CONTRIBUTED PAPER

Building In-the-Cloud Network Functions: Security and Privacy Challenges

The article surveys the state-of-the-art literature on network function outsourcing, with a special focus on privacy and security issues.

By Peipei Jiang, Qian Wang, *Senior Member IEEE*, Muqi Huang, Cong Wang, *Fellow IEEE*, Qi Li, *Senior Member IEEE*, Chao Shen, *Senior Member IEEE*, and Kui Ren, *Fellow IEEE*

ABSTRACT | Network function virtualization (NFV) has been promising to improve the availability, programmability, and flexibility of network function deployment and communication facilities. Meanwhile, with the advancements of cloud technologies, there has been a trend to outsource network functions through virtualization to a cloud service provider, so as to alleviate the local burdens on provisioning and managing such hardware resources. Promising as it is, redirecting the communication traffic to a third-party service provider has drawn various security and privacy concerns. Traditional end-to-end encryption can protect the traffic in transmit, but it also hinders data usability. This dilemma has raised wide interests from both industry and academia, and great efforts have been made to realize privacy-preserving network function outsourcing that can guarantee the confidentiality of network communications while preserving the ability to inspect the traffic. In this article, we conduct a comprehensive survey of the state-of-the-art literature on network function outsourcing, with a special focus on privacy and security issues. We first give a brief introduction to NFV and pinpoint its challenges and security risks in the cloud context. Then, we present detailed descriptions and comparisons of existing secure network function outsourcing schemes in terms of functionality, efficiency, and security. Finally, we conclude by discussing possible future research directions.

KEYWORDS | Network function outsourcing; network function virtualization (NFV); privacy preservation.

Manuscript received May 30, 2021; accepted November 1, 2021. Date of current version December 8, 2021. This work supported in part by NSFC under Grant 61822207, Grant U20B2049, Grant 62132011, Grant U21B2018, Grant 61822309, Grant 61773310, Grant U1736205, Grant 62032021, and Grant 61772236; in part by the Fundamental Research Funds for the Central Universities under Grant 2042021gf0006; in part by the Research Grants Council of Hong Kong under Grant CityU 11217819, Grant CityU 11217620, and Grant R6021-20F; in part by the Beijing National Research Center for Information Science and Technology (BNRist) under Grant BNR2020RC01013; in part by the National Key Research and Development Program under Grant 2020YFB1406900; in part by the Shanxi Province Key Industry Innovation Program under Grant 2021ZDLGY01-02; and in part by the Zhejiang Key Research and Development Plan under Grant 2019C03133. *(Corresponding author: Qian Wang.)*

Peipei Jiang is with the Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan, Hubei 430072, China, and also with the Department of Computer Science, City University of Hong Kong, Hong Kong, SAR, China (e-mail: ppjiang@whu.edu.cn).

Qian Wang is with the School of Cyber Science and Engineering, Wuhan University, Wuhan, Hubei 430072, China (e-mail: qianwang@whu.edu.cn).

Muqi Huang is with the School of Computer Science, Wuhan University, Wuhan, Hubei 430072, China (e-mail: huangmuqi@whu.edu.cn).

Cong Wang is with the Department of Computer Science, City University of Hong Kong, Hong Kong, SAR, China (e-mail: congwang@cityu.edu.hk).

Qi Li is with the Institute for Network Sciences and Cyberspace, Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University, Beijing 100084, China (e-mail: qli01@tsinghua.edu.cn).

Chao Shen is with the School of Cyber Science and Engineering, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: chaoshen@mail.xjtu.edu.cn).

Kui Ren is with the College of Computer Science and Technology and the Institute of Cyber Science and Technology, Zhejiang University, Hangzhou, Zhejiang 310027, China (e-mail: kuiren@zju.edu.cn).

Digital Object Identifier 10.1109/JPROC.2021.3127277

I. INTRODUCTION

The traditional telecommunication industry heavily depends on various specialized hardware equipment of different manufacturing standards, which leads to an unfriendly product development cycle. Early in 2012, the world-leading European Telecommunications Standards Institute Industry Specification Group (ETSI ISG)

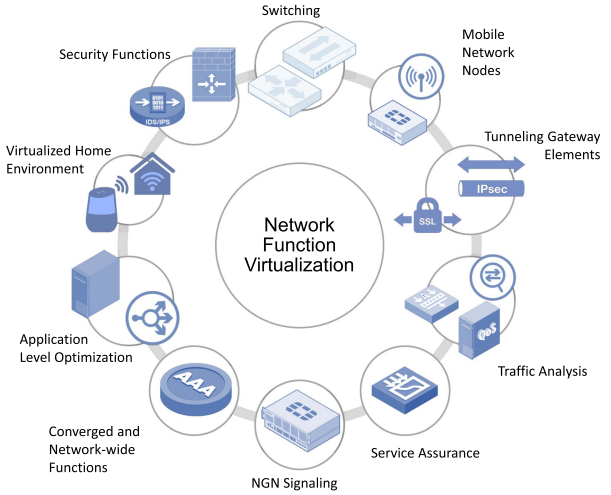


Fig. 1. ***Many representative applications of NFV.***

worked out the framework of network function virtualization (NFV) and proposed accordingly the white paper [1]. NFV relieves the burden of the complicated deployments of networking and communication hardware infrastructures by transferring the communication data to standard hardware (e.g., standard high volume servers, switches, and storage) in the network nodes. It also simplifies the design and operation of network functions, and increases flexibility, programmability, and expansibility of network functions [1]–[3]. Fig. 1 summarizes many representative applications of NFV, as also suggested in [1]. With the increasing demand for advanced network infrastructures, the global NFV market size is expected to grow to $36.3 billion by 2024 [4].

Meanwhile, with the rapid development of cloud technologies [5]–[8], individuals and enterprises are tending to outsource virtualized network functions to the cloud, so as to mitigate the local burdens on provisioning and managing specialized hardware resources. As of today, network functions, also known as middleboxes, such as firewalls, deep packet inspection (DPI), load balancers (LBs), can be easily implemented on cloud platforms. Many cloud platforms have publicly offered the services of in-the-cloud network functions, e.g., Amazon CloudFront [9], Microsoft Azure cloud firewalls [10], and Google's networking services [11], including cloud NAT, cloud load balancing, and cloud router. Very recently, VMware Inc. announced that they had enhanced their VMware Ready for Telco Cloud to enable virtualized network functions with their VMware Telco Cloud platform [12].

The powerful computing capabilities of cloud infrastructure can provide much-improved high-performance packet processing demanded in various network functions. Individuals and enterprises will no longer suffer from cumbersome network hardware configurations and management, especially for the inconvenient network function updates.

Though promising, it is worth noting that, once the network function is outsourced, and the traffic is redirected to the cloud, the cloud server then has the root access to both the outsourced functions and communication traffic, which may raise severe security concerns. The cloud, with its nature of outsourced service offering, can be vulnerable to internal threats (e.g., abuse of the root access) and high-value target of external threats (e.g., various malicious attacks) [64]. The communication traffic over the network may carry confidential information, and thus, directly redirecting the packets to the cloud will violate the user's privacy. Furthermore, the outsourced function itself (e.g., the inspection rules for intrusion detection systems (IDSs), which may be generated by a third-party professional service provider, such as McAfee [65]) might contain proprietary assets. These commercially sensitive data could be exposed if deployed in the environment without security attestation. Today's secure communication practices often employ end-to-end encryption to protect data-in-transit. However, such a practice inevitably hinders the processing and computation of the packets and rules at the middlebox. To better support the middlebox functionality, it may have to require the traffic to be decrypted in the middlebox, which would violate the end-to-end security [66]. In light of this need, there has been a relevant research line to design new transport layer security (TLS) protocols that are compatible with middleboxes [67]–[70]. These protocols enable traffic-owner-controlled decryption at the middleboxes. In other words, the traffic is visible to the middleboxes to some extent. This line of work has great significance in practical and systematic aspects but does not target the security level that we discuss in this article. In our survey, we focus on the designs where middleboxes are never supposed to learn the plaintext content in the traffic but can still inspect the encrypted traffic. As we show later, how to process encrypted packets with respect to protected rules without revealing any confidential information about the communication remains a big challenge.

Network functions can be described as matching and computing on the network flows with pregenerated rules and, finally, getting the action to be performed on the packets [37]. With regard to the secure in-the-cloud network functions, the first challenge lies in how to support enriched operations on encrypted data. It is easy to complete the operations of matching and computation on the plaintext traffic. However, even the simplest single pattern matching can be costly to perform on encrypted traffic. Therefore, before diving into the specific privacy-preserving techniques, it is important to categorize the network functions from the perspective of matching versatility. For example, the matching rule can be classified into equality matching and more enriched ones, such as range matching and regular expression matching. Furthermore, some network functions may also require modification on packets [48] (e.g., the network address translation (NAT) protocol) and inspection over stateful packets [15], [37]. Besides functionality, privacy issues are also important in

Table 1 Development on Different Categories of Secure In-the-Cloud Network Functions

	2012	2013	2015	2016	2017	2018	2019	2020-2021
TEE-based middleboxes			KSHK15 [13]	PRI [14]	SGXBox [15] TrustClick [16]	ShieldBox [17] MAP18 [18] SafeBricks [19]	LightBox [20] MBB19 [21]	Phoenix [22] EVE [23]
DPI/IDS			BlindBox [24]	YWLW16 [25] Embark [26]	CloudDPI [27] SPABox [28] SEST [29]	GWJ18 [30]	ES-DPI [31] PrivDPI [32] CloudDPI* [33]	Pine [34] SHVE+ [35] (2021)
Firewalls			SOFA [36]	PNFV [37] SWZZ16 [38] SE-FWaaS [39]		GWYJ18 [40]		GWWY20 [41]
Stateful inspection				PNFV [37]	SGXBox [15]		LightBox [20]	
Routing	GSPS12 [42]	STRIP [43]	PYCRO [44] KSHK15 [13]		ADSS17 [45] SIXPACK [46]		PYCRO* [47]	
NAT				Splitbox [48] Embark [26]				
Service chaining				Splitbox [48]	SICS [49] vSFC [50]	ShieldBox [17] SafeBricks [19]	LightBox [20]	vSFC* [51] (2021)
Content delivery		MTM13 [52] WWD13 [53]		YWWC16 [54] CYZW16 [55]		REET [56]		Phoenix [22] CYZ21 [57] (2021)
Verifiable middlebox	ZZGH12 [58]	vNFO [59]		YDW16 [60]	vSFC [50]	YDW18 [61]	ZDWL19 [62]	EV-DPI [63] vSFC* [51] (2021)

Note that, * denotes that this work is a journal version as an extension of the conference version. SHVE+ [35], vSFC* [51], and CYZ21 [57] are published in 2021.

cloud scenarios. The traffic should be kept private, and the inspection rules are commercially sensitive. Therefore, it is essential to build market-compliant outsourcing protocols to protect both of them [25], [71]. Furthermore, untrusted cloud servers or even malicious ones possibly with financial incentives, which may deliberately miscalculate results to reduce costs or disobey the procedures to obtain more confidential information, should also be considered [58], [59]. In conclusion, the goals of outsourcing network functions in a privacy-preserving way are to: 1) complete general network functions; 2) preserve the privacy of the traffic and/or rules; and 3) enable verifiable computation on the cloud middleboxes.

We summarize the development and roadmaps of noteworthy secure in-the-cloud network functions over the period of 2012–2021 in Table 1. From the perspective of functionality, we categorize the solutions into different types of middleboxes, e.g., DPI/IDS, firewalls, and NAT. The pioneering work, BlindBox [24], first enabled an outsourced middlebox for DPIs without decrypting the traffic in the middle by using the technology of garbled circuit (GC) [72]. Follow-up works based on different cryptographic tools can achieve more enriched network functions, such as range matching [26], [37], [40] and wildcard matching [38], [39]. The outsourced middleboxes based on the trusted hardware can achieve more general network functions [13]–[16] and also enable the verification. Up until now, efforts have been made to promote the development of in-the-cloud network functions, but there still remains much room to explore. Hence, we believe that it is necessary to give a comprehensive study on the current progress of network function outsourcing, so as to make the remaining challenges and opportunities clearer to interested researchers.

A. Related Work

To the best of our knowledge, we are the first to provide a deep insight into the topic of outsourcing network functions from the aspect of protecting the privacy of the communication traffic. There are surveys about the issues of NFV [1]–[3], [73] and SDN [74], [75], which mainly focuses on the challenges, technologies, and implementations of NFV and SDN and do not consider the issues of privacy. Pattaranantakul *et al.* [73] take the security issues into consideration. However, this survey focuses on

the security of the process on the virtual machine of the service provider, and the privacy problem is out of scope. Zave and Rexford [76] focus on very general security issues of network interactions. Therefore, the complementary viewpoints [73], [76] are orthogonal to ours.

The most related works are [77]–[79]. Compared to our work, the surveys [77], [78] discussed a smaller portion of research on network function outsourcing and did not classify and conclude the state of the arts as extensively as we do. Poh *et al.* [79] have surveyed a larger scale of solutions than [77] and [78] from the perspective of different techniques. Unlike [79], our survey presents existing works in a new light, i.e., from the perspective of the complexity of network functions. Our angle focuses more on the problems than the solutions, which may give the researchers a clearer understanding of the challenges and corresponding privacy protection mechanisms. Besides, the comparisons among the solutions and cryptographic tools in our survey are much more diversified, and we also give metrics to evaluate the effectiveness of existing mechanisms. Therefore, our work presents a more in-depth tutorial that can help interested researchers quickly grasp the motivations, challenges, and techniques of the fast-rolling in-the-cloud network function.

B. Contribution

In this article, we systematically survey the problems and solutions of the in-the-cloud network functions over the period of 2012–2021. Our contributions are listed in the following.

1) We, for the first time, extensively survey the privacy and security issues of in-the-cloud network functions from the perspective of function complexity. We systematically classify network functions into equality matching, function-enriched matching, and general functions, and summarize corresponding outsourcing mechanisms with pros and cons.
2) We introduce detailed definitions of NFV architecture, outsourcing model, usage scenarios, and the threat model, providing a concrete description of the issues of secure network function outsourcing. We also give a rich background of virtualized network functions and cryptographic techniques, which can help lay out the comprehensive ground field for subsequent research.
3) We give metrics to evaluate existing mechanisms and carry out meticulous comparisons among the privacy preservation techniques from the perspectives of functionality, security, and efficiency. We also provide possible future research directions on this topic to encourage readers to explore more practical and secure outsourcing constructions.

C. Organization

The organization of the remainder is shown as follows. Section II introduces the background of NFV and the related outsourcing practice, including the definitions and examples of NFV and the explanation of pattern matching and range matching in the cloud settings. Section III discusses the security issues, including the introduction to the cryptographic tools and the detailed description of the threat model of in-the-cloud middleboxes. In Section IV, we introduce the state of the arts from the perspective of functional completeness, such as equality matching, function-enriched matching, and general network function. Section V provides a deep comparison among the state of the arts on functionality, security, and efficiency. Finally, open research directions are proposed in Section VI.

II. NFV AND THE OUTSOURCING PRACTICE

In this section, we start by introducing the concepts of NFV and several examples of network functions. Then, we categorize the outsourced network functions and introduce the outsourcing model. Finally, we present three representative usage scenarios of outsourcing network functions.

A. Network Function Virtualization

Traditional network architecture suffers from the incompatibility of diverse network hardware appliances. Middleboxes are designed and implemented under different standards, which dramatically hinders network application development and maintenance. It is also difficult and costly to implement new network functions by adding new middleboxes in an existing network architecture with incumbent hardware settings.

NFV is proposed to solve the above problem. It aims to standardize network middleboxes by consolidating various networking equipment, e.g., servers, switches, and storages, in network nodes and datacenters [1]. As shown in Fig. 2, a typical NFV architecture includes three components: 1) virtualized network functions; 2) the NFV infrastructure, including virtual compute, virtual storage, virtual network, and related hardware resources; and 3) NFV management and orchestration. In the following, we present several typical examples of network functions.

1) Deep Packet Inspection: DPI is a network traffic analysis technique that performs traffic analysis, inspection, and filtering [80], [81]. The rule generator determines inspection rules in middleboxes. Different from common filtering functions that only detect the IP header; the rules in DPI may involve the features of both the header and the application layer payload. A DPI middlebox will compare the incoming packets with the inspection rules. If there is a match, the middlebox will perform certain actions to the packets, such as accept and drop, according to the rules. DPI can be used for malicious network traffic detection, precise advertising, and so on.

