A Firewall of Two Clouds: Preserving Outsourced Firewall Policy Confidentiality with Heterogeneity

Lingbo Wei*†‡, Chi Zhang*, Yanmin Gong§, Yuguang Fang§, and Kefei Chen¶‡
*CAS Key Laboratory of Electromagnetic Space Information
University of Science and Technology of China, Hefei 230027, P. R. China
†State Key Laboratory of Information Security, Institute of Information Engineering
Chinese Academy of Sciences, Beijing 100093, P. R. China
‡Science and Technology on Communication Security Laboratory, Chengdu 610041, P. R. China
§ECE Department, University of Florida, Gainesville, Florida 32611, USA
¶School of Science, Hangzhou Normal University, Hangzhou 310036, P. R. China
Email: {lingbowei, chizhang}@ustc.edu.cn, {ymgong@, fang@ece.}ufl.edu, kfchen@sjtu.edu.cn

***Abstract*—It is increasingly common for enterprises and other organizations to outsource firewalls to public clouds in order to reduce the cost and complexity in deploying and maintaining dedicated hardware middleboxes. However, this poses a serious threat to the enterprise network security because sensitive network policies, such as firewall rules, are revealed to cloud providers, which may be leaked and exploited by attackers. In this paper, we design and implement a SE-FWaaS, a secured system that enables cloud providers to support middlebox (e.g., firewall) outsourcing while preserving the network policy confidentiality. The key ingredients in our SE-FWaaS are the distribution of the firewall primitives, namely policy checking and verdict enforcing, to two independent public clouds, and the enabling techniques of efficient firewall rule obfuscation and oblivious rule-matching. Our SE-FWaaS provides the maximum achievable level of protection of network policies by enforcing the principle of the least privilege and removing the threat of offline probing attacks. We evaluate the proposed system over real-world firewall rules and demonstrate its effectiveness and feasibility.**

***Keywords*—Firewall; Network Function Outsourcing; Middlebox Outsourcing; Cloud Computing; Security and Privacy**

I. Introduction

Today's private networks of homes, businesses and institutions rely on a wide spectrum of *network functions* (or *middleboxes*) to improve network security (e.g., firewalls and intrusion detection systems), provide better performance (e.g., proxies or load balancers), and reduce bandwidth costs (e.g., WAN optimizers). Recent studies [1] show that roughly one in three network devices in an enterprise network is a middlebox that inspects, transforms or modifies packets instead of simply forwarding packets as normal routers or switches. However, traditional hardware-based in-house middleboxes also bring significant problems including high cost, inflexibility, and complex management. They incur significant capital investment due to peak-demand provision, are cumbersome to maintain because of expert knowledge required, and cannot be easily extended to accommodate new functionalities when new operational requirements emerge.

With the rise of public cloud computing paradigm, enterprises have started to explore an alternate approach: outsourcing dedicated middleboxes from their private networks entirely to the public cloud, and hence promising many of the well-known benefits of cloud computing such as decreased costs and ease of deployment and management. In this new outsoucing paradigm - termed *network function outsourcing* (NFO) [2], [3] or *network function virtualization* (NFV) [4] in industry - middleboxes are run on virtual machines (VMs) built from commodity server hardware in a public cloud, operating as software processes to provide required network functions on a pay-per-use basis. Now NFO has gained a significant momentum with over 270 industry participants and several successful product offerings [4].

Despite the growing adoption of NFO techniques, serious security concerns are raised because the NFO in its current setting requires customers fully trust the cloud provider and reveal all the detailed network policies, which instruct how network functions are to be performed, to the cloud provider. More specifically, the correctness of current NFO systems is based on two strong security assumptions. Firstly, the remote software middlebox (and therefore the public cloud) is assumed to be honest in the sense that it faithfully work as intended [5]. Secondly, the public cloud is assumed not to be curious, in the sense that it does not collect or leak information on customer's network policies. Unfortunately, the principle of *designed-in security* requires that such assumptions cannot be made and instead all participants in the system should be considered as potentially malicious [6].

To keep the discussion concrete, in this paper we use firewall service as an exemplary network function to illustrate the methodology of securing NFO. Firewalls monitor and control the incoming and outgoing network traffic based on predetermined network policies, and are the cornerstone of today's network security. A rational NFO customer must adopt a *trust-but-verify* strategy, i.e., she must be able to verify that the outsourced firewall applies the intended network policies

This work was supported in part by the Natural Science Foundation of China (NSFC) under Grants 61202140 and 61328208, by the Program for New Century Excellent Talents in University under Grant NCET-13-0548, and by the Fundamental Research Funds for the Central Universities under Grand WK2101020006. The work of Y. Fang was also partially supported by the U.S. National Science Foundation under grants CNS-1409797 and CNS-1343356. The work of K. Chen was partially supported by NSFC under Grants 61133014, 61472114 and STCSL under Grant 9140C110203140C11049.

978-1-5090-1328-9/16/$31.00 ©2016 IEEE

on incoming and outgoing packets, just like it running on a dedicated middlebox inside the customer's network under her direct control. We call this the "*verifiable policy enforcement problem*." We note that this problem has been extensively studied in the literature (see [5] for an example) and therefore is not within the scope of this paper.

