions if some are overloaded. Also, different links and/or nodes may have different contributions to the security risks, as some may be more vulnerable to others. These considerations can mostly be integrated with minimal modifications into our CVaR-based and scenario-based optimization framework. Derivation of efficient solutions in these more complex cases is delegated to our future work.

Further optimization speed-ups: The proposed algorithm uses the basic Benders' decomposition, with the only speedup being the problem-specific analytical model for the dual slaves. Other techniques may also improve the efficiency of Benders' decomposition, such as enhanced cuts [7], the three-phase method [7], trust regions [18], etc. While these techniques can hardly beat the speed-up achieved by our analytical model (and most of them are not trivially compatible with our analytical model), they are helpful in general when our model and framework are extended to other more complex cases as aforementioned. An orthogonal technique is to employ parallelization in each iteration, which is natural through the dual slave decomposition in our algorithm.

VII. Conclusions

In this paper, we studied the security risk associated with offloading security from IoT devices to fog or cloud nodes in the network. To maximize system security and robustness, the operator would want to deploy in-network security functions to minimize the security risk of all users, given various scenarios including varying demands and network failures. We made the following contributions. First, we proposed a stochastic model for uncertainties in IoT. Second, we used an economic model (CVaR) to capture the worst-case security risk of the system, in addition to the conventional expectation-based model. Third, we developed a decomposition-based optimization framework for optimizing both the expected security risk and the its CVaR in scenario-based stochastic programming. We then enhanced the framework with an analytical model tailored to drastically reduce its optimization overhead. Finally, we showed, through simulations, that the proposed model well captures system security risk up to a small tail probability, and that the proposed framework achieves optimal security deployment with limited overhead compared to other algorithms.

Acknowledgement

This research was supported in part by NSF grants 1461886, 1704092, and 1717197. The reported information does not reflect the position or policy of the funding agency.

References

[1] A. Lioy, A. Pastor, F. Risso, R. Sassu, and A. L. Shaw, "Offloading Security Applications into the Network," *Proc. IEEE eChallenges*, 2014.

[2] H. M. Almohr, L. T. Watson, D. D. Yao, and X. Ou, "Security Optimization of Dynamic Networks with Probabilistic Graph Modeling and Linear Programming," Tech. Rep., 2014.

[3] J. F. Benders, "Partitioning Procedures for Solving Mixed-Variables Programming Problems," *Numer. Math.*, vol. 4, no. 1, pp. 238–252, dec 1962.

[4] Z. Cao, M. Kodialam, and T. V. Lakshman, "Traffic Steering in Software Defined Networks: Planning and Online Routing," in *Proc. ACM DCC*, 2014, pp. 65–70.

[5] Y. Chang, S. Rao, and M. Tawarmalani, "Robust Validation of Network Designs under Uncertain Demands and Failures," in *Proc. USENIX NSDI*, 2017, pp. 347–362.

[6] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou, "New Algorithms for Secure Outsourcing of Modular Exponentiations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 9, pp. 2386–2396, sep 2014.

[7] J.-F. Cordeau, G. Stojković, F. Soumis, and J. Desrosiers, "Benders Decomposition for Simultaneous Aircraft Routing and Crew Scheduling," *Transp. Sci.*, vol. 35, no. 4, pp. 375–388, nov 2001.

[8] R. Dewri, N. Poolsappasit, I. Ray, and D. Whitley, "Optimal Security Hardening Using Multi-Objective Optimization on Attack Tree Models of Networks," in *Proc. ACM CCS*, 2007, p. 204.

[9] M. Faloutsos, C. Faloutsos, and C. Faloutsos, "On Power-Law Relationships of the Internet Topology," in *Proc. ACM SIGCOMM*, 1999, pp. 251–262.

[10] Gurobi, "Gurobi Optimizer." URL: http://www.gurobi.com/products/gurobi-optimizer

[11] D. S. Hochbaum, "Heuristics for the Fixed Cost Median Problem," *Math. Program.*, vol. 22, no. 1, pp. 148–162, dec 1982.