2) Intrusion Detection System: IDS is a network function that can prevent malicious access to

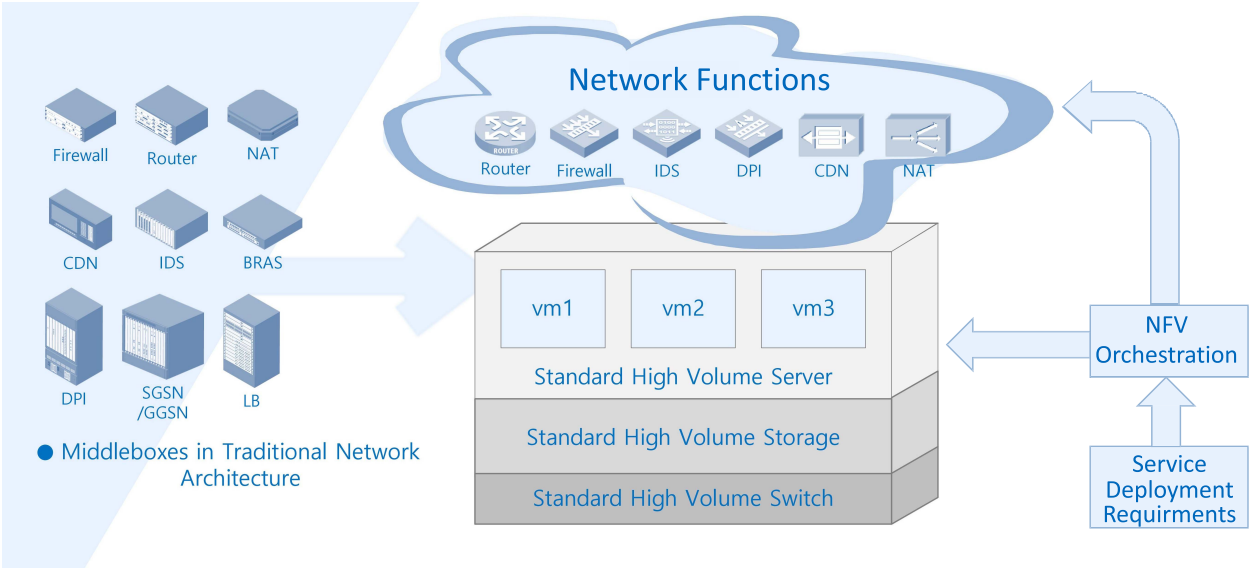


Fig. 2. ***Architecture and examples of NFV [1]. The hardware-based middlebox (left) can be realized in an NFV architecture as virtualized network functions (right). The architecture can be divided into three levels: the network functions, the NFV infrastructure, including virtual compute, virtual storage, virtual network, and related hardware resources, and the NFV management and orchestration.***

internal networks [82]–[84]. IDS has three variations: 1) signature-based detection, which utilizes the pattern match to detect malicious traffic; 2) anomaly-based detection, which relies on machine learning to train a model to differentiate good or bad traffic; and 3) reputation-based detection, which scores each traffic flow to recognize potential threats. The widely used signature-based approach includes two steps: preprocessing and attack signature matching. The packets are first decoded and reassembled into IP packet fragments. When the flow is acquired, the IDS will first conduct the multistring pattern matching to detect specific keywords (or key ports). If nothing is found, the flow is tagged as "innocent." If there is a match, IDS will further perform signature detection, which includes the metadata pattern match (e.g., IP range), string pattern, or binary pattern match in the payload.

3) Firewall: The firewall is a widely used network function that monitors and controls the network traffic between the internal network and the external network [85]. Traditional firewalls run on a specialized middlebox and work as a packet filter based on the ruleset. According to the ruleset, the firewall checks the five tuples in the header, i.e., the source IP address, the destination IP address, the source port number, the destination port number, and the protocol, to decide whether to accept or block the packet. To tackle the problem that traditional firewalls do not keep the track of packet contexts, stateful firewalls have been proposed [86], [87]. A stateful firewall maintains a state table to store the context of each packet so that new connection requests can be associated with previous connections [88]. In this way, stateful firewalls can support more expressive policies and provide a stronger security guarantee.

4) Network Address Translation: NAT is typically deployed at network edges to allow a large number of hosts to connect to the Internet with the same IP addresses [89], [90]. When a user in an internal network wants to communicate with the Internet, the NAT gateway replaces the internal address with a public IP address according to the mapping table. NAT shields the internal network such that all computers within the intranet are invisible to the public network. NAT mitigates the problem of IP address exhaustion by enabling multiple computers to share Internet connections.

5) Load Balancer: LB assigns the incoming traffic to multiple network devices (e.g., firewalls) or links, which effectively improves the processing capability while guaranteeing high reliability. The LB acts as a scheduler in a server cluster, who first receives all requests from the clients and then assigns the requests to the backend servers according to the load condition of each server to optimize the overall network performance [91]–[93].

6) Content Delivery Networks: Content delivery networks (CDNs) [94], [95] aim to deliver large-scale content, e.g., video streams. CDN is a distributed network consisting of proxy servers and data centers that can improve the response speed and hit ratio of users. Through load balancing, content distribution, scheduling, and other functions of the central data centers, CDN relies on the proxy servers deployed in various places to enable users to get the required content nearby. In-network caching and

Table 2 Examples of Different Matching Types in the Style of Snort Rules

Matching type	Example	Meaning
equality match	alert tcp $EXTERNAL_NET$ any -> $HOME_NET$ 21\ (content: "pattern1"; content: "pattern2"; flow: stateless;)	Alert TCP traffic from any port going to port 21 with content "pattern1" and "pattern2".
range match	alert tcp $EXTERNAL NET$ 1: 1024 -> $HOME_NET$ 500: \ (msg: "PortRange"; nocase; flow: stateless;)	Alert TCP traffic from privileged ports from 1 to 1024 going to ports greater than or equal to 500.
regular expression match	alert tcp $EXTERNAL_NET$ any -> $HOME_NET$ 80\ (msg: "RE"; offset: 3; pcre: "/\d{2}-\d{5}/";)	Alert TCP traffic from any port going to port 80 that contains a string with the form "2 digits followed by a hyphen and 5 more digits" after 3 bytes of the payload.
stateful match	alert tcp $EXTERNAL_NET$ any -> $HOME_NET$ 80 (msg: "The client sends an initial SYN to the server"; flags: S; flowbits: noalert;) alert tcp $HOME_NET$ 80 -> $EXTERNAL_NET$ any (msg: "The server responds with SYN/ACK back to the client"; flags: SA; flowbits:set, tcp.est;)	Track the HTTP flow between the server and the clients.

redundancy elimination (RE) [96] can be leveraged to increase the efficiency of content delivery.

B. Outsourcing Network Functions

In an NFV architecture, network functions are performed on virtual machines, making it suitable to leverage cloud services. Outsourcing network functions to a cloud server can relieve the local computational burden and reduce the costs of deploying network equipment. Nonetheless, the traffic of clients has to be transmitted to the untrusted cloud. The clients should encrypt the packets before sending them to the cloud to preserve the privacy of the traffic.

The studies of traditional NFV do not specifically classify the network functions from the perspective of matching granularity. However, the operations on encrypted traffic are much more complicated than those on plaintext. To distinguish the difficulty of realizing different network functions on encrypted traffic, we need to classify the network functions from the terms of the computational effort. Following [24] and [77], we classify the general network functions into two classes: equality matching and function-enriched matching. The function-enriched matching-based network functions can be further divided into range match, substring match, wildcard match, and regular expression match. We present a toy example of matching types in the style of Snort rules in Table 2. In addition to the basic matching, stateful matching and packet modification are also common in network functions. In the context of outsourced network function, stateful match and packet modification can be regarded as advanced functions. Besides, according to Chiosi *et al.* [1], NFV is applicable to switching elements like routers and router functions. Thus, we can also regard routing as a network function from a high level of view. The explanations of the above functions are listed in the following.

- *Equality match* checks whether a packet contains one or more specific keywords (e.g., signatures or watermark) and matches the keywords with an action, such as drop or accept.
- *Function-enriched match* supports various expressive matches beyond equality match, such as range, subset, wildcards, offset, and regular expression.
- *Stateful match* checks the relationship between the packets and can monitor dynamic network states.
- *Modification:* Some network functions require modification on certain fields in packets, e.g., NAT.
- *Routing* requires certain computations to determine the best routing strategies.

C. Outsourcing Model

There are two types of outsourcing models [26], [60]: the bounce model [97] (also called APLOMB system) and the go-through model [24] (also called NFV system). As shown in Fig. 3, in the bounce model, when interacting with external sites, the gateway "bounces" the traffic to the service provider to perform network functions, after which the middlebox sends back the traffic to the gateway. The rules in the middlebox are generated by the enterprise (or the gateway) who requests the network function outsourcing service. The go-through model includes four parties: the sender, the receiver (or the server), the rule generator, and the service provider. In the setup phase, the rule generator outsources the rules to the middlebox. The rule generator can be a third-party professional corporation, such as McAfee or an intranet administrator. During the communication, the sender directs the traffic to the middlebox that runs outsourced network functions, and the middlebox sends the traffic to the receiver.

D. Usage Scenarios

To clarify the motivation of outsourcing network functions to the cloud in a privacy-preserving way, we further introduce three representative usage examples.

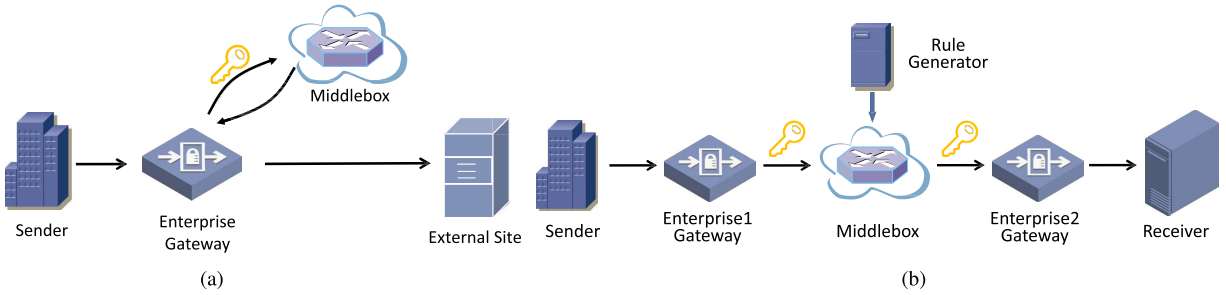


Fig. 3. *Two types of network function outsourcing models [26]: (a) bounce model and (b) go-through model. In the bounce model, the gateway sends the traffic to the service provider, and the traffic will be sent back to the gateway after detection. In the go-through model, the traffic is processed in the middlebox between the sender and the receiver.*

1) Parental Controls: This example is first introduced in [24]. Alice has registered for parental control services from an Internet service provider (ISP) to monitor the traffic and filter adult content for her kids. However, Alice is concerned that the ISP would sell her personal communication data to other organizations. Therefore, Alice would like the ISP to inspect her encrypted traffic instead.

2) Enterprise Network Monitoring: Bob's company has subscribed to an Azure Firewall [10] for real-time malicious traffic filtering with high availability. However, Bob is not comfortable to expose his commercially sensitive data to Azure or the possible eavesdroppers from rival companies. In such a scenario, a privacy-preserving in-the-cloud firewall will meet Bob's requirements.

3) Video Applications: Hulu is an American video platform that offers live TV streams [98]. To achieve a better user experience, Hulu subscribes to Amazon CloudFront, a fast and programmable CDN running on Amazon cloud [9]. In a CDN, the video contents are stored in distributed edge servers to enable quick video delivery. However, Hulu is worried that their clients may be unsatisfied that their private data (e.g., video access history) are exposed to other parties. Hence, Hulu would like to keep their data encrypted in the edge servers, while the CDN in the cloud can still eliminate redundant network traffic, distribute the video contents, and so on.

III. SECURITY ISSUES OF OUTSOURCING

In this section, we introduce the privacy and security issues regarding network function outsourcing. We formally define the threat model to address the challenges of outsourcing network functions. We also present several cryptographic tools that are commonly utilized to design outsourced middleboxes for privacy preservation.

A. Threats in the Outsourcing Practice

Outsourcing network functions to remote cloud benefits a lot in alleviating local computation resources, but, in the meantime, it brings in the risk of exposing sensitive data of both the traffic and the network functions. Here, we introduce possible threats that the clients (or enterprises) may encounter when outsourcing their traffic and network functions, especially from the perspective of privacy. Specifically, we describe the threats from three aspects: adversaries in different network domains, attack surfaces in different network layers, and the attack means. Note that, since this article focuses on the privacy and security issues of building in-the-cloud network functions, we especially discuss the threats arising from the outsourcing practice, where network functions are deployed at untrusted settings. Generally speaking, our considered security and privacy threats are also relevant to other network function deployment scenarios where the context of the deployment might not necessarily be in the trusted domain.

1) Threats in Different Network Domains: From the perspective of administrative domains, threats may come from different network entities involved in the communication process. We consider three types of threats: 1) adversaries residing at the cloud; 2) malicious endpoints, and 3) eavesdroppers in the communication channel.

a) Threats in the cloud: In the outsourcing model, the third-party cloud can get full access to the traffic and the function. There are two types of cloud servers: honest-but-curious servers and malicious ones [59]. An honest-but-curious server strictly follows the protocols but tries to infer as much private information as possible from the traffic, outsourced functions, and processing results [24], [26]. For example, some companies who provide cloud services are reported to sell the clients' private data [99]. More seriously, the cloud is under various external threats, e.g., frequent cloud data breaches [100]. To tackle such a threat, encrypting private data is essential. Malicious cloud servers may disobey the protocol, forge the results to avoid computational expenses, or even learn more sensitive information. In face of a malicious server, a computational verification mechanism is required to ensure the correctness of the results [101], [102].

b) Threats on the endpoints: This type of threat is the same as that considered by traditional network functions, i.e., the original adversaries considered by regular

network functions, such as IDS [24]. A malicious endpoint may send illegal packets to try to pass network functions like DPI and firewalls. Traditional intrusion detection assumes that at least one of the endpoints is honest [82], [83]. Similarly, in the outsourcing context, at least one of the endpoints will actively use the outsourcing service, which we regard as fully trusted. The case of two malicious endpoints is generally not considered in today's literature study. If both endpoints are malicious, they can communicate with each other with a secret encryption key in some secret channel without using NFV. Detecting such kind of encrypted communication requires traffic pattern analysis [103]–[105], which is orthogonal to the topic of this survey.

c) Threats in the channel: The cloud and the communication channels are vulnerable to eavesdroppers. The eavesdroppers may intend to snoop on client traffic or even jam some packets [106]. Encrypting traffic and network function strategies is the most common countermeasure to defend against such threats. For example, security protocols, such as TLS, can defend against eavesdroppers by encrypting the communication traffic. In the design of privacy-preserving network functions, the traffic and the functions are kept encrypted in the communication channel to prevent eavesdroppers from learning private information.

2) Attack Surfaces in Different Network Layers: Since we focus on privacy issues, the attack surface here especially refers to the privacy vulnerabilities of network flows. To be more specific, the private information includes contents in the transportation-layer (i.e., L4) payloads and other header information in the data link and network layers (i.e., L2 and L3). Ideally, the design of the outsourced network functions should keep all the private information from the adversaries and enable specific computation in corresponding fields. Following Duan *et al.* [20], we also divide the private information into two aspects: L2–L4 headers and L4 payload.

a) Private information in L2–L4 headers: The most important information in L2–L4 headers is the five tuples, i.e., the source IP address/port number, destination IP address/port number, and the protocol. It is obvious that such data in plaintext can reveal the identity of the sender and the receiver. Sometimes, clients may not want their private information (e.g., destination addresses) to be exposed to the cloud server or the adversaries in the communication channel. The problem is that it seems to make no sense to forward a packet without letting the routers learn the destination IP address in the actual network architecture. However, in the bounce outsourcing model [see Fig. 3(a)], the real destination IP address is encrypted and forwarded to the in-the-cloud middlebox to perform network functions, such as firewalls and NAT [26]. Such a process protects the private information in the header from the in-the-cloud middleboxes and does not involve physical routers in the path of the sender and the receiver. With regard to defending against the eavesdroppers in the communication channel, the clients can use security protocols like IPsec [107] and TCPcrypt [108] to protect the private data in the headers.

b) Private information in L4 payloads: The L4 payloads carry the specific contents in the communication, and the adversaries strive to learn as much as sensitive information from them. The goal of designing an in-the-cloud network function is to protect private information while preserving the cloud's ability to inspect the traffic.

3) Attack Means: When outsourcing the network functions, we should consider two types of attacks: 1) the original attacks against entire traditional network functions and 2) the attacks on the primitives used to realize computation on encrypted data. The former includes the spying and tempering attacks, the denial-of-service (DoS) attacks, and so on. The latter includes the side-channel attacks on trusted hardware, the cryptanalysis-based attack against customized cryptographic techniques, and so on.

a) Spying attack: In this attack, the adversaries listen to the communication traffic of two innocent end-users, trying to extract as much private information as possible [76]. We have detailed the private information in Section III-A2, and encrypting the traffic can well protect such private information. Preventing spying attacks is the basic security goal of the network function designs.

b) Denial-of-service attack: The DoS attack has a wide category, with the goal of making the target network service unavailable [109]. In DoS attacks, the adversaries strive to exhaust the target network's bandwidth by controlling botnets to launch the User Datagram Protocol (UDP) flood, the Internet Control Message Protocol (ICMP) flood, the Domain Name System (DNS) flood, the HTTP flood, and so on. Many traditional network functions can prevent the DoS attacks, e.g., firewalls, IDS, and DPI. Fayaz *et al.* [110] also proposed to leverage the SDN/NFV techniques to defend against the DoS attacks. As for the outsourced network functions, whether a design can detect or prevent the DoS attacks can be regarded as an advanced feature [25], [111].

c) Side-channel attack: A line of research on outsourced network functions relies on trusted hardware, especially the Intel SGX. However, SGX is vulnerable to various side-channel attacks [112]. For example, in SGX, the memory pages need to be loaded into EPC with limited memory. Since the operating system has direct access to memory management, the malicious OS can decide whether to flush the translation lookaside buffer (TLB). Thus, the adversary can analyze the code of SGX applications to locate the attack address and then learn what the SGX has accessed by flushing the TLB and recording the memory footprint. Unfortunately, most existing hardware-based outsourced network function designs do not consider side-channel attacks. For security considerations, we emphasize that it is essential to consider such

attacks when applying the outsourced network functions in practice.

d) Cryptanalysis-based attack: Besides the trusted hardware, another line is to utilize customized cryptographic tools, which may face potential security threats of cryptanalysis. Take the brute-force attacks for example. In this attack, the adversaries analyze the cryptographic protocols by exhausting the input space. Ideally, the cryptographic protocols are secure against brute-force attacks. However, the design of the cryptography-based outsourced network functions may have defects when applying the encryption schemes. For example, Ning *et al.* [34] pointed out that PrivDPI [32] is vulnerable to the brute-force guessing when the rule set is small.