In this paper, we try to address a more challenging "*policy confidentiality problem*," i.e., how to design a firewall outsourcing scheme while preserving firewall policy confidentiality. Firewall policies often contain sensitive information (e.g., IP addresses of hosts, network topology and defense strategies) and often have uncovered security vulnerabilities that can be exploited by attackers. Therefore, in practice, even within the same organization, often no employees other than the firewall administrator are allowed to access the firewall policies. To expose this information to the public cloud will bring high security risk because even the cloud can be trusted as an organization, information leakage through employee theft is still a serious concern [7].

We tackle the above problem by leveraging the features of *heterogeneous cloud* environment. More specifically, we assume each customer can utilize two kinds of independent public clouds. The *telecom clouds* which are operated by telecommunication companies can offer cloud services with higher communication qualities because they are closer to customers and can utilize the existing network infrastructures controlled by themselves. Normally the operators of telecom clouds are also the *Internet service providers* (ISPs). Larger cloud service providers such as Amazon and Google, which are called *IT service cloud* providers, normally have much richer computation and storage resources and can provide customers with cheaper cloud services. In general, we can safely assume these two kinds of cloud providers are independent and have no incentive to collude. The customers will combine the advantages of both clouds and achieve a higher performance/cost ratio especially in mobile cloud settings [8].

In this paper, we present the design and implementation of our **SE-FWaaS**, a *Security-Enhanced FireWall as a Service* that utilizing two independent clouds to provide outsourced firewall services while preserving firewall policy confidentiality from the clouds. What distinguishes this work from the previous NFO security design is the following. First, we limit information leakage by distributing the firewall's basic primitives, namely *policy checking* and *verdict enforcing*, to the IT service cloud and the telecom cloud respectively. Secondly, we enforce the *principle of the least privilege*, which applied here states that the clouds providing firewall services should have the lowest amount of privileges they need in order to do their work. For example, at the *policy-checking point* (PCP) in an IT service cloud, we obfuscate the firewall rules and perform oblivious rule-matching. Therefore, the PCP does not know which rule matches which packet; thus it makes the selective policy updating attacks infeasible. The PCP can only observe the inbound and outbound traffic. The information gained by the *verdict enforcement point* (VEP) in a telecom cloud does not differ significantly from that of a regular ISP. Thirdly, most NFO security schemes proposed in the literature are vulnerable to offline probing attack, i.e., the hosting cloud can generate dummy traffic and probe and learn the firewall ruleset through brute-force trial and error. In our scheme, by decoupling firewall functionality into two steps and placing them separately in two independent clouds, it is impossible for each of the clouds to launch probing attack independently and successfully. Therefore, we postulate that our **SE-FWaaS** provides the maximum achievable level of protection of customer's network policies in NFO scenarios.

The rest of the paper proceeds as follows. First we review related work in Section II. We then introduce our system model and trust assumptions in Section III. In section IV we present our scheme **SE-FWaaS** in detail and the performance of our scheme is analysed in Section V.

II. Related Work

Khakpour and Liu [9] present *Ladon*, the first framework in the literature attempting to preserve cloud-based firewall policy confidentiality. The basic idea is to convert the original access policy to an Firewall Decision Diagram (FDD) and anonymize the FDD with Bloom filters. The limitation of *Ladon* is that it cannot prevent the public cloud from deducing the original firewall policy by offline probing or just by traffic eavesdropping and analysis. To address this problem, *Ladon Hybrid Cloud* is proposed to augment *Ladon* by leveraging a hybrid cloud model (i.e., a combination of public and private clouds) [10]. The drawback of this remedy approach is that the customer still needs to maintain a firewall in her private cloud. In our **SE-FWaaS**, the firewall is entirely outsourced to public clouds and therefore we can take the full advantage of NFO. Furthermore, the security of these two schemes are based on the one-wayness assumption of Bloom filter, which may not be true in practice [11].

Shi, Zhang, and Zhong [12] propose the framework *SOFA* which utilizing cryptographic multilinear map to obfuscate the original firewall rules. The cloud provider can perform the firewall functionality by filtering inbound and outbound traffic with this configured obfuscated firewall but cannot recover the original firewall rules. The construction scheme of multilinear map adopted in *SOFA* is from Coron, Lepoint and Tibouchi (CLT) [13], which has been found recently to be insecure [14]. Also *SOFA* is vulnerable to offline probing attacks.

The work of Melis et al. [15] considers not only the firewall outsourcing but also other network functions such as load balancer, carrier-grade NAT, intrusion detection system, and deep packet inspection. Based on different trust assumptions on VMs on the public cloud servers, the authors of [15] propose two solutions by utilizing partial homomorphic encryption and public-key encryption with keyword search (PEKS). The drawback of this scheme is that it cannot outsource the entire network function to the cloud, and the customer still requires her own middlebox to carry out the remaining network function. Note that when outsourcing firewall to the cloud, the cloud middlebox has to send all the results of every rule in the firewall to the client middlebox due to the fact that cloud middlebox does not know which rule the packet matches. This increases communication overhead because the results to be transmitted increase linearly with the number of firewall rules.