[12] J. Horn, *et. al*, "Meltdown and Spectre." URL: https://spectreattack.com

[13] W. Jung, S. Hong, M. Ha, Y.-J. Kim, and D. Kim, "SSL-Based Lightweight Security of IP-Based Wireless Sensor Networks," in *Proc. IEEE WAINA*, 2009, pp. 1112–1117.

[14] M. S. Kiraz and O. Uzunkol, "Efficient and Verifiable Algorithms for Secure Outsourcing of Cryptographic Computations," *Int. J. Inf. Secur.*, vol. 15, no. 5, pp. 519–537, oct 2016.

[15] D. Kotz, T. Henderson, I. Abyzov, and J. Yeo, "CRAWDAD Dataset Dartmouth/Campus (V. 20090909)." URL: https://crawdad.org/dartmouth/campus/20090909

[16] K. Kumar and Yung-Hsiang Lu, "Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?" *Computer (Long. Beach. Calif).*, vol. 43, no. 4, pp. 51–56, apr 2010.

[17] J.-j. Kuo, S.-h. Shen, H.-y. Kang, D.-n. Yang, M.-j. Tsai, and W.-t. Chen, "Service Chain Embedding with Maximum Flow in Software Defined Network and Application to the Next-Generation Cellular Network Architecture," in *Proc. IEEE INFOCOM*, 2017.

[18] S. J. Maher, G. Desaulniers, and F. Soumis, "Recoverable Robust Single Day Aircraft Maintenance Routing Problem," *Comput. Oper. Res.*, vol. 51, pp. 130–145, nov 2014.

[19] C. Manifavas, G. Hatzivasilis, K. Fysarakis, and K. Rantos, "Lightweight Cryptography for Embedded Systems – A Comparative Analysis," in *Data Priv. Manag. Auton. Spontaneous Secur.*, 2014, pp. 333–349.

[20] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici, "ClickOS and the Art of Network Function Virtualization," in *Proc. USENIX NSDI*, 2014, pp. 459–473.

[21] H. Ning and Z. Wang, "Future Internet of Things Architecture: Like Mankind Neural System or Social Organization Framework?" *IEEE Commun. Lett.*, vol. 15, no. 4, pp. 461–463, apr 2011.

[22] S. Noel and S. Jajodia, "Optimal IDS Sensor Placement and Alert Prioritization Using Attack Graphs," *J. Netw. Syst. Manag.*, vol. 16, no. 3, pp. 259–275, sep 2008.

[23] N. Poolsappasit, R. Dewri, and I. Ray, "Dynamic Security Risk Management Using Bayesian Attack Graphs," *IEEE Trans. Dependable Secur. Comput.*, vol. 9, no. 1, pp. 61–74, jan 2012.

[24] R. T. Rockafellar and S. Uryasev, "Optimization of Conditional Value-at-Risk," *J. Risk*, vol. 2, pp. 21–41, 2000.

[25] M. Rost and S. Schmid, "Service Chain and Virtual Network Embeddings: Approximations Using Randomized Rounding," *arXiv:1604.02180*, 2016.

[26] A. Rullo, E. Serra, E. Bertino, and J. Lobo, "Shortfall-Based Optimal Placement of Security Resources for Mobile IoT Scenarios," in *Proc. ESORICS*, 2017, pp. 419–436.

[27] A. Sajid, H. Abbas, and K. Saleem, "Cloud-Assisted IoT-Based SCADA Systems Security: A Review of the State of the Art and Future Challenges," *IEEE Access*, vol. 4, pp. 1375–1384, 2016.

[28] R. E. Sawilla and X. Ou, "Identifying Critical Attack Assets in Dependency Attack Graphs," in *Proc. ESORICS*, 2008, pp. 18–34.

[29] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, M. Tyson, A. Texas, C. Station, and M. Park, "FRESCO: Modular Composable Security Services for Software-Defined Networks," in *Proc. USENIX NDSS*, 2013.

[30] S. Subashini and V. Kavitha, "A Survey on Security Issues in Service Delivery Models of Cloud Computing," *J. Netw. Comput. Appl.*, vol. 34, no. 1, pp. 1–11, jan 2011.

[31] W. Trappe, R. Howard, and R. S. Moore, "Low-Energy Security: Limits and Opportunities in the Internet of Things," *IEEE Secur. Priv.*, vol. 13, no. 1, pp. 14–21, jan 2015.