B. Privacy and Security Goals

In the paradigm of outsourcing the network functions, both the traffic and the outsourced functions (rules) should be protected from the cloud server and other possible adversaries.

1) Rule Privacy: Typical rules contain inspection strategies, such as the filtering rules in DPI/IDS or firewalls [84] and the export policy in routing [42]. With the rules, the middlebox calculates and compares the content of the packet to match the corresponding rule and, finally, informs the gateway what action it should take on the packet. Disclosing rules to adversaries will cause significant potential risks to network security. With the knowledge of the rules, attackers can deliberately construct malicious traffic that can circumvent these rules or even infer the contents of the packet from the results of the inspection process.

There are three types of rule generators: 1) the receiver; 2) the administrator of the sender; and 3) a professional third-party corporation. In all cases, the rules should be kept private from the in-the-cloud middleboxes. The cloud server can only perform the virtualized network functions without knowing the content of the rules. For the first two cases, one of the endpoints can learn the rules, but the other endpoint should not learn them. If the rules are generated by a third-party (i.e., case 3), the rules are trade secrets in this sense, so they should not be revealed to both the endpoints and the cloud middlebox [71].

2) Packet Privacy: The privacy goal of network function outsourcing is that the private information in packets (as discussed in Section III-A2) should not be known to third parties other than the sender and receiver [24], [67]. Therefore, outsourced network functions will be performed on encrypted packets. For outsourcing, the encryption scheme of the packets should be designed together with that of the function to enable computation and comparison. One of the tremendous challenges is the allocation of keys. The endpoints can encrypt the packets, but the secret key should not be revealed to the rule generator who encrypts the rules. The one who encrypts the rules should not decrypt the packets with the rule keys, either. Hence, it is necessary to support middlebox functions over encrypted packets and rules using different keys.

Table 3 Abbreviations of the Security Primitives

Abbreviation	Full Name
FHE	Fully homomorphic encryption
GC	Garbled circuit
OPE	Order-preserving encryption
ORE	Order-revealing encryption
OT	Oblivious transfer
PHE	Partially homomorphic encryption
PRF	Pseudo-random function
SGX	Intel software guard extension
MPC	Secure multi-party computation
SSE	Searchable symmetric encryption
SHE	Somewhat homomorphic encryption
TCB	Trusted computing base
TEE	Trusted execution environment

C. Cryptographic Primitives and Trusted Hardware

To enable privacy-preserving computation on encrypted data, cryptographic tools, such as searchable encryption [113]–[115], homomorphic encryption (HE) [116], [117], secure multiparty computation (MPC) [72], [118], and trusted hardware [119], can be used as the building blocks. Although these tools have been well studied in the field of cryptography, how to apply them in the context of network functions remains to be fully explored. To help the readers better understand the privacy-preserving network function outsourcing designs, we briefly overview a comprehensive list of security and cryptographic tools that could be used to enable privacy-preserving network functions without exposing the packet content and/or the rules. A summary of the abbreviations is presented in Table 3.

1) Oblivious Transfer: Oblivious transfer (OT) [118], [120], [121] solves the problem that a party A wants to share one of the dataset $\mathbf{D} = (D_1, D_2, \ldots, D_n)$ with another party B who wants D_b. A does not want to share other data than D_b, and B does not want to reveal b. OT ensures that B will obtain D_b without learning $\mathbf{D} \backslash D_b$

or revealing b. OT can be built on asymmetric primitives, such as RSA [122].

2) Garbled Circuit: The notion of GC was first proposed by Yao [72] to enhance secure MPC (SMPC). A GC allows two parties to compute results jointly without revealing their own inputs. At a high level, a GC protocol includes two steps: 1) GC preparation and 2) circuit evaluation. In the preparation phase, one party generates the circuit according to the functions. Afterward, the other party evaluates the GC by her choice of the input and the generator's input keys The evaluated results are sent back to the generator to get the final results. Note that the evaluator will obtain the keys obliviously with the auxiliary of OT. Thus, the generator will not learn the input of the evaluator.

3) Homomorphic Encryption: HE is a class of public-key encryption schemes with homomorphic properties [123], where the operations (i.e., addition, multiplication, or both) on the ciphertext will exactly match the same operations on the plaintext. There are roughly three types of HE systems: partially HE (PHE), somewhat HE (SHE), and fully HE (FHE). In PHE, the operations on plaintext correspond to either standard arithmetic addition or standard arithmetic multiplication e.g., the famous Pallier cryptosystem [116]. In SWHE, the operations are limited to some "low-degree" polynomials, e.g., the famous BGV system [117]. In FHE, the operations can be arbitrary composition of addition and multiplication [124]–[128].

4) Multilinear Map: The multilinear map is an extension of the bilinear map. Multilinear maps of the symmetric form and the asymmetric form were first realized in [129] from the ideal lattice. Later, Coron *et al.* [130] brought about multilinear maps over integers. Shortly afterward, Gentry *et al.* [131] came up with a scheme from general lattices. Braderski and Rothblum [132] proposed construction based on the asymmetric multilinear map to obfuscate conjunctions.

5) Order-Preserving Encryption: Order-preserving encryption (OPE) [133] is designed to map the nonuniform distributed plaintext data into ciphertext intervals that are uniformly distributed such that the characteristics of data distribution can be hidden. In OPE, the order relation of the ciphertexts is the same as that of the plaintexts. However, it reveals the original order. Therefore, OPE is vulnerable to statistical inference attacks [134]. According to Cash *et al.* [135], OPE reveals more information than plaintexts order. In the literature, many schemes have been devoted to improving the security of OPE [136], [137].

6) Order-Revealing Encryption: Order-revealing encryption (ORE), also known as efficiently orderable encryption, is symmetric encryption, such as OPE, which can be used for range search on the ciphertext. Compared with OPE, ORE achieves stronger security by obscuring the order relations of the plaintexts. Chenette *et al.* [138] and Lewi and Wu [139] proposed modified solutions to have its security ensured. Cash *et al.* [135] presented the first ORE construction with the bilinear map to reduce the leakage of sensitive information rather than the multilinear map [140]. Nevertheless, Grubbs *et al.* [141] and Durak *et al.* [142] demonstrated that ORE could be under the latest leakage-abuse attacks.

7) Searchable Symmetric Encryption: Searchable symmetric encryption (SSE) [113]–[115], [143]–[145] was first introduced by Song *et al.* [143]. Searchable encryption allows the client to outsource his/her private data while preserving the ability to search on the encrypted database. Curtmola *et al.* [113] further defined the formal security model of SSE and proposed an SSE scheme of optimal search complexity (which is linear with the number of the matched results). Later, Kamara *et al.* [114] proposed a dynamic SSE based on the work of Curtmola *et al.* [113]. Typically, an SSE scheme is built on symmetric primitives, such as pseudorandom functions (PRFs) and pseudorandom permutations (PRPs). Thus, SSE has the advantages of lightweight computation and high run-time efficiency. In an SSE scheme, the dataset is first parsed as a keyword-identifier index, which is called an inverted index, and then, the index is encrypted and uploaded to the server. The encrypted index is generated with trapdoors for later search. With the search token (trapdoor), the server can search encrypted data containing the related keyword without learning either the keyword or the data. Similar to OPE and ORE, recent results have shown that certain SSE constructions may be subject to leakage-abuse attacks with various underlying adversarial assumptions [146]–[150].

8) Trusted Hardware: Trusted hardware, also known as a trusted execution environment (TEE), provides confidentiality and integrity guarantees for processes running inside the specific hardware. Here, we introduce a typical TEE design, i.e., Intel Software Guard Extension (SGX). SGX is a set of special CPU instructions along with a specially designed CPU architecture called secure enclave (enclave for short). Enclave mainly refers to a particular container with software codes, confidential data, and memory stacks. SGX places confidential information inside the CPU package, where the data and codes are transparent to the inside software, and the integrity of the software can be verified remotely. The container cannot be accessed or tampered with by either malicious adversaries or curious infrastructure owners unless opening the CPU package [119].

The enclave cannot be accessed or tempered because of its special hardware design. In SGX, there is a special memory region inside a CPU package called EPC [119]. EPC can only be accessed from the inside of the enclave. However, EPC has a constrained 256-MB space as of today. To run the normal application inside the enclave, SGX leverages the memory encryption engine to handle the data swapping between untrusted memory and EPC. The memory

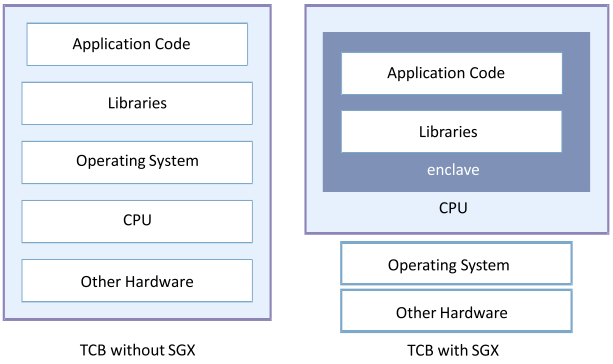


Fig. 4. ***Comparison between TCB with SGX and without SGX. The gray blue area stands for TCB.***

encryption engine encrypts traffic between the processor package and main memory and verifies its integrity. The sophisticated hardware design of SGX makes its trusted computing base (TCB) smaller than the vanilla approach. The TCB of the SGX-based approach is the enclave and the hardware itself, while the traditional application includes an operating system and virtual machine monitor inside its TCB. Fig. 4 shows the TCB area of both approaches. Programs running inside the enclave must be trustworthy to ensure that the secrets are not leaked. To achieve this, SGX leverages the attestation and sealing technology [151]. The instantiation of the enclave program can be attested at the very beginning of the deployment process, which makes the following confidential information transported to the enclave securely.

IV. PRIVACY PRESERVING NETWORK FUNCTIONS

To preserve the privacy of network communication, the clients should encrypt the packets before transmission. In the early years, to perform network functions on the encrypted traffic, the traffic is decrypted in the middle by mounting the man-in-the-middle attack on the middlebox, and then, the middlebox can inspect the decrypted traffic [66]. Although this method is intuitive and effective, decryption in the middle of the communication violates end-to-end security, which leads to privacy information leakage and brings security risks.

Network functions can be extracted as inputs that include packet contents and predefined rules and outputs that include the actions for the packet. Most network functions require matching on the packets and rules, or computation on the header of the packets. Functions such as NAT also require modification on the header. In this section, we introduce state-of-the-art approaches that achieve network functions in a privacy-preserving way by using cryptographic tools, such as symmetric encryption, HE, and MPC, and trusted hardware, such as, Intel SGX. We give an overview of the properties of the network function outsourcing schemes in Table 4.

A. Equality Match

Equality (pattern) matching is one of the most basic functions. Network functions, such as signature/watermark detection in DPI or exact IP matching in firewalls, require exact string match over a packet and predefined rules in the middlebox. Besides, matching can also be performed between packets, e.g., in-network caching for content delivery. Table 5 summarizes the main characteristics of the representative equality matching schemes.

1) Packet-Rule Matching: Generally, the rules, which can be regarded as sensitive keyword–action pairs in Snort [84], are tokenized, encrypted, and outsourced to the middlebox in the setup phase. Then, the traffic is also tokenized and encrypted by the sender. There are generally two types of packet parsing methods: delimiter-based segmentation used in [24] and window-based n-gram used in [25] and [27]. In some schemes, additional information, such as the offset of tokens, is attached to the tokens to fit more complex rules, such as multikeyword matching and domain matching. The middlebox performs the inspection on the encrypted rule database. Actions, such as accept or drop, will then be matched, and the middlebox will inform the result to the gateway of the receiver.

The main challenge of packet-rule matching is to encrypt the rules and packets while preserving the data association between the packets and rules. An intuitive idea is to encrypt the rule (watermark/signature) tokens and the packet tokens, respectively, and then compare the packet tokens with the rule tokens. In the following discussions, we introduce the OT-based schemes and searchable encryption-based schemes in detail.

a) Secure computation-based method: OT and GC can be used to exchange data privately. Sherry *et al.* [24] first introduced the GC to perform DPI on encrypted traffic without decrypting the packet. As shown in Fig. 5, the traffic is first tokenized and then encrypted through a deterministic symmetric encryption scheme, such as AES. The middlebox needs to encrypt the rules in the same way without learning the secret key of the packet. To hide the secret key from the middlebox, BlindBox designs a GC [72] to encrypt the rules by AES. Then, the encrypted rule tokens are kept in a search tree that enables logarithmic lookups. For single keyword matching, e.g., document watermarking, the middlebox checks whether there is a match between the search token and the rules in the search tree. For multiple-keyword matching, additional information, such as absolute and relative offset, is sent together with the token to the middlebox to check if the offset is a match. However, it has disadvantages on long setup time and the exposure of inspection rules to the middlebox.

To reduce the setup time in BlindBox [24], Lin *et al.* [152] replaced the AES GC with one-way hash functions and XOR functions to encrypt the rules and messages. During the setup phase, the sender randomly

Table 4 Properties of the Representative Privacy-Preserving Network Function Designs

Scheme	Year	Server	Network Functionality							Protection			
		S/M	SW	MW	RG	WC	GE	SF	UPD	HD	PL	RE	RM
BlindBox [24]	2015	single	●	●	◐	◐					●	●	
SOFA [36]	2015	single	●	●	●	●				●		◑	●
YWLW16 [25]	2016	single	●	●							●	●	●
Embark [26]	2016	single	●	●	●				●	●	●	◑	●
PNFV [37]	2016	single	●	●	●			●	●			◑	●
Splitbox [48]	2016	multiple	●	●	●					●	●	◑	●
BlindIDS [71]	2017	single	●								●	●	●
CloudDPI [27]	2017	single	●	●		●					●	●	●
SPABox [28]	2017	single	●	●	●	●					●	●	
GWJ18 [30]	2018	multiple	●					●	●		●	◑	●
PrivDPI [32]	2019	single	●	●	◐	◐					●	●	
Pine [34]	2020	single	●						●		●	●	●
GWWY20 [41]	2020	single	●	●	●	●				●		◑	●
SHVE+ [35]	2021	single	●	●		●				●	●	◑	●
Trusted Hardware-based Schemes													
KSHK15 [13]	2015	single	●	●	●	●	●		●	●		●	●
PRI [14]	2016	single	●	●	●	●	●		●		●	●	●
SGXBox [15]	2017	single	●	●	●	●	●	●	●		●	●	●
TrustedClick [16]	2017	single	●	●	●	●	●		●		●	●	●
ShieldBox [17]	2018	single	●	●	●	●	●		●		●	●	●
MAP18 [18]	2018	single	●	●	●	●	●		●	●	●	●	●
SafeBricks [19]	2018	single	●	●	●	●	●	●	●	●	●	●	●
LightBox [20]	2019	single	●	●	●	●	●	●	●	●	●	●	●
EVE [23]	2020	single	●	●	●	●	●	●	●	●	●	●	●

Explanation on some abbreviations: S/M = single server or multiple servers, SW = single keyword match, MW = multi-keyword match, RG = range match, WC = wildcard match, GE = generic network functions, SF = stateful match, UPD = updatable ruleset, HD = L2-L4 headers, PL = L4 payload, RE = protect rules from endpoints, RM = protect rules from the middelbox, ● = fully support or fully protection, ◐ = the range match can only performed on plaintext, ◑ = protect rules from one of the endpoints, where the rules may be encrypted by the trusted gateway of one of the endpoints.