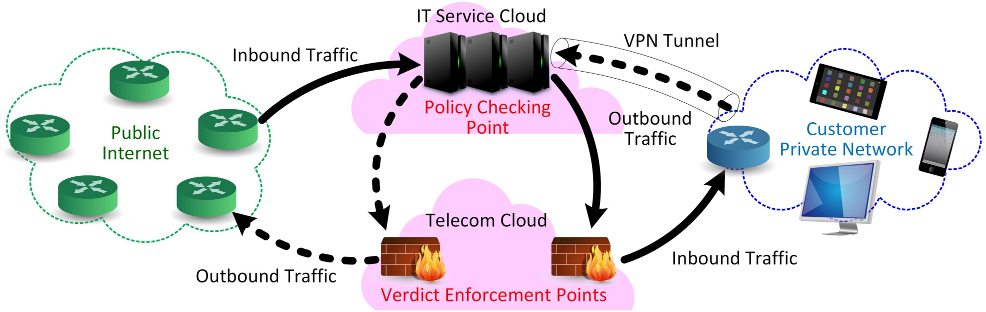

Fig. 1. The System Settings for SE-FWaaS

Moreover, the client middlebox has to decrypt the results one by one until the client middlebox finds that the packet matches a rule and this increases the computation cost.

III. System Model and Trust Assumptions

A. System Model

We consider a scenario where an enterprise, the customer private network, outsources its firewall services to two independent clouds as illustrated in Figure 1. The outsourced firewall functionalities run within VMs on the cloud servers, and are called *policy checking point* (PCP) and *verdict enforcement point* (VEP) in this scenario. The PCP is installed in an IT service cloud, as it requires more computation resources while the VEP in a telecom cloud with more networking resources. This scenario is realistic in the sense that two kinds of clouds have their own advantages, coexist in the market, and show a strong complementarity. Rational customers can achieve better performance/cost ratio by combining the utilization of both kinds of cloud resources. We illustrate the two redirection setups for supporting NFO in Figure 1. The PCP receives inbound traffic destined for the customer, processes the policy-checking function assigned to it, and forward the checking results and related inbound traffic to the VEP which processes the verdict enforcing function. Outbound traffic is the one originating from the customer private network which is forwarded to the PCP through a VPN channel, processed by the PCP and VEP sequentially, and finally relayed to its intended destination in the Internet. The obfuscated network policies which describe how the network functions are to be processed are installed in the PCP and VEP by the customer.

A typical IP firewall is a packet filter which looks at the network addresses, ports and protocol type of the packet and determines if that packet should be allowed or blocked. A firewall rule is composed of 5-tuple (source IP, source port, destination IP, destination port, protocol type) and corresponding action (verdict), as shown in Table I. In [9], [12], only one public cloud is involved and knows the corresponding action of a firewall rule. Although this cloud does not know what the obfuscated firewall really is, it knows how to handle the packet when finding the packet matches an obfuscated firewall rule. In our **SE-FWaaS**, we split the firewall function into two independent clouds. The PCP does not know the action of a firewall rule and is only responsible for the policy checking. The PCP relays the traffic and the corresponding policy checking results to the VEP. The VEP reveals the matching results and processes the traffic accordingly.

TABLE I
Firewall Rule Examples

Rule	Source		Destination		Proto	Action
ID	IPsrc	srcPort	IPdst	dstPort	type	
1	192.168.*.*	*	*	*	*	Block
2	192.168.*.*	*	202.38.64.1	23	TCP	Block
3	202.38.*.*	8090	*	*	UDP	Allow
4	10.*.*.*	*	115.25.*.*	8080	TCP	Block

Throughout the paper, for each rule in the firewall we denote it as $r = (\overrightarrow{v}, W, A)$, where $\overrightarrow{v} \in \{0,1\}^n$ is the bitwise representation of the field values in each rule. The i-th bit of $\overrightarrow{v}$, i.e. $\overrightarrow{v}[i]$, is 0 or 1 as expected if $i \notin W$ (i.e., the *non-wildcard bits*) and the set $W \subseteq [k]$ represents the *wildcard bits* (or *don't care bits*). Note that the set W is often the subnet mask for IP addresses. We use A to denote the corresponding action (allow/block). For example, the rule 1 in Table I "192.168. $*$.$*$, $\cdots$" can be presented as $\overrightarrow{v} =$ "11000000 10101000 00000000 $\cdots$", $W = \{17, 18, \cdots, 32\}$, and $A = \{block\}$. Similarly, for a packet's header, we also present it as an n-bit vector $\overrightarrow{p} \in \{0,1\}^n$, with $\overrightarrow{p}[i] = 0$ or 1.

B. Trust Assumptions

In the traditional outsource setting, network functions are run on dedicated hardware middleboxes located within the customer private network. As a result, the network policies are hidden from outsiders as long as the hardware is secure. Once a network function is outsourced to the cloud as a software, obviously, it is no longer the case. Ideally, the customer would want its network policies to remain confidential while maintaining the same standards of networking services.