[32] J. Wallen, "Five Nightmarish Attacks That Show the Risks of IoT Security." URL: http://www.zdnet.com/article/5-nightmarish-attacks-that-show-the-risks-of-iot-security/

[33] L. Wu, M. Shahidehpour, and T. Li, "Stochastic Security-Constrained Unit Commitment," *IEEE Trans. Power Syst.*, vol. 22, no. 2, pp. 800–811, may 2007.

[34] Y. Xia, Y. Liu, C. Tan, M. Ma, H. Guan, B. Zang, and H. Chen, "TinMan: Eliminating Confidential Mobile Data Exposure with Security Oriented Offloading," in *Proc. ACM EuroSys*, 2015, pp. 1–16.

[35] Y. Ye, "An O(n3L) Potential Reduction Algorithm for Linear Programming," *Math. Program.*, vol. 50, no. 1-3, pp. 239–258, mar 1991.

[36] N. Zhang, S. Demetriou, X. Mi, W. Diao, K. Yuan, P. Zong, F. Qian, X. Wang, K. Chen, Y. Tian, C. A. Gunter, K. Zhang, P. Tague, and Y.-H. Lin, "Understanding IoT Security Through the Data Crystal Ball: Where We Are Now and Where We Are Going to Be," *arXiv:1703.09809*, 2017.

[37] J. Zhou, Z. Cao, X. Dong, and A. V. Vasilakos, "Security and Privacy for Cloud-Based IoT: Challenges," *IEEE Commun. Mag.*, vol. 55, no. 1, pp. 26–33, jan 2017.

APPENDIX

Proof of Theorem 1: For simplicity, we omit subscript i, since the dual slave subproblem is independent for each scenario. We call a node *active* if it has $x_v = 1$ in the current iteration, otherwise it is *inactive*. First, for any inactive node v, we can assume that $\mu(a, v)$ takes an arbitrarily large value to enforce Constraint (13c), as it has a zero coefficient in the objective function. For each AP a, a node v can have a positive $\mu(a, v)$ only when the corresponding Constraint (13c) is binding, *i.e.*, equality holds instead of inequality at any optimal point, for this node; otherwise, the objective value can be solely increased by decreasing $\mu(a, v)$ without violating Constraint (13c). We claim that there is at most one active node v with positive $\mu(a, v)$; if multiple active nodes have positive $\mu(a, v)$, then we can reduce $\phi(a)$ along with all the positive $\mu(a, v)$s by a small amount, reducing objective value without violating Constraint (13c) for any node. Further, we claim that the node with positive $\mu(a, v)$ must be the one and only one active node v^* who has the minimum distance from a; if more than one node has the same minimum distance, they must all have $\mu(a, v) = 0$. Otherwise, say if an active node v' with non-minimum distance has a positive $\mu(a, v')$ and the corresponding Constraint (13c) is binding, then clearly $\mu(a, v^*) > \mu(a, v') > 0$, because $\text{dist}_a(v^*) < \text{dist}_a)(v')$. Then we have two positive $\mu(a, v)$ values, which conflicts with the above. In the case of a single minimum-distance active node $v^* = v_a^i[1]$, both $\phi(a)$ and $\mu(a, v^*)$ can take positive values up to the values specified in Eq. (14), without violating Constraint (13c) for the second minimum-distance node, $v_a^i[2]$. We pick the upper bounds to motivate the master problem to search for new solutions in each iteration.

Now, for the inactive nodes, though they can take arbitrarily large values, we pick their lower bounds, such that reducing any μ value in this class will lead to constraint violation. This gives Eq. (14) for $x_v = 0$. This choice is for simplicity of the equation, but can also help in numerical stability in practice.

Based on the above, we know that in the optimal solution, the first term of the objective function is always equal to $\xi(1+\lambda)$ where $\xi = \sum_a \delta_a \text{dist}_a[1]$. Then, it is clear that the λ value is based on the comparison between ξ and c. If $\xi \geq c$, we set λ to the maximum value $\frac{\rho}{1-\alpha}$ as bounded by Constraint (13b); otherwise, we set λ to 0. This completes the proof. ■