PYCRO [44] and SIXPACK [46] aim at the issue of secure routing, *i.e.*, calculating the best routes privately. The functionality is different from the matching actions like firewalls and DPI.

generates m pairs of strings (k_0, k_1) for every rule of m bits. Then, the middlebox uses the OT protocol [118] to obliviously get the random strings in the rule sets and encrypts the rule by XOR the strings. The packet is encrypted similarly. Different from BlindBox, the middlebox obliviously obtains the keys from the sender and encrypts the rules by itself, not from the GC. The encryption needs no interaction, and the shared keys do not reveal the content of the forwarding packet. However, the search tree cannot be used in the process phase because the rule tokens cannot be preprepared. The matching procedure needs to compute the one-way hash and XOR function on the encrypted rules, other than checking the equivalence of the rule token and the

Table 5 Approaches of Equality Matching

Scheme	Issue	Solution	Pros	Cons
BlindBox [24]	DPI	GC	- Sublinear (with the ruleset size) inspection time.	- Impractical setup time. - The middlebox knows the rules.
LSYYC16 [152]	DPI	OT	- Short setup time. - Low communication overhead.	- Only handle single keyword matching.
PrivDPI [32]	DPI	OT	- The encrypted ruleset can be reused. - Short setup time.	- The middlebox learns the rules. - Vulnerable to brute-force attacks.
Pine [34]	DPI	Bilinear map	- Protect the rules from the middlebox. - Support rule updates.	- Only support single keyword matching.
BlindIDS [71]	IDS	DSE	- Short setup time - Lightweight primitive.	- Long detection time (linear with rule number and packet size).
YWLW16 [25]	DPI	Cuckoo hashing	- Sublinear (with the ruleset size) inspection time.	- Only support simple rules.
GWJ18 [30]	DPI	BE	- Support updatable rules and forward privacy.	- Only support simple rules.
YWWC16 [54]	RE	Cuckoo hashing	- The inspection process is efficient detection.	- The index is not updatable.
CYZW16 [55], [57]	RE	MKSE, GC	- Efficient duplicate content detection. - Consider malicious middlebox.	- One time security of garbled circuit.
REET [56]	RE	AES	- Low overhead. - Little performance degradation.	- CPU is hard to keep up when the throughput increases.
SICS [49], [153]	SFC	AES	- Strong security guarantee. - High throughput. - Support fast update.	- The client does most of the matching process.

Explanation on some abbreviations: DSE = decryptable searchable encryption, BE = broadcast encryption, RE = redundancy elimination, MKSE = multi-key searchable encryption, SFC = service function chaining, and ABE = attribute-based encryption.

message token, thus leading to a longer detection time compared with BlindBox.

A recent work, PrivDPI [32], also managed to lower the overhead caused by the one-time GC in BlindBox by providing a reusable encrypted rule generation method following the idea of a practical and simple OT protocol [154]. In PrivDPI, the encrypted packet tokens in a new session can be derived from the encrypted tokens in the last session by preserving a count table, which makes the encrypted ruleset in the middlebox reusable. However, according to Ning *et al.* [34], PrivDPI is vulnerable to brute-force attacks, where the middlebox can forge any encrypted rules by itself and then infer the content of the encrypted traffic. Ning *et al.* [34] then proposed

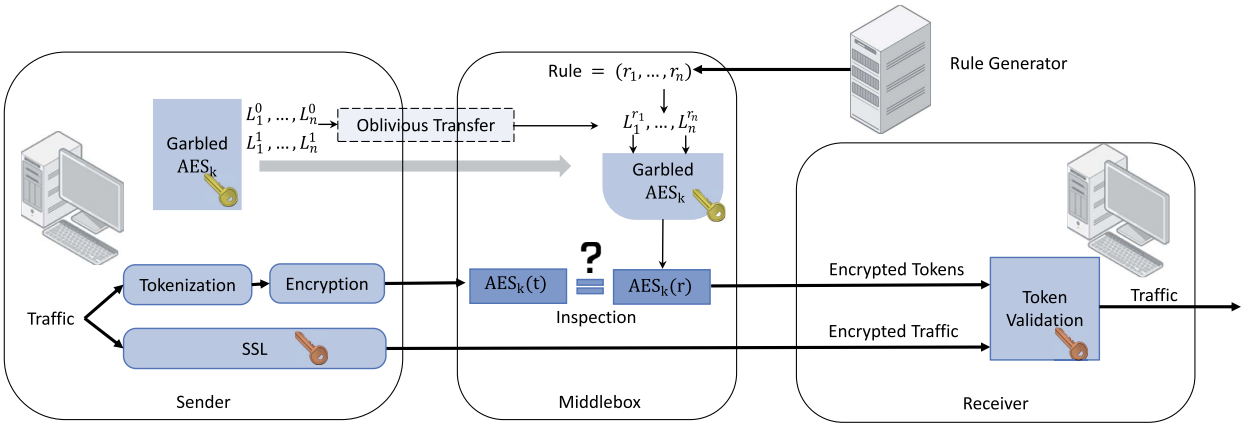


Fig. 5. ***Architecture of BlindBox [24]. In the setup phase, the endpoint prepares a garbled AES embedded with the encryption key on the tokens. With the garbled AES, the middlebox can encrypt the rules with the secret key of the tokens. The traffic is then tokenized, encrypted, and transmitted to the middlebox for inspection. In addition, for correctness assurance, the encrypted tokens and traffic will then be sent to the receiver to validate the tokens in case the sender may be malicious.***

an improved scheme called Pine, which is more efficient and secure than PrivDPI. Pine also supports updatable rule sets, i.e., additions on the rules after the prepossessing step.

b) Searchable encryption-based method: For virtual network functions that mainly focus on pattern matching and filtering, searchable encryption can be used to perform matching operations directly on encrypted traffic. Searchable encryption, especially the index-based SSE, is efficient to search for specific keywords in an encrypted rule index while preserving both the privacy of keywords and the content of the packet.

When checking whether the content of a packet includes a malicious signature, the middlebox needs to search for the malicious rule tokens in the payload tokens. Decryptable searchable encryption (DSE) [155] can be leveraged to solve this problem in a privacy-preserving way. DSE can detect whether a particular token is in a given ciphertext by using a bilinear map and XOR the results. However, once there is a match, the plaintext of the token (the keyword) would be revealed. BlindIDS [71] fixes this weakness by adding a secret key and makes sure that there is no leakage over the rules (or the keywords). During the detection, the middlebox compares every token in the encrypted traffic with each preuploaded rule token on a DSE-based construction. Unfortunately, this method suffers from considerable detection time due to the one-to-one comparison and the private-key setting. Nonetheless, compared to BlindBox [24], the above two designs, indeed, reduce the setup time by encrypting the patterns only once for all connections.

Index-based searchable encryption [114], [115], [143]–[145] is suitable and efficient for exact signature matching because of the sublinear search on the encrypted index. Yuan *et al.* [25] first used an index-based SE to realize the single keyword match on encrypted traffic. In their method, an admin server parses the rules as string–action pairs and encrypts them with trapdoors in a way that the trapdoors can be searched later. The encrypted rule index is built based on cuckoo hashing [156], [157]. The pairs are stored in two hash tables, and the locations of the pairs in the hash tables serve as the trapdoors, which can be generated from the encrypted strings in the traffic. The payloads in the traffic are parsed into strings based on predefined principles. Then, the string is transformed into a random token using a PRF with a pregenerated secret key. When the middlebox receives both the encrypted traffic and tokens, it can search for the tokens on the encrypted rule index in the middlebox. Once a token matches an entry in the filter, the middlebox takes the result action on the encrypted traffic, e.g., dropping the packet. Note that the cryptographic primitives are all symmetric so that their method can be quite efficient. Nonetheless, the encrypted rule index is static, and only equality-match rules are supported.

In the dynamic network environment, updatable rule index is essential for enterprises to upgrade their network functions, such as firewalls and DPI. Guo *et al.* [30] extended the broadcast encryption (BE) construction to support updates on the rule index, i.e., the rules for DPI can be added or deleted dynamically. However, updates may lead to more leakage than static searchable encryption schemes [148], [158]. Guo's method achieves forward privacy [159] (which prevents adversaries from inferring the keywords in newly added data with search tokens). Their scheme consists of two noncolluding servers: a rule server and a filter server. The rules are encrypted as a searchable index. During the inspection, the gateway parses the packet as a header and a message, then generates the search token, and delivers the token to the rule server to search for an encrypted message. If there is a match, the filter server will check the corresponding action and inform the gateway. For forward privacy, the entries for a string will be reencrypted under a fresh key when it appears in a newly added rule. The disadvantage of this work is that it considers watermark-like rules, and string-like rules (e.g., watermark fragments of different lengths) are not supported.

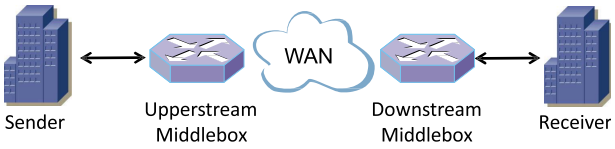


Fig. 6. ***Architecture of RE.***

2) Packet-Packet Matching: In-network caching can be well leveraged to increase the efficiency of content delivery, especially scalable media, such as images and videos. In-network caching is helpful to mitigate a large amount of video traffic. However, the cache-enabled routers are vulnerable to potential attacks and, thus, threaten the privacy of user data. Secure transport protocols, such as HTTPs, bring difficulty in leveraging in-network caching, for the confusion caused by the encryption over the traffic. Different from functions such as DPI, there is no specific rule in an in-network caching scheme, and only matching between packets is required. Generally, packets are processed and outsourced on middleboxes, such as routers, and then, they are compared with subsequent packets.

RE, which was first introduced in [96], is one of the essential functions of efficient video delivery. The architecture of RE is illustrated in Fig. 6. The video trunks are cached in a router. When the client requests a certain video trunk, he/she can first check whether there is a preloaded trunk in the router. If there is a match, the client can directly download the trunk without having to request from the video provider. Technically, the middlebox first generates fingerprints for incoming packets, tokenizes the traffic with a fixed-length sliding window before hashing each token, and, then, checks if the subset of fingerprints of a packet can match one in the fingerprint cache. If so,

the corresponding packet should be taken out from the packet cache to maximize the matching region. Otherwise, the fingerprints and the packet should be put into the cache. Taking privacy into consideration, the client may be reluctant to expose the content of the requested video to the middlebox. Thus, it is crucial to design a secure RE method to protect privacy while preserving the search function.

In the early days, Misra *et al.* [52] utilized BE to control access to the encrypted video trunks in nearby routers for different clients. Wu *et al.* [53] protected the confidentiality of the scalable video coding videos by attribute-based encryption in a content centralized network. These are the earliest attempts to achieve secure video delivery. However, Misra's work is not designed for general contexts but only for its specific content centralized network, and Wu's work does not leverage the efficient in-network cache.

Yuan *et al.* [54] first combined searchable encryption with efficient video delivery. They proposed a secure RE method to perform high-efficiency video delivery through encrypted in-network caching without revealing sensitive information about the videos. To ideally make use of the in-network caching, video chunks should be encrypted in a way that the routers can locate and access them easily without learning the content of the video. An encrypted fingerprint index is generated in advance by the application server, e.g., YouTube, and stored in a middle semi-honest request handler. The encrypted index is built from a cuckoo hashing-like method. Each video chunk has a pseudonym that will be inserted in the hash tables. The address in the hash tables is calculated by the video fingerprint and several PRFs, which can be used as the search token for the handler to locate the pseudonym in the encrypted index. When the handler succeeds in locating the related pseudonyms, it looks up the addressing table for the pseudonyms and sends the pseudonyms along with the user addresses to targeted routers. Then, the routers can forward the requested chunks to the users.

Fan *et al.* [56] proposed REET, which can support both intrauser and interuser REs over encrypted network traffic. As shown in Fig. 7, the sender encrypts fingerprints and payloads with two-layer encryption. During the first encryption, the sender uses AES to encrypt every chunk of the fingerprints. As for the second layer, the sender continues random encryption on the payload chunks. This framework uses the public-key traitor tracing scheme proposed in [160] and extends the present BE algorithm to deliver the content only to legitimate users with high-level security. Besides, they managed to cache the content at a nearby router such that valid users can receive the content even when the providers are offline.

Near-duplicate detection (NDD) is a general data caching function that can help to reuse near-duplicate data and alleviate network traffic congestion. Cui *et al.* [55], [57] proposed a secure NDD function by resorting to multi-key searchable encryption (MKSE) [161] to enable queries on encrypted content uploaded from multiple users under

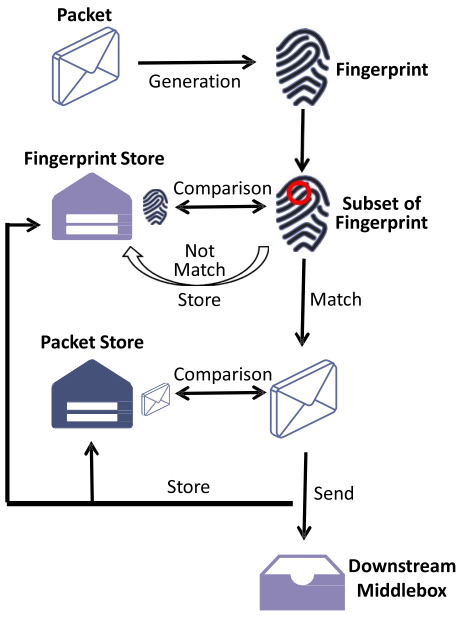


Fig. 7. *System model of REET [56].*

different secret keys. Using MKSE, the service provider can transform the user's query encrypted with her own secret key into the form of different content providers' keys. Besides, their method adopts the locality-sensitive hashing (LSH) functions [162] to label the data items, which can be regarded as the keywords in MKSE. Because LSH may bring false positives, a secure two-party computation protocol based on Yao's GC is implemented to determine whether the difference between the candidate fingerprints and the query item is small enough (i.e., the distance between the hashes generated from LSH is under an expected threshold). However, due to the one-time security of the GC, the GC needs to be refreshed every time, thus leading to a long detection delay.

B. Function-Enriched Matching

While the equality match is adequate for certain applications, it is essential to support more enriched functions, such as range match [26], [37], [40], [41], [163], [164], substring match [27]–[29], [33], wildcard match [27], [36], [38], [39], and regular expression match [28]. The rules that require enriched matching are much more complicated than single keyword matching. A simple solution is to decrypt the suspicious traffic and perform the regular expression on the plaintext, as introduced in BlindBox [24]. However, this solution discloses the content of the packet to the middlebox. In this section, we will introduce several methods that succeed in performing the function-enriched matching on encrypted packets without leaking the private data. Table 6 illustrates the main characters of the approaches of enriched matching.

Table 6 Approaches of More Enriched Matching

Scheme	Issue	Solution	Matching Type	Pros	Cons
Ladon [163]	Firewall	Bloom filter	Range	- Short construction time. - High throughput.	- Adversaries may infer the policy through long-time actions. - Misclassification due to false positive of bloom filter.
SOFA [36]	Firewall	Multilinear map	Wildcard	- Compatible with existing SDN outsourcing architecture. - Elastic with different number of rules	- The efficiency (overhead or running time) is strongly affected by the multilinearity level.
Ladon Hybrid Cloud [164]	Firewall	Bloom filter	Range	- Eliminate the result of false positive of bloom filter.	- Time-consuming computation.
SEST [29]	DPI	Bilinear map	Wildcard	- The token is not fixed and can be generated after plaintext encryption.	-The size of secret keys and public keys is linear with the plaintext size.
Embark [26]	Firewall NAT LB	PrefixMatch	Range	- Support wide range of middleboxes. - High speed. - Strong security.	- Much cost of rule updates.
SPABox [28]	DPI	Garbled DFA	Substring RE	- Support multiple query types. - No interaction between the middlebox and clients in the setup phase.	- Use complex primitives such as GC. - Do not protect the rules from the middlebox.
CloudDPI [27], [33]	DPI	Bloom filter	Substring Wildcard	- Error-free. - No hash collision.	- Too much leakage of rules (the structure of rules, number of fragments, distance between two fragments).
GWYJ18 [40]	Header matching	ORE	Range	- Flexible balance of security, space utilization, and time efficiency. - Support a large scale of firewall rules. - Modest overhead on bandwidth cost.	- Slower than the ORE designs without random permutation.
GWWY20 [41]	Header matching	ORE FSE	Range Wildcard	- Improve the efficiency of [40].	- The improved scheme only supports rules of continuous range.
SHVE+ [35]	DPI	SSE	Range Substring Wildcard	- Efficient due to the use of symmetric encryption. - Support a wide range of matching functions.	- The index is not updatable.

Explanation on some abbreviations: ORE = order-revealing encryption, FSE = fuzzy searchable encryption, RE = regular expression matching, and SSE = symmetric searchable encryption.

1) Range Matching: Range match is a common function in firewalls, NAT, and DPI, which checks whether the value in a field matches a predefined range. For example, a firewall may have a policy that requires dropping the packets whose port number is between 0 and 2048.

Khakpour and Liu [163] first attempted to outsource the firewall to the cloud privately. They proposed Ladon, a design leveraging bloom filter [165] as an efficient tool to anonymize the firewall rules. The bloom filter is a binary vector data structure that can be used to detect whether an element is a member of a set or not. The bloom filter maps data to bit vectors by multiple hash functions, with the corresponding position being 1. Ladon is built on a new data structure named the Bloom Filter Firewall Decision Diagram (BFFDD), which is a range-based decision tree generated from the general firewall decision diagrams (FDDs) [166]. Fig. 8 shows the process of BFFDD. The key idea of Ladon is to use a bloom filter to represent the edge sets in an FDD constructed from a given ACL such that the ISPs could only access anonymized firewall policies. Another work [164], however, points out that, although the cloud service providers cannot learn the exact information of the original firewall policies, they are still aware of the final decision of a certain packet. In this case, the cloud service provider can know whether a packet is good or not in a period anyway. Instead of building a set of BFFDDs to eliminate the ambiguities as Ladon did, Kurek *et al.* [164] decided to use a single BFFDD and get multiple decisions on purpose. Afterward, the packets resulting in the same decisions are processed as they are supposed to be, while packets resulting in multiple decisions need to be filtered additionally in the private cloud.