We assume the PCP and VEP to be honest-but-curious, i.e., they performs network functions dutifully yet wishes to infer the customer's policies. The PCP or VEP may intercept and analyze traffic and try to infer network policies based on the pattern of inbound and outbound packets. Likewise, the PCP or VEP may generate its own traffic destined for the customer and analyze the packets it receives in response (i.e., offline probing attacks). Note that we do not need to provide any confidentiality beyond what can be achieved in the traditional setting (when firewalls are set within the customer private network). Furthermore, we assume that the PCP and VEP will

TABLE II
A RUNNING EXAMPLE OF SE-FWAAS

To ease presentation, we assume that there are $m = 3$ rules and only one header field with $n = 4$ bits. The rules are only known to the customer.
◇ Rule 1: $r_1 = (\overrightarrow{v}_1, W_1, A_1) = ((1, 0, 1), \{\}, \{allow\})$;
◇ Rule 2: $r_2 = (\overrightarrow{v}_2, W_2, A_2) = ((1, 1, 1), \{\}, \{block\})$;
◇ Rule 3: $r_3 = (\overrightarrow{v}_3, W_3, A_3) = ((0, 1, *), \{3\}, \{block\})$.

Policy Obfuscating and Decoupling:

The customer obfuscates each rule and constructs a new firewall $\left\{r'_j\right\}_{j\in[3]}$ as follows:

r'_j	$(u_{0,1,j}, v_{0,1,j})$	$(u_{0,2,j}, v_{0,2,j})$	$(u_{0,3,j}, v_{0,3,j})$	$u_{4,j}$	
	$(u_{1,1,j}, v_{1,1,j})$	$(u_{1,2,j}, v_{1,2,j})$	$(u_{1,3,j}, v_{1,3,j})$	$\boxed{v_{4,j}}$	$\boxed{A_j}$
r'_1	$\left(g^{\alpha_{0,1,1}}, \beta_{0,1,1} h_x^{\alpha_{0,1,1}}\right)$	$\left(g^{\alpha_{0,2,1}}, \beta_{0,2,1} h_x^{\alpha_{0,2,1}}\right)$	$\left(g^{\alpha_{0,3,1}}, \beta_{0,3,1} h_x^{\alpha_{0,3,1}}\right)$	$g^{\alpha_{4,1}}$	
	$\left(g^{\alpha_{1,1,1}}, \beta_{1,1,1} h_x^{\alpha_{1,1,1}}\right)$	$\left(g^{\alpha_{1,2,1}}, \beta_{1,2,1} h_x^{\alpha_{1,2,1}}\right)$	$\left(g^{\alpha_{1,3,1}}, \beta_{1,3,1} h_x^{\alpha_{1,3,1}}\right)$	$\boxed{\beta_{1,1,1}\beta_{0,2,1}\beta_{1,3,1} \cdot h_y^{\alpha_{4,1}}}$	$\boxed{allow}$
r'_2	$\left(g^{\alpha_{0,1,2}}, \beta_{0,1,2} h_x^{\alpha_{0,1,2}}\right)$	$\left(g^{\alpha_{0,2,2}}, \beta_{0,2,2} h_x^{\alpha_{0,2,2}}\right)$	$\left(g^{\alpha_{0,3,2}}, \beta_{0,3,2} h_x^{\alpha_{0,3,2}}\right)$	$g^{\alpha_{4,2}}$	
	$\left(g^{\alpha_{1,1,2}}, \beta_{1,1,2} h_x^{\alpha_{1,1,2}}\right)$	$\left(g^{\alpha_{1,2,2}}, \beta_{1,2,2} h_x^{\alpha_{1,2,2}}\right)$	$\left(g^{\alpha_{1,3,2}}, \beta_{1,3,2} h_x^{\alpha_{1,3,2}}\right)$	$\boxed{\beta_{1,1,2}\beta_{1,2,2}\beta_{1,3,2} \cdot h_y^{\alpha_{4,2}}}$	$\boxed{block}$
r'_3	$\left(g^{\alpha_{0,1,3}}, \beta_{0,1,3} h_x^{\alpha_{0,1,3}}\right)$	$\left(g^{\alpha_{0,2,3}}, \beta_{0,2,3} h_x^{\alpha_{0,2,3}}\right)$	$\left(g^{\alpha_{0,3,3}}, \beta_{3,3} h_x^{\alpha_{0,3,3}}\right)$	$g^{\alpha_{4,3}}$	
	$\left(g^{\alpha_{1,1,3}}, \beta_{1,1,3} h_x^{\alpha_{1,1,3}}\right)$	$\left(g^{\alpha_{1,2,3}}, \beta_{1,2,3} h_x^{\alpha_{1,2,3}}\right)$	$\left(g^{\alpha_{1,3,3}}, \beta_{3,3} h_x^{\alpha_{1,3,3}}\right)$	$\boxed{\beta_{0,1,3}\beta_{1,2,3}\beta_{3,3} \cdot h_y^{\alpha_{4,3}}}$	$\boxed{block}$

Then the customer separates r'_j into two parts, i.e. $r'_{j,\text{PCP}}$ and $r'_{j,\text{VEP}}$, and sends them to the PCP and VEP respectively.
◇ Note that in the above table the boxed items belong to the set $\{r'_{j,\text{VEP}}\}_{j\in[3]}$, and other items belong to the set $\{r'_{j,\text{PCP}}\}_{j\in[3]}$.

Policy Checking:

For the newly arrived packet with header $\overrightarrow{p} = (0, 1, 1)$, the PCP computes $U_{\overrightarrow{p},j}$, $V_{\overrightarrow{p},j}$ and $(u_{4,j})^y$ for each rule r'_j by selecting the encodings for each bit $\overrightarrow{p}[i]$ $(i \in [3])$ in the above obfuscated table and sends $\{j, U_{\overrightarrow{p},j}, V_{\overrightarrow{p},j} \cdot (u_{4,j})^y\}_{j\in[3]}$ with the corresponding traffic to the VEP:
◇ For rule r'_1, $U_{\overrightarrow{p},1} = g^{\alpha_{0,1,1}+\alpha_{1,2,1}+\alpha1,3,1}$, and $V_{\overrightarrow{p},1} = \beta_{0,1,1}\beta_{1,2,1}\beta_{1,3,1} \cdot h_x^{\alpha_{0,1,1}+\alpha_{1,2,1}+\alpha1,3,1}$;
◇ For rule r'_2, $U_{\overrightarrow{p},2} = g^{\alpha_{0,1,2}+\alpha_{1,2,2}+\alpha1,3,2}$, and $V_{\overrightarrow{p},2} = \beta_{0,1,2}\beta_{1,2,2}\beta_{1,3,2} \cdot h_x^{\alpha_{0,1,2}+\alpha_{1,2,2}+\alpha1,3,2}$;
◇ For rule r'_3, $U_{\overrightarrow{p},3} = g^{\alpha_{0,1,3}+\alpha_{1,2,3}+\alpha1,3,3}$, and $V_{\overrightarrow{p},3} = \beta_{0,1,3}\beta_{1,2,3}\beta_{3,3} \cdot h_x^{\alpha_{0,1,3}+\alpha_{1,2,3}+\alpha1,3,3}$.

Verdict Enforcing:

Upon receiving a new policy checking result $\{j, U_{\overrightarrow{p},j}, V_{\overrightarrow{p},j} \cdot (u_{4,j})^y, j\}_{j\in[3]}$ from the VEP, the VEP calculates $(U_{\overrightarrow{p},j})^x$ with its private key x and checks whether $V_{\overrightarrow{p},j} \cdot (u_{4,j})^y = (U_{\overrightarrow{p},j})^x \cdot v_{4,j}$ holds for each rule r'_j.
◇ Only for rule r'_3, we have $V_{\overrightarrow{p},3} \cdot (u_{4,3})^y = (U_{\overrightarrow{p},3})^x \cdot v_{4,3}$;
◇ Therefore the VEP will block the traffic according to A_3 in the rule r'_3.

not collude. Malicious PCP or VEP is not within the scope of this paper, as verifying outsourced functionality is another research topic (see [5] for a good survey on this topic).

IV. SECURITY-ENHANCED FIREWALL AS A SERVICE

Our proposed firewall outsourcing scheme **SE-FWaaS** preserves network policy confidentiality in four phases of operations: offline construction of obfuscated firewall rulesets on the customer side, distribution of obfuscated firewall rulesets to different clouds, online policy checking for new connections on the IT service cloud side, and per-packet based verdict enforcement on the telecom cloud side. In Table II, we illustrate our **SE-FWaaS** by a simplified firewall ruleset.

A. *Offline Obfuscated Firewall Construction*

In this phase, the customer first bootstraps the whole **SE-FWaaS** system by setting the cryptographic parameters and keys for all involved parties. The obfuscating construction of our scheme is based on the ElGamal encryption system [16]. The customer generates a cyclic group $\mathbb{G}$ of order q with generator g, and then chooses x and y randomly from $\{1, \cdots, q-1\}$ as the private keys for the VEP and PCP respectively. The customer computes $h_x = g^x$ and $h_y = g^y$ as the corresponding public keys and publishes h_x and h_y with the security parameters $\mathbb{G}$, q and g. Note that for the ElGamal encryption, the ciphertext of a message z under the public key $h_x = g^x$ is a message pair in the form of $\mathsf{Enc}_{h_x}(z) = (g^u, z \cdot h_x^u)$, for random $u \in \{1, \cdots, q-1\}$. Therefore ElGamal encryption is probabilistic and achieves semantic security if the decisional Diffie-Hellman assumption holds in $\mathbb{G}$ [16]. For two messages z_1 and z_2 with two random u_1 and u_2, the following homomorphic property holds: $\mathsf{Enc}_{h_x}(z_1) \cdot \mathsf{Enc}_{h_x}(z_2) = (g^{u_1}, z_1 \cdot h_x^{u_1}) \cdot (g^{u_2}, z_2 \cdot h_x^{u_2}) = (g^{u_1+u_2}, z_1 z_2 \cdot h_x^{u_1+u_2}) = \mathsf{Enc}_{h_x}(z_1 \cdot z_2)$.