Embark [26] is the first system designed to support outsourcing a wide range of network functions. Compared with BlindBox, Embark enables more middleboxes, such as firewalls, NAT, and HTTP proxies, to be outsourced with the privacy protected using a combination of three cryptographic tools, namely, traditional AES, KeywordMatch from the BlindBox construction in [24], and their innovative scheme PrefixMatch. PrefixMatch allows an encrypted value to be compared with the encrypted endpoints of a certain range using the operators $\leq$ and $\geq$ so that the

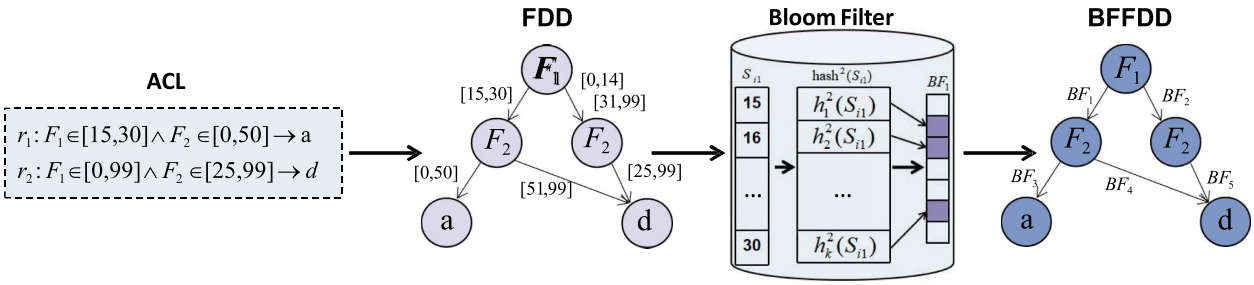


Fig. 8. ***Flowchart of BFFDD [163].***

middlebox can decide if the encrypted value is situated in the encrypted range. Specifically, when the prefixes are encrypted, the endpoints of prefixes or ranges $[s_i, e_1]$ are arranged in increasing order, and each pair of endpoints divides $P_0 = [0, 2^{\text{len}} - 1]$, where len is the size of the endpoints, into several nonoverlapping intervals I_i. The interval belonging to the same set of prefixes is assigned with one encrypted prefix, which is a random value with the same size as the prefixes. This encrypted prefix of the interval I_i is also assigned to the value $v \in I$, and the suffix of v is randomly chosen. From the perspective of performance, it is discussed in [26] that the long-lived connection between the gateway and the in-cloud middlebox saves time of handshake. In addition, compared with existing schemes based on OPE [137], [167], Embark achieves significantly better performance with a higher security level. However, we should not underestimate the prohibitive cost of rule updates in Embark, as it is possible that the new prefixes or ranges overlap with the old ones.

However, the prefix match in Embark becomes ineffective if the filtering rules cannot be represented in the form of a prefix. To tackle this issue, Guo *et al.* [40], [41] proposed an efficient ORE scheme to realize more general range matching with limited leakage. In their schemes, the header strings and rules are decoded as data values and encrypted into blocks with their order information preserved. Guo *et al.* [40], [41] also protect the location information of the blocks by randomly permuting them with searchable encryption. The extension version [41] improves the efficiency of range matching for contiguous rules. The key idea is to formulate range matching as a fuzzy search problem. The contiguous range is represented as wildcards and encrypted by fuzzy searchable encryption (FSE) [168]. The wildcard-based design only needs one round of fuzzy search, improving the efficiency greatly.

2) Substring Matching: Some rules have particular constructions, where multiple segments of unequal lengths in different locations in a packet are required to inspect [84]. For such rules, the construction information is also needed to be securely outsourced. One idea is to outsource the construction information on an encrypted index, which is similar to equality matching. SPABox [28] supports many keyword-based functions over encrypted data, including single keyword matching, keyword sequence matching (i.e., rule matching), and regular expression matching. The substring matching uses a hierarchical hash table to build the encrypted rule index. The first level restores the first five bytes in the first keyword in every rule. The value contains a pointer to another hash table whose entries are related to all possible following keyword tokens. If there is a match through all the tables, the packet is related to some rules. Compared to [24], [25], and [30], the matching process is linear with the rule size, which may be inefficient when dealing with large rule sets. For more general regular expression matching, SPABox adopts the technique of garbled-DFA [169] to do the RE matching on the receiver side without letting the receiver learn the rules.

A similar rule outsourcing method is CloudDPI [27], which uses the reversible sketch instead of cuckoo hashing to avoid insertion failures, compared to [25]. CloudDPI parses a rule into several fragments by the wildcards defined in ClamAV [170] and uses a sliding window of a specific size to segment the fragments into tokens. The tokens are then restored in the reversible sketch. Before the insertion, the hash bucket is checked through a bloom filter to ensure that the bucket is available. Each hashed token is associated with a pointer list of related signature fragments, each of which is stored together with pointers to a rule, and the previous and next fragments. The pointer lists are uploaded to the middlebox for fragment checking and rule checking. Thus, the relationship between the rules, fragments, and hashed tokens is revealed. Using the pointer lists, CloudDPI achieves complex rule types, such as substring matching and wildcards. However, CloudDPI reveals too much information about the rules, e.g., the number and repetition of fragments in a rule, and the co-occurrence of the same fragment in different rules. As an extended version of CloudDPI, Li *et al.* [33] further extended the famous ac pattern matching algorithm [171], [172] to operate on encrypted data. They replace the plaintext character, which is the input to the goto function, with the hashed token, and then, every ending state is associated with a pointer to the related signature fragment list. Because of the adoption of a finite-state pattern machine, the inspection throughput is independent of the size of the ruleset. However, the storage in the middlebox becomes

larger as the ruleset grows. Similar to CloudDPI, it also suffers from the leakage of the rule structure.

To solve the pattern matching problem where the search tokens may require arbitrary keywords of arbitrary lengths, Desmoulins *et al.* [29] designed a new searchable encryption mechanism, named Searchable Encryption with Shiftable Trapdoors (SEST). SEST is a public searchable encryption scheme that deals with substring search. Given a ciphertext and an appropriate search token, SEST can return whether the corresponding substring is in the corresponding plaintext and the positions where the pattern appears. The scheme is constructed by bilinear groups that consist of three cyclic groups and a bilinear map. The plaintext string is encrypted character by character. Every element in a string has a corresponding secret key and a public key. Based on the properties of bilinear groups, the encrypted string and keyword (in other words, the trapdoor) remain the relationship that can be elegantly designed to check the occurrence of these patterns. One of the benefits of SEST is that it can generate arbitrary trapdoors after the encryption of the original string. The search tokens can be universal because they can be generated from arbitrary keywords and used for arbitrary ciphertexts. However, the size of secret keys and public keys is linear in the plaintext size, and asymmetric cryptographic primitive is not as efficient as symmetric ones in [25] and [30]. To address this problem, Lai *et al.* [35] proposed a practical matching protocol, SHVE+, based on symmetric hidden vector encryption (SHVE) [173]. SHVE+ encodes the encrypted messages into query trapdoors of SHVE and lets the middlebox search the trapdoors on the precomputed encrypted rulesets. SHVE+ achieves better inspection performance than SEST [29] and supports a wider range of matching functions than [25], [30].

3) Wildcard Matching: There are a large number of matching rules with wildcards in firewall policies. A naive way is to label the positions of the wildcards explicitly and skip the labeled positions when matching [27], which will expose the location of wildcards. Based on the multilinear map, Shi *et al.* [36] built a framework in SDN, named Secure framework for Outsourcing FireWAll (SOFA). In SOFA, middleboxes are obfuscated by the cryptographic multilinear map before outsourcing so that the policies remain confidential from the service providers. There are two basic phases in SOFA: the obfuscation phase and the execution phase. In the obfuscation phase, the local control plane sets up parameters of the multilinear map to construct an obfuscator. In the execution phase, the cloud service provider filters inbound and outbound network traffics to execute corresponding functionality, with the detailed configurations confidentiality preserved. Sheng *et al.* [38] and Wei *et al.* [39] leverage HE to encrypt the firewall policies and support wildcard matching. In their designs, the rules are abstracted as $r = (v, W, A)$, where v denotes the bitwise value of the rule, W is the set of wildcard positions, and A is the action.

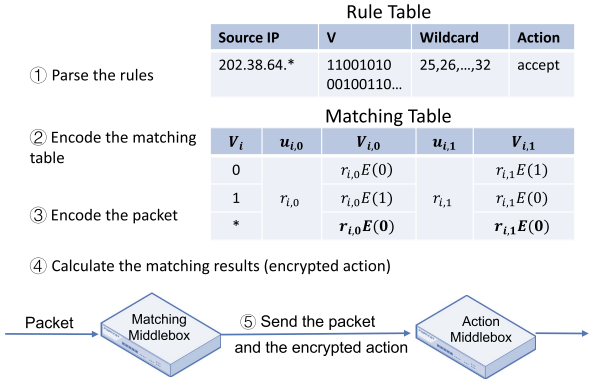


Fig. 9. ***Example of the wildcard matching scheme [38] that applies PHE to bit-level representation for the rules and packets with respect to wildcards.***

For every bit in v, the client who generates the rules needs to precompute all possible results for the corresponding value in the packet and calculate trapdoors for values 0 and 1. When obfuscating the firewall, the client will choose the same trapdoor of 0 and 1 for the bit in wildcard set W. An illustrative example is shown in Fig. 9. To hide the action, the schemes split the cloud middlebox into two: one for packet matching and the other for the action process. The packet is transmitted to the matching middlebox (which is loaded with obfuscated policies) in plaintext. If there is a match, the encrypted action will be sent to the action middlebox for decryption. The main disadvantage of the above schemes is that the content of the packet is disclosed. Furthermore, the two clouds cannot collude. Otherwise, the adversary can observe the packets, matching results, and actions to infer the policies.

C. Routing

Unlike network functions that are mainly based on rule matching (such as IDS), secure routing involves computation on the outsourced rules and encrypted packets. Here, we specifically discuss the progress on secure routing, i.e., realizing the Boarder Gateway Protocol (BGP) on encrypted data. Although the widely used BGP can well manage available routings among different groups, i.e., corporations and administrations, it has disadvantages in reliability [174], efficiency [175], and privacy [176]. For example, information such as routing policies can be inferred from BGP settings [177]. Internet eXchange Points (IXPs) provide a centralized route server (RS) service for ranking, selecting, and distributing BGP routes [178]. However, IXP members may be reluctant to distinctly forward their private routes to an RS. For ideal privacy concerns, both the routing results and the routing policies should be kept from the IXPs. Generally, route policies include import policy, next-hop policy (which includes the local preference and the shortest path computation), and export policy. The challenges lie in how

Table 7 Approaches of Secure Routing

Scheme	Solution	Pros	Cons	Domain
SIXPACK [46]	GMW	Preserve the privacy of the routes from the outsourced route servers.	The two parties need to be in very close proximity and not to collude.	Inter-domain
ADSS17 [45]	GMW	Only reveal the next hop.	Impractical performance.	Inter-domain
STRIP [43]	HE	Support the shortest path computation.	Impractical performance.	Inter-domain
PYCRO [44], [47]	HE	The first cross domain privacy-preserving routing approach.	Leak the shortest distance.	Cross-domain

to preserve both the routing policies (e.g., the local route preference) and communication information (e.g., topology information, destination, and distance information) while obtaining the best choice of routing. Different from network functions, such as DPI and firewalls, the routing function requires computation on the packets and policies, other than simple matching. Thus, most schemes are based on SMPC and HE. Table 7 demonstrates the characteristics of several representative schemes of privacy-preserving routing.

1) Secure Computation-Based Methods: In the very early years, Gupta *et al.* [42] first introduced SMPC to solve the secure interdomain routing problem. They elaborate on the advantages and challenges of the problem and explore a common cryptographic scheme to implement it. This work presents a heuristic idea. However, it does not meet practical needs. Asharov *et al.* [45] took a more in-depth insight into secure interdomain routing. They use a two-party secure Boolean circuit, Goldreich–Micali–Wigderson (GMW) [179], to calculate the routes. The GMW protocol can efficiently evaluate the same subcircuits in parallel. Asharov *et al.* [45] converted two interdomain BGP routing algorithms into MPC-based ones, which supports two routing policies: the neighbor relation-base policy [180] and neighbor preference-based policy [42]. The routing choices are calculated by precomputed AND and MUX gates implemented on GWM. Different from the above schemes, instead of designing a substitute solution for BGP, SIXPACK [46] focused more on privacy issues on IXPs. In SIXPACK, there are two noncolluding RSs: one is performed on the IXP, and the other is outside the IXP but locally close to it. The application of MPC makes sure that the RSs do not learn anything about the input and the output of the route computation, e.g., the route exportation and route selection. The routing procedure can be divided into three phases. First, taking the route policies and BGP routes as input, the RSs compute all exportable routes using the ABY framework [181] and forward the result routes to the members. Then, the members locally rank the routes and forward the ranking values to RSs. Finally, the RSs select the best route according to the next-hop ranking and their performance-related information using a MaxIdx tree circuit [182]. SIXPACK relieves the computation burden by performing the most complex ranking computation on plaintext in the IXP member instead of in the RSes where more rounds are required.

MPC-based schemes can handle various functions in routing (e.g., shortest path computation) while preserving the privacy of the routes. However, MPC-based schemes incur complex setup and bandwidth overhead due to large circuit size and the usage of OT, which could hurt the scalability of the routing designs.

2) Homomorphic Encryption-Based Methods: To deal with the shortest path computation, Henecka and Roughan [43] proposed the Secure Transitive Routing Information Protocol (STRIP), a privacy-preserving interdomain routing protocol, which allows participants (routers) to find the shortest path in a network without learning the topology information. To conceal the path lengths, the authors use the additive HE to compute the weight-sum in the shortest path computation approach, path-vector protocols (PVPs). When the origin router receives announcements from its neighbors, it sends a probe message, including essential information to all the neighbors who have announced a route to probe the path. In the path, every intermediate router adds this path length (encrypted by Paillier [116], an HE scheme) to the original one. When the destination router receives all the probe massages arriving in a limited time, it decrypts the sum of the lengths using the secret key to find the smallest one. Then, it sends a response along the shortest path until the original router receives it. In this scheme, the intermediate routers only know their last and next hops and the lengths from themselves to the next router. As for the destination router, it can only learn the last hop and the distance of the paths. In this way, topology information is protected. Besides, the distance information is also private because no one knows the distance of the paths except the destination router. Though STRIP ensures strong privacy of the routes and routing rules, it only realizes limited routing protocols, that is, it cannot cover as many routing functions as BGP does. Moreover, the HE

system, Paillier, introduces 20% extra overhead over PVP, which heavily impacts the efficiency of STRIP.

While the above schemes focus on interdomain routing, Chen *et al.* [44] proposed PYCRO, a secure cross-domain routing design that supports policy-compliant shortest path computing. To compute the shortest path, PYCRO constructs an equivalent cost graph for the significant nodes and builds a privacy-preserving shortest path tree whose root is the source switch using the cost graph. All the domain controllers use the additively HE to encrypt intradomain path lengths in their own domain and interdomain link lengths from their own domain. Then, they send them to the source domain controller to perform subsequent computation. Note that the operations above only involve addition computation and rerandomization based on the "secure-if" operation proposed by the authors (which allows choosing two options based on a condition in a privacy-preserving way). Finally, the tree is leveraged to establish the shortest path. In this protocol, only the distances from the source node to every significant node and the parent node are leaked. Similar to STRIP, PYCRO also suffers from a long delay. The extended version of PYCRO [47] improves the efficiency by performing a one-time off-line preprocessing. In this way, the execution time is reduced to 20 ms, and the computation cost is significantly saved. The communication cost is reduced since the online part is reduced to 1 kB.

D. Stateful Matching and Packet Modification

In this section, we further introduce more advanced functions, e.g., stateful matching and packet modification.

1) Stateful Packet Inspection: Stateful packet inspection, e.g., stateful firewall, is an advanced function that can inspect the traffic to detect wider illegal packets according to dynamic network states.

Melis *et al.* [37] utilized public-key encryption with keyword search (PEKS) to handle the stateful network function. The state table describing the traffic context can be regarded as a dynamic rule–action pairs, where the rules contain additional virtual fields, including "timeout" (which records the valid time of the packets) and "state" (e.g., new arriving or established). The client maintains the dynamic encrypted state table by PEKS when the network state changes. To keep the cloud from learning the packets and rules, the client (rule generator) will compute the trapdoor for necessary fields, shuffle them, and encrypt the state and action. Receiving the trapdoors, the cloud will create an entry in the encrypted state table and return the entry identifier to the client. The client can then decrypt the packet and check the ACK and FIN flags. Based on the changes in the flag, the client tells the cloud middlebox to update or delete the state table entry.