For a given firewall with m rules $r_j = (\overrightarrow{v}_j, W_j, A_j)_{j\in[m]}$, the customer obfuscates the predicate part (i.e., $\text{Predicate}_j = (\overrightarrow{v}_j, W_j)$) of each rule, and generates a new firewall ruleset in the form of $r'_j = (\text{Predicate}'_j, A_j)_{j\in[m]}$ as follows. Note that $(\overrightarrow{v}_j, W_j)$ defines a predicate on n-bit packet header inputs. To obfuscate the original predicate of rule j, for each input bit $i \in [n]$, the customer picks four numbers $\{\alpha_{0,i,j}, \beta_{0,i,j}; \alpha_{1,i,j}, \beta_{1,i,j}\}$ from $\{1, \cdots, q-1\}$ in an independent and uniformly random way. If $i \in W_j$, namely the entry is a wildcard "$*$", we add an additional constraint that $\beta_{0,i,j} = \beta_{1,i,j} = \beta_{i,j}$. Then the customer can compute two pairs of encodings $\{u_{b,i,j}, v_{b,i,j}\}_{b\in\{0,1\}}$ for each bit $i \in [n]$ in the original predicate j as follows:

$$u_{b,i,j} = g^{\alpha_{b,i,j}}; \quad v_{b,i,j} = \beta_{b,i,j} \cdot h_x^{\alpha_{b,i,j}} = \beta_{b,i,j} \cdot g^{x \cdot \alpha_{b,i,j}}. \quad (1)$$

Note that $(u_{b,i,j}, v_{b,i,j}) = \mathsf{Enc}_{h_x}(\beta_{b,i,j})$ with random $\alpha_{b,i,j}$.

To facilitate oblivious policy matching, the customer still needs to generate two more encodings $\{u_{n+1,j}, v_{n+1,j}\}$ with a random $\alpha_{n+1,j}$ selected from $\{1, \cdots, q-1\}$ as follows:

$$u_{n+1,j} = g^{\alpha_{n+1,j}}; v_{n+1,j} = g^{y \cdot \alpha_{n+1,j}} \cdot \prod_{i\in[n]} \beta_{\overrightarrow{v}_j[i],i,j}. \quad (2)$$

Obviously, $(u_{n+1,j}, v_{n+1,j}) = \mathsf{Enc}_{h_y}\left(\prod_{i\in[n]} \beta_{\overrightarrow{v}_j[i],i,j}\right)$ with random $\alpha_{n+1,j}$. Finally the obfuscated rule r'_j is given by

$$r'_j = \left(\{u_{b,i,j}, v_{b,i,j}\}_{b\in\{0,1\},i\in[n]}, \{u_{n+1,j}, v_{n+1,j}\}, A_j\right).$$

Note that this firewall obfuscation phase is done offline and once on the customer side. If there are some updates for the original firewall rules, the customer just reconstructs the obfuscation for related rules and replaces the old ones easily.

B. Offline Decoupling of Firewall Functionality

In our **SE-FWaaS**, the firewall functionality is decomposed into two steps: policy checking and verdict enforcement. We choose to assign the policy-checking operation to the PCP in an IT service cloud and the verdict enforcement to the VEP in a telecom cloud. In order to facilitate the separated firewall operations, the customer first securely sends the private keys y and x to the PCP and VEP, respectively. Then, for each obfuscated rule r'_j, the customer separates it into two parts, i.e., $r'_{j,\text{PCP}} = \left(\{u_{b,i,j}, v_{b,i,j}\}_{b\in\{0,1\},i\in[n]}, u_{n+1,j}\right)$ and $r'_{j,\text{VEP}} = (v_{n+1,j}, A_j)$, and sends $r'_{j,\text{PCP}}$ with rule ID to the PCP and $r'_{j,\text{VEP}}$ with rule ID to the VEP securely. After this phase, the configurations of a distributed firewall in the clouds are settled, and the outsourced firewall is ready to perform packet filtering to protect the customer's enterprise network.

C. Online Per-connection based Policy Checking

In the third phase, the PCP performs policy checking operations on the inbound or outbound traffic of the protected enterprise with its obfuscated ruleset $\{r'_{j,\text{PCP}}\}_{j\in[m]}$.

For each newly initiated connection, the packet header $\overrightarrow{p} = \{0,1\}^n$ is checked with each firewall rule $r'_{j,\text{PCP}}$. The PCP selects encodings from $\{u_{b,i,j}, v_{b,i,j}\}_{b=\{0,1\},i\in[n]}$ according to every bit $\overrightarrow{p}[i]$ ($i \in [n]$) in the packet header, and calculates two numbers $U_{\overrightarrow{p},j}$ and $V_{\overrightarrow{p},j}$ as follows:

$$U_{\overrightarrow{p},j} = \prod_{i\in n} u_{\overrightarrow{p}[i],i,j} = g^{\sum_{i\in[n]} \alpha_{\overrightarrow{p}[i],i,j}}; \quad (3)$$

$$V_{\overrightarrow{p},j} = \prod_{i\in n} v_{\overrightarrow{p}[i],i,j} = \left(\prod_{i\in n} \beta_{\overrightarrow{p}[i],i,j}\right) \cdot h_x^{\left(\sum_{i\in[n]} \alpha_{\overrightarrow{p}[i],i,j}\right)}. \quad (4)$$

Then the PCP calculates $V_{\overrightarrow{p},j} \cdot (u_{n+1,j})^y$ with its private key y and sends $\{j, U_{\overrightarrow{p},j}, V_{\overrightarrow{p},j} \cdot (u_{n+1,j})^y\}_{j\in[m]}$ with the corresponding traffic to the VEP. Note that this policy checking operation occurs only at the time of connection setup.