PEKS is based on asymmetric cryptographic primitives, which does not meet practical efficiency requirements. Trusted hardware is a more efficient tool and can achieve general functions. However, considerable effort is needed when dealing with the stateful process. SGX-Box [15] integrated mOS [88], which is a networking stack monitoring the stateful flow, inside the system to support stateful flow-level processing. An mOS built-in event is triggered every time the flow is changed. SGX-Box, thus, gains the ability to handle stateful processing. LightBox [20] stores the enormous states inside the disk encrypted and develops a special data structure to index and search for the needed flow table entries on disk while making the most required entries available inside EPC (a protected memory region inside CPU). With the performance bump, LightBox can perform the stateful process at near-native speed.

Note that both SGX-Box [15] and LightBox [20] rely on the trusted hardware, in particular, Intel SGX, to achieve secure network functions. As such, they are capable to support arbitrary network functions as in the plaintext domain, to be discussed in Section IV-E.

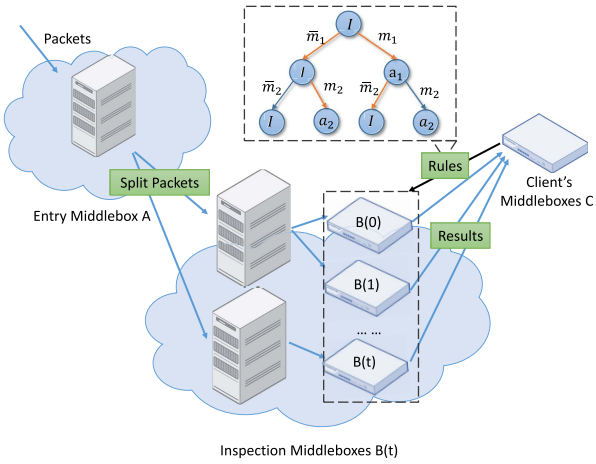


Fig. 10. ***System model of Splitbox [48].***

2) Packet Modification: Network functions, such as NAT, require modification on the packet. Intuitively, it is hard to modify an encrypted packet. To address this issue, Splitbox [48] realized modification on the content by splitting the packets into many parts and forwarding them to different middleboxes, as shown in Fig. 10. The entry middlebox A encrypts the traffic and splits the packet into t pieces and forward them to a distributed set of middleboxes. The distributed middleboxes $B(t)$ perform the matching and action process collaboratively by a t-out-of-t secret sharing scheme. The match rule is preencrypted by a trusted middlebox C (the client's middlebox) and forwarded to A, so the matching pattern is hidden. The ruleset is described as a tree, where each edge denotes a matching function $m : \{0,1\}^n \rightarrow \{0,1\}$, and each node denotes an action function $a : \{0,1\} \rightarrow \{0,1\}$. The opposite result of the matching function is denoted as $\bar{m}$. C also splits the actions using the secret-sharing method and forwards the pieces to $B(t)$. Thus, the action (how to modify the packet) is hidden. During the inspection, each

middlebox $B(t)$ will traverse the policy tree by calculating each matching result (the edge). If $m(x)$ is 1, then traverse to the left node; otherwise, traverse to the right node to perform $x = a(x)$. Finally, the final action function outputs the final packet x. The results of the middleboxes will be merged in middlebox C. Splitbox can support network function chaining due to the policy tree. However, Splitbox does not tackle the issue of packet privacy protection since the entry middlebox can directly obtain the plaintext of the traffic.

E. Generic Network Functions

Generic network functions refer to the design of the middlebox that can handle arbitrary network functions. General privacy-preserving computation tools include HE and trusted hardware, such as Intel SGX. Here, we introduce existing designs based on FHE and trusted hardware.

1) Homomorphic Encryption-Based Method: Melis *et al.* [37] put forward ways to address the private issue of general network functions. The authors abstract network functions (e.g., firewall, LB, carrier-grade NAT, IDS, and simple DPI) as a function $\psi(x)$, where the input x represents the original packet, and the output vector $\psi(x)$ represents the packet processed by the function ψ. ψ is split into a pair of functions: (m, a), where $m(x)$ denotes the match process and $a(x)$ denotes the action part. The goal of network function is that, when the result of $m(x)$ is valid, we can obtain the final result of the network functions from $a(x)$. The above ψ can be represented with the formula

$$\psi(x) = m(x) \cdot a(x) + (1 - m(x)) \cdot x.$$

Based on the above presentation of network function, Melis *et al.* [37] proposed a general private NFV (PNFV) system on FHE, with the goal of protecting the inspection rules and results. With the nature of FHE, all the computations can be performed on ciphertexts without revealing the content of the rules and results. However, in the design of PNFV, the cloud middlebox is assumed to obtain packets in plaintext to lighten the computation on the client side. Hence, the packets are not protected in this work.

2) Trusted Hardware-Based Method: Trusted hardware enables hardware support for applications to run securely. Migrating network function toward trusted hardware is natural, considering the efficiency and security of TEEs. Among the various TEE designs, Intel SGX provides a state-of-the-art choice, which includes a smaller TCB and better efficiency [119]. However, the limitation lies in the relatively small enclave space. Applications running in the secure enclave suffer from the performance penalty whenever it exceeds the space limitation.

The common workflow of network function on SGX is depicted in Fig. 11. The service provider deploys the network function application into the remote service vendor host enclave [119]. After finishing deploying a basic application (which is open to the public and can be verified by everyone), the service provider requests an attestation from the targeted enclave. Moreover, the enclave makes a report of its metadata and sends it to the Intel Quoting Enclave. The quoting enclave then verifies the report with its report key. Obtaining the quote, the service provider verifies the legitimation of the quote and establishes a secret channel with the enclave. Through this channel, the service provider secretly sends the configuration and secret information.

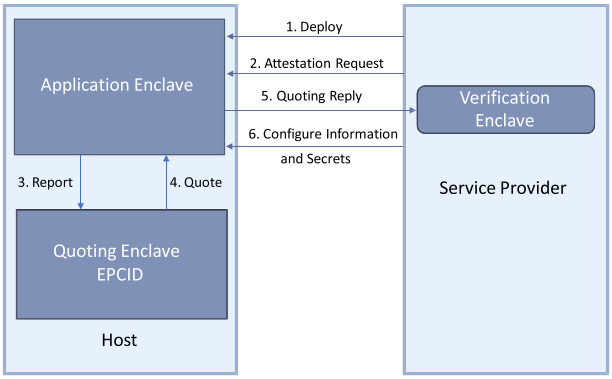


Fig. 11. *Workflow of network functions on SGX.*

Table 8 summarizes the main characteristics of several representative SGX-based designs. Kim *et al.* [13] took the first step toward leveraging a hardware-assisted approach to solve the problem of efficiency. SGX is used to verify the promise while keeping its privacy intact. Kim *et al.* [13] also demonstrated the usability of SGX in in-network functionalities of TLS sessions. They break a TLS session into two sessions. Each endpoint exchanges a unique session key with the middlebox and, thus, makes the middlebox "the Man in the Middle." Both key exchanges are finished inside the enclave and stored in a special memory region, EPC. Every packet is decrypted inside the enclave. Thus, arbitrary functions can be enforced on the payload of traffic. All packets will be encrypted and sent back to the network. Fig. 12 shows the general system model of an SGX-based scheme.

Secure as it is, the man-in-the-middle approach breaks the end-to-end encryption, which leads to a considerable modification to existing utilities. PRI [14], SGXBox [15], and TrustedClick [16] establish a secure channel between a single endpoint and the middlebox, share the session key through the secure channel with the other endpoint, and, thus, ensure the end-to-end policy. EVE [23] provides programmer-friendly Rust APIs, which makes it flexible to set the client's own strategies. These approaches follow the privacy-preserving manner and simplify the workloads to modify TLS. A very recent work, Phoenix [22], explored the possibility of achieving secure CDNs from a new perspective, i.e., protecting session keys. Phoenix is the first keyless CDN that protects both sensitive key materials

Table 8 Comparison Among Trusted Hardware-Based Approaches

Scheme	Issue	Stateful	Meta.	Header	SFC	Modification	Claimed advantages
KSHK15 [13]	Routing	✗	✗	✓	✗	N/A	- Small TCB. - Efficient.
PRI [14]	IDS	✗	✗	✗	✗	N/M	- The first DPI design supporting flexible and robust ruleset.
SGXBox [15]	IDS	✓	✗	✗	✗	Moderate	- Highly abstracted language.
ShieldBox [17]	General	✗	✗	✗	✓	High	- State persistence.
TrustedClick [16]	General	✗	✗	✗	✗	High	- Extensive use cases.
MAP18 [18]	General	✗	✗	✓	✓	High	- Versatile use cases. - High scalability.
SafeBricks [19]	IDS	✗	✗	✓	✓	High	- Extra efficiency in network function chaining provided by language feature.
LightBox [20]	General	✓	✓	✓	✓	High	- High scalability to perform stateful process. - Metadata protection. - Near-native speed.
EVE [23]	General	✓	✗	✓	✗	Moderate	- High performance. - Flexible implementation.
Phoenix [22]	CDN	N/A	N/A	N/A	N/A	Low	- Protect the private keys and session keys. - Compatible to existing web architectures.

Explanation on some abbreviations: Meta. = metadata protection, Header = header protection, SFC = service function chaining, N/A = not considered, and N/M = not implemented.

and web content. Phoenix is built on a novel design called conclaves, i.e., containers of enclaves, which enables Phoenix to perform on multiple processes and realizes the scalability. It also supports web-application firewalls and multitenancy, with only minor efforts to modify existing web architectures.

Aside from payloads, many network functions are performed on the header. Modification has to be made to enable hardware-assisted designs to support general network functions. Take the routing function as an example.

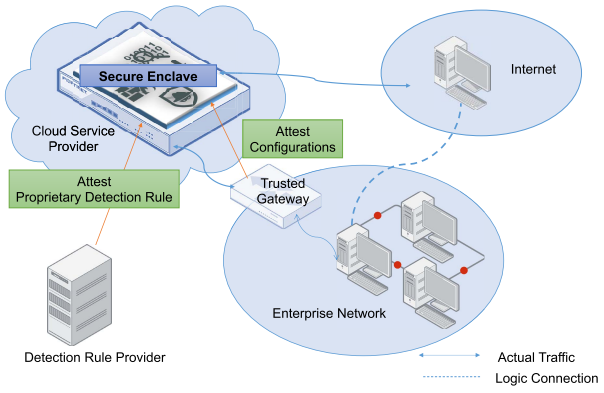


Fig. 12. ***General architecture of SGX-based schemes.***

Trusted Click [16] and EndBox [111] leveraged the Click modular router to make header reading and modification possible, where Click is a tool containing network elements to perform various tasks, such as IP tunneling configuration and ethernet switches. Trusted Click integrates the trusted hardware into Click to support arbitrary NFV applications. Protecting the sensitive information of header and metadata is essential. The work of [18] adds security extension to traditional IP and MAC protocol, providing network and link layer end-to-end security. SafeBricks [19] leverages the IPSec tunnel to protect the header. The work of Mastorakis *et al.* [18] and SafeBricks [19] use a similar approach to protect forwarding IP and MAC addresses. However, they both leak some metadata, such as packet size, count, and timestamps unprotected, which leads to potential exploits. LightBox [20] segments the flow into indistinguishable parts and encapsulates them into a secure tunnel, thus eliminating the possible metadata leakage.

Secure NFV chaining lets SGX-based middlebox achieve better performance and provide rich functions. Trach *et al.* [17] and Mastorakis *et al.* [18] leverage the Intel Data Plane Development KIT (DPDK) [183] feature to establish a secure channel between middlebox instances. LightBox [20] designed etap (enclave tap, a virtual network device similar to tun/tap) to enable NFV multithreading while tracking the flow states without

data racing. Though different network function instances are running independently on each socket, they share huge memory pages on untrusted memory. These designs adopt packet buffers stored on the shared memory page. Each network function instance gets a new packet from the receiving (Rx) rings and writes the packet back to transmitting (Tx) rings. Rx ring can be chained with Tx rings if the NF functions are chained together. SafeBricks [19] achieved NFV chaining from a different perspective. SafeBricks utilizes the Rust programming language to isolate different network functions. Rust, with a reliable compiler, makes it impossible for the irrelevant code to access private data, thus isolating different network functions inside a single enclave instance. The approach reduces extra overhead by eliminating the unnecessary process via the exchange of information inside a single enclave.

Trusted hardware-based methods preserve security and privacy while maintaining decent performance compared to software-based methods. However, due to the limitation of SGX, these approaches are vulnerable to side-channel attacks and may also suffer from the complicated implementation of existing network architecture. Specifically, SGX-based methods may require modification on the common network protocols to fit inside the enclave, which leads to an extra pile of deployment workloads.

F. Verifiable Network Functions

When the server is malicious, the middlebox may not be always faithful to follow the protocols, for the malicious intentions, such as saving computing costs or even stealing more private information. Fayazbakhsh *et al.* [59] first considered the problem conceptually and proposed a verifiable network function outsourcing design, vNFO. vNFO mainly consists of two parts, a trusted shim running inside the TPM of the cloud and a trusted central logging entity (CLE). The trusted shim samples the inward and outwards traffic and generates reports about traffic. The CLE instrument each shims instances to work properly, collects the reports, and sends them back to the consumer. As a conceptual design, vNFO does not intend to meet practical efficiency requirements.

Among the literature, there have been software and hardware methods to guarantee the integrity of network functions. The trusted hardware-based methods naturally support verification with the help of attestation and sealing technology. For example, the work of Kim *et al.* [13] runs the traditional verification program in the secure enclave. Yuan *et al.* [60], [61] utilized lightweight cryptographic primitives, such as hashing and PRFs, to assure the correctness of the inspection results and, thus, improve the likelihood that the middlebox captures the malicious packets. They proposed a ringer-based construction, which is a cryptographic sampling method to probabilistically check the correctness [184]. Ren *et al.* [63] designed a two-layer architecture with two noncollusion servers. In their design, a large scale of packets is first filtered by bloom filter, and the middlebox verifies the tokens based on an efficient no-dictionary verifiable SSE scheme [185]. The above works are practical due to the use of lightweight primitives. However, they only support verification on pattern matching.

As for more advanced middleboxes, Zhang *et al.* [50], [51] considered outsourced virtualized service function chaining (vSFC) and proposed the first scheme that could verify the correctness of the path traversal in vSFC hop by hop. This work focuses on path verification, and the correctness of the execution on middleboxes is not considered. A recent work [62] further explored the verification of stateful middleboxes. To deal with dynamic states, it adopts stateful sampling to capture the packets of the same state and then replays the packet samples locally. It is worth noting that existing works have not covered all the middleboxes, e.g., middleboxes that perform in-network caching and content delivery.

V. SYSTEMATIC COMPARISONS FROM USABILITY, PERFORMANCE, AND SECURITY

In this section, we carry out meticulous comparisons among the privacy preservation techniques and give metrics to evaluate the outsourcing network function designs. We particularly compare the techniques (e.g., cryptographic tools, such as HE and searchable encryption, and the trusted hardware, such as SGX) from usability, performance (including computational cost, communication cost, and storage problem), and security.

A. Usability

Functional property is one of the primary factors to be considered when designing a scheme. NFV includes many functions, such as firewall, DPI, NAT, and load balance. When outsourcing a network function, middleboxes may operate on the header (for functions such as firewalls, routing, and NAT), the payload (for functions such as in-network caching), or both (for functions such as DPI and IDS). Interestingly, these functions all contain the matching process. Middleboxes match the data in some field of the packet with the outsourced rules and then make the corresponding action on the packets according to the matching result. To divide the functions by the complexity of the rules, as introduced in Section IV, there are equality matching and more enriched matching, such as range matching and stateful matching.

Schemes introduced in Section IV can cover these functions in different degrees. Here, we further introduce the capability of the tools. Intuitively, methods can use cryptographic tools, such as deterministic encryption, to encrypt sensitive information to let the middlebox perform simple equality matching rules according to the hidden information and the encrypted rules on the middlebox [49]. Using more complicated multilevel encryption with the

Table 9 Estimated Performance of Different Privacy Preservation Technologies

Primitive	Computation Cost				Communication Cost	Representative Schemes
	Pre-computation‡	Setup§	Processing♯	Processing Complexity†		
HE	High	Low	High	●	N/M♮	[37]–[39]
GC	N/A♭	High	Low	○	High	[24]
Bilinear map	N/A	Moderate	Low	●	Low	[32]
OT	N/A	Low	Moderate	●	Moderate	[152]
ORE	Moderate	Low	Low	●	N/M	[40]
SSE	Moderate	Low	Low	○	Low	[25] [30]
SGX	Low	Low	Very Low	○	Low	[17] [20]

Note that, we estimate the cost levels according to the experimental results given by the authors. Considering that the experiments are conducted in different settings, the results are roughly estimated into levels of low, moderate, and high, which can reflect the efficiency of different primitives.
†: In the column of processing complexity, ● denotes that the processing time is linear with the number of the rule, and ○ denotes that the processing time is sublinear with the number of the rule.
♭: N/A means "not considered".
♮: N/M means "not mentioned".

basic deterministic AES encryption, the work of [56] can realize RE, i.e., equality matching on the packet level. Similar to deterministic encryption, index-based SSE, which is naturally suitable for searching on encrypted traffic, can be utilized to perform the equality search, such as watermark and signature matching in DPI and in-network caching. However, it is difficult for AES or index-based SSE schemes to more function-enriched matching (e.g., range matching), which are, however, quite common in many network functions, such as port and IP address range matching in the firewall. The bloom filter [163], [164] can check whether an element is included in a set, which can be modified to judge whether the data in a specific field of a packet are in the ruleset, or whether the data in a field are within a range. Prefixmatch, OPE, and ORE can reserve certain information in plaintext and, thus, can be extended to range match by encrypting the endpoints of the range and comparing the encrypted boundary with the traffic. Besides matching, sometimes, it needs more complicated computations over the traffic and the rules, e.g., network routing. HE can directly perform calculations, such as addition and multiplication on ciphertext, so it can be used to compute the shortest path and bandwidth allocation for the route decision. Also, SMPC can be utilized to calculate the routings or filtering results by multiple cooperative and middleboxes without collusion. GC and OT can provide secure information exchange among the middleboxes. The methods mentioned above are based on software, while, with the auxiliary of the secure running context provided by secure hardware, such as Intel SGX, more general functions can be performed securely. Intuitively, the packets can be decrypted and processed in the enclave. With SGX, almost all the functions can be achieved, including general DPI, NAT, and routing on L2 (data link layer) and L3 (IP layer).