D. Online Per-packet based Verdict Enforcement

Upon receiving a new policy checking result $\{j, U_{\overrightarrow{p},j}, V_{\overrightarrow{p},j} \cdot (u_{n+1,j})^y\}_{j\in[m]}$ from the PCP, the VEP first calculates $(U_{\overrightarrow{p},j})^x$ with its private key x. Then the VEP performs oblivious policy matching by testing the following equalities:

$$V_{\overrightarrow{p},j} \cdot (u_{n+1,j})^y = \left(U_{\overrightarrow{p},j}\right)^x \cdot v_{n+1,j}, \text{ for } \forall j \in [m]. \quad (5)$$

Equation (5) holds for rule j if and only if:

$$\prod_{i\in[n]} \beta_{\overrightarrow{p}[i],i,j} = \prod_{i\in[n]} \beta_{\overrightarrow{v}_j[i],i,j}, \quad (6)$$

which means that the packed header $\overrightarrow{p}$ satisfies the predicate $(\overrightarrow{v}_j, W_j)$ of rule j, i.e., $\forall i \notin W_j : \overrightarrow{p}[i] = \overrightarrow{v}_j[i]$. Note that this oblivious matching occurs only once for a newly initiated connection. The VEP can cache the action A_j in the matched rule $r'_{j,\text{VEP}}$ as the verdict of this connection, and perform the per-packet filtering task for the subsequent data packets.

E. Security Analysis

The aim of firewall obfuscation is to make it unintelligible to an adversary, or impossible to reverse-engineer, while preserving its original functionality. It is easy to check that our obfuscation construction preserves the functionality of each firewall rule's predicate and matching condition (cf. Equation (6)). To maintain the confidentiality of firewall rulsets from the clouds, our **SE-FWaaS** utilizes two mechanisms. First, we obfuscate the firewall rules in a secure way, such that the information an adversary can learn from this obfuscated firewall can also be learned from observing the input and output packets from the original firewall. Secondly, we separate the obfuscated firewall into two parts and install them into two independent clouds. Based on the assumption that these two clouds will not collude, the adversary even cannot learn the wildcard locations (which means that for an obfuscated predicate we can hide which bits are ignored and which ones are influential). By separating the firewall functionality, our scheme also makes it impossible for each individual cloud to launch offline probing attacks successfully.

Previous obfuscation constructions (like *SOFA* in [12]) are based on the existence of multilinear maps [17]. Unfortunately, all candidate instantiations of multilinear maps in the literature are found recently to be insecure. By separating the firewall functionality, our obfuscation construction does not rely on any multilinear mapping scheme. Our construction is based on ElGamal encryption, which is proven semantically secure [16].

V. Experimental Evaluation

To demonstrate the practicality of our **SE-FWaaS**, we implement a prototype system and evaluate the performance of it through the experimentation with real-life firewall rules. We conduct our experiments on three machines running Windows 7 professional with a 3.4 GHz Inter(R) Core(TM) i3-4310 CPU and 4GB RAM. The firewall rules are randomly selected from the real network database. We are interested in the delays associated with the phases of our **SE-FWaaS** operations.

Overhead of Policy Obfuscating Phase: This phase incurs a one-time cost and only needs to be re-initiated if the firewall rules are modified. We measure the computation and communication delay involved, from the moment the customer begin to obfuscate the original firewall rules until the PCP and VEP receive the obfuscated firewall rulsets. The result is shown in Fig. 2 as a function of the number of firewall rules. We observe an approximate linear increase in overhead with the increase in mumbler of rules. We also find that obfuscating time is about 112 seconds for 30 firewall rules. We argue that

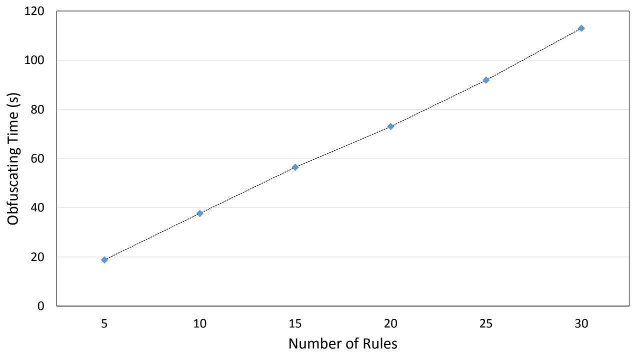

Fig. 2. Overhead of Policy Obfuscating Phase

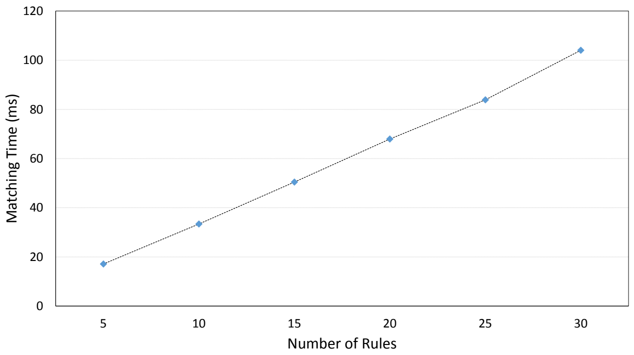

Fig. 3. Overhead of Policy Checking Phase

the time is acceptable because the policy obfuscating needs to be done only once and can be processed offline.