Remark: We can evaluate a network function outsourcing design from a functional perspective, i.e., whether the design can achieve equality matching, more enriched matching, stateful matching, or general function. We conclude that the techniques of trusted hardware, HE, and MPC are more suitable for general and complex network functions. Other lightweight primitives can only handle relatively simple network functions. Hence, when designing in-the-cloud middleboxes, we can choose appropriate techniques according to the complexity of the network function.

B. Performance

Generally speaking, the performance of a network function is twofold: computation cost and communication cost. Besides, in the context of NF outsourcing, the storage occupation on the middlebox also needs consideration. In this section, we compare the state of the arts from the perspective of computation, communication, and storage cost, and summarize the results in Table 9.

1) Computation Cost: Computation cost mainly includes *precomputation time* in advance of all communication connections, *setup time*, and *processing time* during a connection. The middlebox needs to be configured in advance, e.g., the rules may be encrypted and uploaded at the beginning, which forms the precomputation time. The precomputation time does not affect the real-time performance for communication, while the latency strongly influences

the communication efficiency and the user experience. Latency consists of the *setup time* to establish a secure connection or prepare for secure data exchange and the *processing time*, i.e., inspection time in functions, such as DPI and firewalls, and computation time in functions, such as secure routing.

For different applications, such as long- or short-lived instant connections, the requirements for the setup time are different. Long-lived connections are more tolerable for longer setup time, while the short-lived ones are sensitive to setup time. For example, the pioneering work of outsourcing DPI, BlindBox [24], suffers from a long setup time due to the preparation for AES GC. There are lots of works aiming to achieve lower setup time. Embark [26], for example, reduces the average setup time by preserving a long-lived connection between the gateways and middleboxes instead of changing the encryption keys for every connection between the clients, which needs a fresh setup every time there is a connection. Besides, the proposed PrefixMatch in Embark is four orders of magnitude faster than OPE. In SGX-based methods [13], [17], there is a necessary remote attestation for enclave initialization, which accounts for approximately 10% of the latency in the setup phase. For example, according to the experiments in ShieldBox [17], the latency of SGX remote attestation takes about 26.4 ms.

The computational complexity of the processing phase is generally related to the number of rules. In many crypto-based schemes, the middlebox needs to match the packets with the outsourced rules one by one, i.e., linear inspection complexity [29], [37]–[39]. Sublinear matching is an advantage of index-based searchable encryption methods. For example, in BlindBox, the encrypted rules are stored in a tree-based structure, thus leading to a search time logarithmic with the number of rules. Other index-based methods [25], [28], [30] utilize hash tables to store the rules to enable sublinear inspection. Besides the time of inspection for a packet, the latency of each matching operation will also affect the processing efficiency. Generally, the SSE-based method is much more efficient than the HE-based method because SSE is based on symmetric cryptographic primitives, such as PRF. For example, the latency of index-based methods [24], [25], [28] for processing one packet over thousands of rules is within milliseconds. The HE-based method [37] takes seconds for processing a packet with five fields on ten policies, which is slower than the index-based methods. The SGX-based approach LightBox [20] takes about 20 μs to inspect a 1500-byte packet over a 5000-sized ruleset, which achieves near-native speed. Moreover, SGX-based approaches can scale linearly according to the core count. SEC-IDS [186] shows, in the experiments, that, with a limited number of flows, SEC-IDS dual-core performance is almost twice of its single core. Mastorakis *et al.* [18] demonstrated similar performance scalability with up to four enclaves running simultaneously.

2) Communication Cost: Excessive communication costs will occupy network resources, thus affecting overall efficiency. For example, one disadvantage of the GC-based BlindBox is that it uses OT for data exchange. The size of a 128-AES-circuit is about 500 kB, which brings much bandwidth for a large ruleset. Compared to BlindBox, DPFET replaces the GC with XOR, which reduces the bandwidth of AES-circuit, but the OT still consumes higher bandwidth than the lightweight cryptographic primitives. SSE-based approaches require additional transfers of tokens of the packets, which are related to the size of the packets and the tokenization method. Tokens are generally encrypted by PRF, so the size of each token is relatively small. Besides the inherent communication costs brought by the primitives, some designs also introduce multiple servers to cooperatively perform the functions, which also introduced extra communication [38], [48].

3) Storage Cost: The storage cost on the middlebox is another performance indicator. In general, outsourced rules will be kept on the storage in the middlebox. For software-based schemes, the storage requirements are not very strict because the cost of memory is not high for cloud servers. Here, we discuss the storage problem of SGX, whose memory is limited. In SGX, the programs are split into the public and secret parts, as adopted in S-NFV [187] and SheildBox [17]. LightBox [20] considers how to reduce the consumption caused by EPC paging. In LightBox, the state of each flow is tracked, and the temporarily unnecessary parts are placed outside the enclave. Meanwhile, the authors utilized a dual lookup design with cuckoo hashing, which significantly reduces the lookup time consumption due to EPC paging.

Remark: Both computational cost and communication cost are important indicators for evaluating the performance of the designs. We can measure the computation cost from three parts: the precomputation time for rule generation, the setup time for connections, and the processing time. As for communication cost, we can measure the additional bandwidth caused by search tokens in SSE, exchange messages in GC, and configuration data in SGX. When designing in-the-cloud middleboxes, we should strive to minimize the computational and communication overhead. The ultimate goal is to achieve the performance on par with local middleboxes performing on plaintext traffic.

C. Privacy and Security

The issues on privacy and security are twofold: 1) the protection of sensitive information, such as inspection rules and packets and 2) the security problems of the techniques themselves.

1) Privacy: According to Sections III-A2 and III-B, the privacy information includes the rule privacy and traffic privacy. The traffic privacy can be further subdivided

into header privacy and payload privacy. Table 4[1] presents the protection coverage of the representative privacy-preserving network functions, i.e., the protection of header/payload and the protection of rules from the endpoints/middleboxes.

In the context of network function outsourcing, the rules are generally generated by a third-party rule provider. Hence, ideally, the inspection rules should be kept from both the middlebox and the clients to protect trade secrets. BlindBox [24] protects the rules from the clients by adopting an AES-GC. However, it reveals the rules to the middlebox. In the SSE-based schemes [25], [28]–[30], the rules are encrypted and kept in hash tables to hide the private information. Note that some designs, such as [25], [26], and [48], do not explicitly protect the rules from the endpoints because the rules are generated by one of the trusted endpoints. In the context of routing, the privacy issue of rules can be further refined into three aspects: the protection of the destination address [42], topological information [43], and the routing policies, including the ranking of routes [43], [45], the shortest path [43], and export policy [42], [44], [47]. The above-mentioned designs all managed to protect the rules from the middleboxes.

As for packets, both the information in the header and the content in the payload should be protected. It is easy to protect the payload, while, for the header and the metadata, especially in the bottom layer, encryption makes it hard for the middleboxes to transfer the packets. The payload can be easily protected by encryption without affecting the low-layer packet forwarding. However, the protection of the header information is complicated. Recall that there are two outsourcing models of network functions, i.e., the go-through model and the bounce model. In the bounce model, the encrypted header can be inspected in the cloud-side middlebox and then sent back to the client's gateway. Thus, the five tuples (i.e., the source/destination IP address, the source/destination port number, and the protocol) of the original packets can be freely encrypted and filtered. The designs using the bounce model can protect the full information in the header from the cloud [36], [41]. However, in the go-through model, the packets need to be forwarded in real-world network routers. Encrypting the destination address will make it hard for the routers to forward the packets. Nonetheless, many works have managed to protect other fields in the header in different scenarios. For example, Embark [26] hides the source IP for NAT middleboxes and the URIs for HTTP headers. The HTTP header protection has a weaker security guarantee because the comparison information between the fields is revealed. SHVE+ [35] hides specific fields in the header, for example, the HTTP method for HTTP headers. In some SGX-based methods, the MAC and IP addresses are replaced with the addresses of the middlebox [18], [20]. Besides, to hide the metadata (e.g., packet size), LightBox [20] reorganizes the packet, fixes the size of each packet, and uses a streaming manner to forward the packets.

2) Security: The security of cryptographic primitives or trusted hardware is another factor affecting the security of the whole system. For example, OPE is proven to be insecure for the disclosure of the order of the encrypted fields and is vulnerable to brute force attacks [26]. SSE is a promising lightweight cryptographic technique with high efficiency. However, efficient SSE schemes generally leak access patterns and query patterns, which may disclose the linkage between the tokens in the packets and the frequency of the words in the packets and the inspection rules. Due to such leakage, SSE schemes are vulnerable to the leakage-abuse attacks [146]–[150]. It is also reported that PrivDPI [32] is vulnerable to brute-force attacks when the space of the rule set is small [34].

Different from the cryptographic primitives, SGX-based approaches are mainly faced with the problems that the system calls may be issued from an untrusted host, and the system clock may cause security flaws. As a result of the design of SGX, the attacker can easily get the enclave memory mapping. SGX-Box [15] proposed a new programming abstraction called SB lang, which encapsulates the underlying C and C++ languages, and ensures that the API does not contain any insecure pointers that are vulnerable to cache overflow attacks. As for the problem of the system clock, LightBox [20] and ShieldBox [17] adopt the etap clock and the NIC clock to avoid the attacks brought by the unsafe clock, respectively.

Remark: According to Sections III-A2 and III-B, we can evaluate the privacy-protection mechanisms by analyzing how much private information they have protected. The private information can be detailed into three parts: the L4 payload, the L2–L4 headers, and the inspection rules. The protection of inspection rules can be further divided into protection from the endpoints and protection from the middlebox. The more private information a design protects, the more secure it is.

VI. OPEN RESEARCH PROBLEMS

Albeit existing works have shown the potential of network function outsourcing, there are still challenges to design an efficient, secure, and functional outsourcing architecture. Here, we list several research challenges and potential future research directions.

A. Limitations of the Cryptographic Tools

The cryptographic tools, such as SSE, HE, and multilinear maps, can well solve the privacy problems in the outsourcing process. However, when applied to NFV, these cryptographic tools have limitations on efficiency or functionality to different degrees.

[1]Table 4 is the overview of the properties of the designs. Since the part of the protection can well help the security analysis here, we do not add a duplicate table in this section.

The SSE-based NF outsourcing designs have the advantage of high efficiency and lightweight computation and communication burden. Nonetheless, existing efficient SSE-based methods only enable the most straightforward equality matching rules, such as the signature inspection. There are potentials to combine more functional SSE techniques, such as tree-based range search [115], [188], [189] and Boolean search [190]–[192]. Interestingly, the structured encryption [193] can encrypt and query the structured data, such as graphs, which may be adopted to solve the secure routing problem, such as the shortest distant computation [194]–[196].

A drawback to HE is its poor efficiency. The nature of HE is very suitable for solving the privacy problem of NFV, i.e., computations on encrypted data. However, the FHE-based methods are not ready to be applied in practice due to their low efficiency. From the perspective of functionality, HE-based methods can only be adopted to the network functions, such as firewalls and routing, where the involved data to be computed are relatively small. For the low-efficiency applications, they could not employ HE-based designs. In order to improve, researchers can explore more efficient HE schemes in the future. As an alternative direction, it is possible to limit the NFV operations to specific operations by meticulously designing the privacy-preserving network function computation algorithms, so as to replace fully HE with efficient partial HE.

The multilinear map is a potential cryptographic tool that can be applied to the arbitrary polynomial circuit, thus enabling the obfuscation of the bit vector of the rules in firewalls in an efficient way. However, the obfuscation methods based on the multilinear map [129], [130] have been proven insecure [197]. Besides, in multilinear map-based schemes, noise is often introduced to interfere with the man-in-the-middle attacks and ensure security. However, the noise will increase rapidly with the coding level, which may affect the efficiency. How to balance the security and efficiency of the multilinear map is also a key point that we need to consider.

B. Chaining of Outsourced Network Functions

A complete network service requires the packet to go through a set of network functions [198] (a.k.a. the service chain). For example, an HTTP packet should go through a IDS $\rightarrow$ proxy chain, and packets from the internal sites should be processed through a NAT $\rightarrow$ firewall chain. When the functions are chained, the communication between the middleboxes may leak more privacy information than the single function setting. SICS [49], [153] uses deterministic encryption, such as AES to encrypt the header, which contains the destination and processing action as a label, and the middleboxes will learn the behavior (the destination and processing action) according to the label and a prebuilt label-behavior dictionary. However, the rule generator needs to enumerate all the destinations and actions, and the analysis on the header is not actually outsourced. SGX-based methods, such as SheildBox [17] and LightBox [20], can mount the functions on different virtual machines in a physical machine. So far, research on outsourcing service chains remains underexplored. There is no formal definition of security (e.g., the leakage definition) for service chaining. More tools can be used to ensure minimum information leakage when transferring among different middleboxes. It is also important to consider how to improve the flexibility and availability, i.e., update the original chain topology without affecting current business.

C. Verifiable Outsourced Network Functions

Verifiability [199] is another major research issue in the field of outsourced NFV. Outsourcing increases the cost-efficiency of NFV. Since the network function providers code to control the computing infrastructure, the consistent provision of such service can be compromised by cloud vendors so that they can gain extra profit. Oversight of such behaviors can be computationally intensive and place extra overhead on performance. Finding a proper and efficient way to detect malicious vendors forms a big problem.

Verifiability, though important, is not discussed in most schemes that we introduced before. In a privacy-preserving middlebox, the main focus is to prevent the third party from stealing precious information. The work of [13] leverages the trusted hardware to verify the correctness of network function but is vulnerable to the denial of service attack and, thus, is hard to ensure the quality of service. Besides, especially in the SSE-based method, the validation of the search tokens is indispensable if one of the endpoints is malicious. Besides the correctness of the functions performed on the server side, the integrity and consistency of the tokens and traffic should also be checked, which has not been widely studied. How to develop a middlebox, which is both private and verifiable, would be a promising yet challenging future direction.

D. Attacks on Outsourced Network Functions

To enable detection, analysis, or process on encrypted traffic with outsourced (encrypted) network functions, more information will be disclosed than the traditional unmanageable encrypted packets (e.g., HTTPs traffic). For example, the SSE-based methods may be vulnerable to the leakage-abuse attacks [141], [147], [149], [150]. The deterministic search tokens reveal the statistical distribution of the traffic. According to existing weaknesses, we can design possible attacks to infer the content of the traffic or rules. Besides cryptography-based attacks, traditional attacks, such as DoS attacks and man-in-the-middle attacks, can also be mounted on the middleboxes. Encryption makes it easier to send spam messages and occupy resources. However, very few network function outsourcing schemes consider the DoS attacks, which remains a security threat to the practical application of the schemes [200].

E. Side-Channel Attacks on SGX-Based Middleboxes

The side-channel attacks against SGX have been in a state of not being considered in the SGX-based middleboxes. Although these schemes claim that there is a general method to respond to the side-channel attack [17], [19], [20], the general countermeasures may bring problems to the performance-sensitive virtualized network functions and invalidate the functions that have been realized.

The side-channel attacks on SGX can be conducted by leveraging the TLB flushing, page faults, exceptions, and so on. Take the TLB flushing as an example. Since the operating system has direct access to memory management, a malicious OS can decide whether to flush the TLB. Then, the malicious OS can locate the attack address by analyzing the code of SGX applications and learn what the SGX has accessed by flushing the TLB and recording the memory footprint. To defend such side-channel attack, Xu *et al.* [201] proposed four general solutions: 1) disable the operating system paging, which limits the functions of enclave application; 2) enable self-paging, which requires extensive changes to the configurations, such as new hardware or new paging interfaces; 3) use the techniques of oblivious RAM or noise injection, both of which may produce extra computation and communication overhead; and 4) check the execution time. Since side-channel attacks will affect the execution efficiency, it is reasonable to check the time to find out whether an attack has occurred.

It is not hard to see that, while existing SGX-based schemes assume that conventional defense methods can evade the side-channel attacks, the countermeasures either require case-by-case constructions, incur overhead, require new hardware support and program modifications, or are not secure enough. Therefore, it is necessary to take the side-channel attacks into consideration for the SGX-based middleboxes. LightBox [20] emphasizes that a stateful middlebox may often support millions of flows concurrently, which shows the necessity of low overhead in the enclave. However, the advantage of LightBox might be whittled down if the general defense is adopted. Hence, it is essential to design specialized countermeasures to side-channel attacks for SGX-based solutions.