Overhead of Policy Checking Phase: This phase takes place for every new connection through the firewall. We measure the delay that is used for calculating the policy checking result for each new connection. Fig. 3 shows the computational delay as a function of the number of firewall rules. The computational delay in this phase is almost linear with the number of the firewall rules. The checking time for a packet and one firewall rule (5-tuple of 104-bit length) is about 3.46 ms as shown in Fig. 3. Thus our scheme is much more efficient comparing with the schemes in [12], in which the execution time of three proposed schemes is 300 ms, 100 ms and 6 ms.

Overhead of Verdict Enforcing Phase: This phase consists in performing oblivious rule matching and checking for the verdicts stored, and consequently applying this verdict. The experiments shows that it takes about 1 ms in verdict enforcing for one packet, and this delay grows logarithmically with the number of rules.

The measurements in both policy checking and verdict enforcing phases show a remarkable increase in latency with increasing number of firewall rules. However our experiments are performed on an ordinary PC and no firewall optimization techniques are adopted. In practice, when these two phases are executed in the clouds, many parallelism and acceleration mechanisms can be utilized to greatly reduce the latency.

VI. Conclusion

This paper addressed the problem of outsourcing network function like firewalling to public clouds, where network function policies need to be kept confidential from the clouds and third parties. We design and implement the **SE-FWaaS** by leveraging the unique features of heterogeneous cloud environment. The key ingredients in our **SE-FWaaS** are the distribution of the firewall primitives, namely policy checking and verdict enforcing, to two independent public clouds, and the enabling techniques of efficient firewall obfuscation and oblivious rule matching.

Although in this paper we choose firewalls, a typical and indispensable middlebox, as the case study to present the details of **SE-FWaaS**, we believe that the techniques and framework developed in **SE-FWaaS** can be applied to secure other middleboxes outsourced to heterogeneous clouds, including IDSs, NATs, deep packet inspections and HTTP proxies.

References

[1] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: network processing as a cloud service," in *Proc. of ACM SIGCOMM 2012*, Helsinki, Finland, August 2012.

[2] T. Benson, A. Akella, A. Shaikh, and S. Sahu, "CloudNaaS: a cloud networking platform for enterprise applications," in *Proc. of ACM SOCC 2011*, Cascais, Portugal, October 2011.

[3] G. Gibb, H. Zeng, and N. McKeown, "Outsourcing network functionality," in *Proc. of ACM HotSDN 2012*, Helsinki, Finland, August 2011.

[4] European Telecommunications Standards Institute, "NFV whitepaper." [Online]. Available: https://portal.etsi.org/nfv/nfv_white_paper.pdf

[5] S. K. Fayazbakhsh, M. K. Reiter, and V. Sekar, "Verifiable network function outsourcing: requirements, challenges, and roadmap," in *Proc. of ACM HotMiddlebox 2013*, Santa Barbara, CA, December 2013.

[6] National Science and Technology Council and Executive Office of the President, "Trustworthy cyberspace: Strategic plan for the federal cybersecurity research and development program," December 2011. [Online]. Available: https://www.whitehouse.gov/sites/default/files/microsites/ostp/fed_cybersecurity_rd_strategic_plan_2011.pdf

[7] "Survey: 88% of ICT Employees Would Steal," 2008. [Online]. Available: http://www.scoop.co.nz/stories/BU0811/S00203.htm

[8] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing: taxonomy and open challenges," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 369–392, May 2014.

[9] A. R. Khakpour and A. X. Liu, "First step toward cloud-based firewalling," in *Proc. of IEEE 31st Symposium on Reliable Distributed Systems*, Irvine, CA, Oct. 2012.

[10] T. Kurek, M. Niemiec, and A. Lason, "Taking back control of privacy: a novel framework for preserving cloud-based firewall policy confidentiality," *International Journal of Information Security*, pp. 1–16, May 2015.

[11] O. Rottenstreich and I. Keslassy, "The Bloom paradox: When not to use a bloom filter," *IEEE/ACM Transactions on Networking*, vol. 23, no. 3, pp. 703–716, June 2015.

[12] J. Shi, Y. Zhang, and S. Zhong, "Privacy-preserving network functionality outsourcing," Feb. 2015. [Online]. Available: http://arxiv.org/abs/1502.00389

[13] J.-S. Coron, T. Lepoint, and M. Tibouchi, "Practical multilinear maps over the integers," in *Proc. of CRYPTO 2013*, Santa Barbara, CA, August 2013.

[14] J. H. Cheon, K. Han, C. Lee, H. Ryu, and D. Stehle, "Cryptanalysis of the multilinear map over the integers," in *Proc. of EUROCRYPT 2015*, Sofia, Bulgaria, April 2015.

[15] L. Melis, H. J. Asghar, E. D. Cristofaro, and M. A. Kaafar, "Private processing of outsourced network functions: Feasibility and constructions," in *Proc. of ACM SDN-NFVSec 2016*, New Orleans, LA, March 2016.

[16] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE transactions on information theory*, vol. 31, no. 4, pp. 469–472, 1985.

[17] Z. Brakerski and G. N. Rothblum, "Obfuscating conjunctions," in *Proc. of CRYPTO 2013*, Santa Barbara, CA, Aug. 2013.