F. Virtual Machine Isolation for NFV

In addition to the threats from the cloud server and external adversaries, it is also significant to consider potential attacks from the tenants in the same cloud. The cloud service provider is a physical host virtual machine inevitably shared by multiple users [202]. The malicious tenant can make use of the loopholes in virtual machine isolation, such as the side-channel attacks, to steal the private data of her neighbors [73]. Currently, this problem has not been discussed in the field of network function outsourcing. The environment of virtual machines on the cloud service is much more complicated than that of the enterprise or individual level virtual machines. Therefore, how to prevent the attacks of malicious neighbor tenants in NFV from enhancing the credibility of the hypervisor is another interesting topic in the field of network function outsourcing.

VII. CONCLUSION

In this article, we have conducted a comprehensive survey on the latest literature on NFV and related security and privacy issues when moving network functions into the cloud. To better demonstrate the issue, we raised the challenges and goals of the outsourcing model. We categorized the outsourced network functions into exact matching, function-enriched matching, and very general functions, and introduced existing solutions, respectively. Cryptographic tools, such as searchable encryption, GC, and HE, and trusted hardware, such as SGX, can be utilized to design an outsourced network middlebox in a privacy-preserving way. Furthermore, we carefully compared the privacy preservation technologies from the perspective of functionality, efficiency, and security, and also concluded the metrics to evaluate the solutions. Finally, we put forward several open research problems for future investigation. ■

REFERENCES

[1] *Network Functions Virtualisation: An Introduction, Benefits, Enablers, Challenges and Call for Action*. Accessed: Nov. 2021. [Online]. Available: https://portal.etsi.org/nfv/nfv_white_paper.pdf

[2] R. Mijumbi *et al.*, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, 1st Quart., 2016.

[3] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Commun. Mag.*, vol. 53, no. 2, pp. 90–97, Feb. 2015.

[4] *Global NFV Market Size is Projected to Grow From USD 12.9 Billion in 2019 to USD 36.3 Billion by 2024, at a CAGR of 22.9%*. [Online]. Available: https://www.businesswire.com/news/home/20200127005450/en/Global-NFV-Market-Size-Projected-Grow-USD

[5] P. M. Mell and T. Grance, *The NIST Definition of Cloud Computing*, document Sp 800-145, 2011.

[6] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 5, pp. 847–859, May 2011.

[7] C.-T. Li, C.-C. Lee, and C.-Y. Weng, "A secure cloud-assisted wireless body area network in mobile emergency medical care system," *J. Med. Syst.*, vol. 40, p. 117, Mar. 2016.

[8] C.-C. Lee, S.-D. Chen, C.-T. Li, C.-L. Cheng, and Y.-M. Lai, "Security enhancement on an RFID ownership transfer protocol based on cloud," *Future Gener. Comput. Syst.*, vol. 93, pp. 266–277, Apr. 2019.

[9] *Amazon CloudFront*. Accessed: Nov. 2021. [Online]. Available: https://aws.amazon.com/cloudfront/

[10] *Azure Firewall*. Accessed: Nov. 2021. [Online]. Available: https://azure.microsoft.com/services/azure-firewall

[11] *Google Networking Services*. Accessed: Nov. 2021. [Online]. Available: https://cloud.google.com/products/networking

[12] *VMware Expands Its VMware Ready for Telco Cloud Program to Accelerate the Deployment of 5G Services on Its Telco Cloud Platform*. Accessed: Nov. 2021. [Online]. Available: https://news.vmware.com/releases/vmware-expands-its-vmware-ready-for-telco-cloud-program-to-accelerate-the-deployment-of-5g-services-on-its-telco-cloud-platform

[13] S. Kim, Y. Shin, J. Ha, T. Kim, and D. Han, "A first step towards leveraging commodity trusted execution environments for network applications," in *Proc. 14th ACM Workshop Hot Topics Netw.*, Nov. 2015, pp. 7:1–7:7.

[14] L. Schiff and S. Schmid, "PRI: Privacy preserving inspection of encrypted network traffic," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2016, pp. 296–303.

[15] J. Han, S. Kim, J. Ha, and D. Han, "SGX-Box: Enabling visibility on encrypted traffic using a secure middlebox module," in *Proc. 1st Asia–Pacific Workshop Netw.*, Aug. 2017, pp. 99–105.

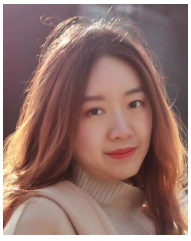
[16] M. Coughlin, E. Keller, and E. Wustrow, "Trusted

click: Overcoming security issues of NFV in the cloud," in *Proc. ACM Int. Workshop Secur. Softw. Defined Netw. Netw. Function Virtualization*, Mar. 2017, pp. 31–36.
[17] B. Trach, A. Krohmer, F. Gregor, S. Arnautov, P. Bhatotia, and C. Fetzer, "ShieldBox: Secure middleboxes using shielded execution," in *Proc. Symp. SDN Res.*, Mar. 2018, pp. 2:1–2:14.
[18] S. Mastorakis, T. Ahmed, and J. Pisharath, "ISA-based trusted network functions and server applications in the untrusted cloud," *CoRR*, vol. abs/1802.06970, pp. 1–12, Feb. 2018.
[19] R. Poddar, C. Lan, R. A. Popa, and S. Ratnasamy, "SafeBricks: Shielding network functions in the cloud," in *Proc. USENIX NSDI*, 2018, pp. 201–216.
[20] H. Duan, C. Wang, X. Yuan, Y. Zhou, Q. Wang, and K. Ren, "LightBox: Full-stack protected stateful middlebox at lightning speed," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 2351–2367.
[21] E. Marku, G. Biczok, and C. Boyd, "Towards protected VNFs for multi-operator service delivery," in *Proc. IEEE Conf. Netw. Softwarization (NetSoft)*, Jun. 2019, pp. 19–23.
[22] S. Herwig, C. Garman, and D. Levin, "Achieving keyless CDNs with conclaves," in *Proc. USENIX Secur.*, 2020, pp. 735–751.
[23] J. Han, S. Kim, D. Cho, B. Choi, J. Ha, and D. Han, "A secure middlebox framework for enabling visibility over multiple encryption protocols," *IEEE/ACM Trans. Netw.*, vol. 28, no. 6, pp. 2727–2740, Dec. 2020.
[24] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy, "BlindBox: Deep packet inspection over encrypted traffic," in *Proc. ACM Conf. Special Interest Group Data Commun.*, Aug. 2015, pp. 213–226.
[25] X. Yuan, X. Wang, J. Lin, and C. Wang, "Privacy-preserving deep packet inspection in outsourced middleboxes," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (IEEE INFOCOM)*, Apr. 2016, pp. 1–9.
[26] C. Lan, J. Sherry, R. A. Popa, S. Ratnasamy, and Z. Liu, "Embark: Securely outsourcing middleboxes to the cloud," in *Proc. USENIX NSDI*, 2016, pp. 255–273.
[27] J. Li, J. Su, X. Wang, H. Sun, and S. Chen, "CloudDPI: Cloud-based privacy-preserving deep packet inspection via reversible sketch," in *Proc. Cyberspace Saf. Secur.*, 2017, pp. 119–134.
[28] J. Fan, C. Guan, K. Ren, Y. Cui, and C. Qiao, "SPABox: Safeguarding privacy during deep packet inspection at a MiddleBox," *IEEE/ACM Trans. Netw.*, vol. 25, no. 6, pp. 3753–3766, Dec. 2017.
[29] N. Desmoulins, P. Fouque, C. Onete, and O. Sanders, "Pattern matching on encrypted streams," in *Proc. ASIACRYPT*, 2018, pp. 121–148.
[30] Y. Guo, C. Wang, and X. Jia, "Enabling secure and dynamic deep packet inspection in outsourced middleboxes," in *Proc. 6th Int. Workshop Secur. Cloud Comput.*, May 2018, pp. 49–55.
[31] H. Ren, H. Litt, D. Liu, and X. S. Shen, "Toward efficient and secure deep packet inspection for outsourced middlebox," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.
[32] J. Ning, G. S. Poh, J.-C. Loh, J. Chia, and E.-C. Chang, "PrivDPI: Privacy-preserving encrypted traffic inspection with reusable obfuscated rules," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 1657–1670.
[33] J. Li, J. Su, R. Chen, X. Wang, and S. Chen, "Practical privacy-preserving deep packet inspection outsourcing," *Concurrency Comput., Pract. Exper.*, vol. 31, no. 22, Nov. 2019.
[34] J. Ning *et al.*, "Pine: Enabling privacy-preserving deep packet inspection on TLS with rule-hiding and fast connection establishment," in *Proc. ESORICS*, 2020, pp. 3–22.
[35] S. Lai *et al.*, "Practical encrypted network traffic pattern matching for secure middleboxes," *IEEE Trans. Dependable Secure Comput.*, early access, Mar. 11, 2021, doi: 10.1109/TDSC.2021.3065652.
[36] J. Shi, Y. Zhang, and S. Zhong, "Privacy-preserving network functionality outsourcing," *CoRR*, vol. abs/1502.00389, pp. 1–9, Feb. 2015.
[37] L. Melis, H. J. Asghar, E. De Cristofaro, and M. A. Kaafar, "Private processing of outsourced network functions: Feasibility and constructions," in *Proc. ACM Int. Workshop Secur. Softw. Defined Netw. Netw. Function Virtualization*, Mar. 2016, pp. 39–44.
[38] H. Sheng, L. Wei, C. Zhang, and X. Zhang, "Privacy-preserving cloud-based firewall for IaaS-based enterprise," in *Proc. Int. Conf. Netw. Netw. Appl. (NaNA)*, Jul. 2016, pp. 206–209.
[39] L. Wei, C. Zhang, Y. Gong, Y. Fang, and K. Chen, "A firewall of two clouds: Preserving outsourced firewall policy confidentiality with heterogeneity," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.
[40] Y. Guo, C. Wang, X. Yuan, and X. Jia, "Enabling privacy-preserving header matching for outsourced middleboxes," in *Proc. IEEE/ACM 26th Int. Symp. Quality Service (IWQoS)*, Jun. 2018, pp. 1–10.
[41] Y. Guo, M. Wang, C. Wang, X. Yuan, and X. Jia, "Privacy-preserving packet header checking over in-the-cloud middleboxes," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5359–5370, Jun. 2020.
[42] D. Gupta *et al.*, "A new approach to interdomain routing based on secure multi-party computation," in *Proc. 11th ACM Workshop Hot Topics Netw. (HotNets-XI)*, 2012, pp. 37–42.
[43] W. Henecka and M. Roughan, "STRIP: Privacy-preserving vector-based routing," in *Proc. 21st IEEE Int. Conf. Netw. Protocols (ICNP)*, Oct. 2013, pp. 1–10.
[44] Q. Chen, C. Qian, and S. Zhong, "Privacy-preserving cross-domain routing optimization—A cryptographic approach," in *Proc. IEEE 23rd Int. Conf. Netw. Protocols (ICNP)*, Nov. 2015, pp. 356–365.
[45] G. Asharov *et al.*, "Privacy-preserving interdomain routing at internet scale," *Proc. Privacy Enhancing Technol.*, vol. 2017, no. 3, pp. 147–167, Jul. 2017.
[46] M. Chiesa, D. Demmler, M. Canini, M. Schapira, and T. Schneider, "SIXPACK: Securing internet eXchange points against curious onlooKers," in *Proc. 13th Int. Conf. Emerg. Netw. EXperiments Technol.*, Nov. 2017, pp. 120–133.
[47] Q. Chen, S. Shi, X. Li, C. Qian, and S. Zhong, "SDN-based privacy preserving cross domain routing," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 6, pp. 930–943, Nov. 2019.
[48] H. J. Asghar, L. Melis, C. Soldani, E. D. Cristofaro, M. A. Kâafar, and L. Mathy, "SplitBox: Toward efficient private network function virtualization," in *Proc. ACM HotMiddlebox SIGCOMM*, 2016, pp. 7–13.
[49] H. Wang, X. Li, and C. Qian, "SICS: Secure and dynamic middlebox outsourcing," in *Proc. IEEE 25th Int. Conf. Netw. Protocols (ICNP)*, Oct. 2017, pp. 1–2.
[50] X. Zhang, Q. Li, J. Wu, and J. Yang, "Generic and agile service function chain verification on cloud," in *Proc. IEEE/ACM 25th Int. Symp. Quality Service (IWQoS)*, Jun. 2017, pp. 1–10.
[51] X. Zhang, Q. Li, Z. Zhang, J. Wu, and J. Yang, "VSFC: Generic and agile verification of service function chains in the cloud," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 78–91, Feb. 2021.
[52] S. Misra, R. Tourani, and N. E. Majd, "Secure content delivery in information-centric networks: Design, implementation, and analyses," in *Proc. 3rd ACM SIGCOMM Workshop Inf.-Centric Netw. (ICN)*, 2013, pp. 73–78.
[53] Y. Wu, Z. Wei, and R. H. Deng, "Attribute-based access to scalable media in cloud-assisted content sharing networks," *IEEE Trans. Multimedia*, vol. 15, no. 4, pp. 778–788, Jun. 2013.
[54] X. Yuan *et al.*, "Enabling secure and efficient video delivery through encrypted in-network caching," *IEEE J. Sel. Area Commun.*, vol. 34, no. 8, pp. 2077–2090, Aug. 2016.
[55] H. Cui, X. Yuan, Y. Zheng, and C. Wang, "Enabling secure and effective near-duplicate detection over encrypted in-network storage," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (IEEE INFOCOM)*, Apr. 2016, pp. 1–9.
[56] J. Fan, C. Guan, K. Ren, and C. Qiao, "Middlebox-based packet-level redundancy elimination over encrypted network traffic," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1742–1753, Aug. 2018.
[57] H. Cui, X. Yuan, Y. Zheng, and C. Wang, "Towards encrypted in-network storage services with secure near-duplicate detection," *IEEE Trans. Services Comput.*, vol. 14, no. 4, pp. 998–1012, Jul. 2021.
[58] M. Zhao, W. Zhou, A. J. T. Gurney, A. Haeberlen, M. Sherr, and B. T. Loo, "Private and verifiable interdomain routing decisions," in *Proc. ACM SIGCOMM Conf. Appl., Technol., Architectures, Protocols Comput. Commun. (SIGCOMM)*, 2012, pp. 383–394.
[59] S. K. Fayazbakhsh, M. K. Reiter, and V. Sekar, "Verifiable network function outsourcing: Requirements, challenges, and roadmap," in *Proc. Workshop Hot Topics Middleboxes Netw. Function Virtualization (HotMiddlebox)*, 2013, pp. 25–30.
[60] X. Yuan, H. Duan, and C. Wang, "Bringing execution assurances of pattern matching in outsourced middleboxes," in *Proc. IEEE 24th Int. Conf. Netw. Protocols (ICNP)*, Nov. 2016, pp. 1–10.
[61] X. Yuan, H. Duan, and C. Wang, "Assuring string pattern matching in outsourced middleboxes," *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1362–1375, Jun. 2018.
[62] X. Zhang, H. Duan, C. Wang, Q. Li, and J. Wu, "Towards verifiable performance measurement over in-the-cloud middleboxes," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, Apr. 2019, pp. 1162–1170.
[63] H. Ren, H. Li, D. Liu, G. Xu, N. Cheng, and X. Shen, "Privacy-preserving efficient verifiable deep packet inspection for cloud-assisted middlebox," *IEEE Trans. Cloud Comput.*, early access, Apr. 29, 2020, doi: 10.1109/TCC.2020.2991167.
[64] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Comput.*, vol. 16, no. 1, pp. 69–73, Jan./Feb. 2012.
[65] *McAfee Network Security Platform*. Accessed: Nov. 2021. [Online]. Available: https://www.mcafee.com/enterprise/en-us/products/network-security-platform.html
[66] L. S. Huang, A. Rice, E. Ellingsen, and C. Jackson, "Analyzing forged SSL certificates in the wild," in *Proc. IEEE Symp. Secur. Privacy*, May 2014, pp. 83–97.
[67] D. Naylor *et al.*, "Multi-context TLS (mcTLS): Enabling secure in-network functionality in TLS," in *Proc. ACM Conf. Special Interest Group Data Commun.*, Aug. 2015, pp. 199–212.
[68] H. Lee *et al.*, "MaTLS: How to make TLS middlebox-aware?" in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2019.
[69] D. Naylor, R. Li, C. Gkantsidis, T. Karagiannis, and P. Steenkiste, "And then there were more: Secure communication for more than two parties," in *Proc. 13th Int. Conf. Emerg. Netw. EXperiments Technol.*, Nov. 2017, pp. 88–100.
[70] K. Bhargavan, I. Boureanu, A. Delignat-Lavaud, P.-A. Fouque, and C. Onete, "A formal treatment of accountable proxying over TLS," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 799–816.
[71] S. Canard, A. Diop, N. Kheir, M. Paindavoine, and M. Sabt, "BlindIDS: Market-compliant and privacy-friendly intrusion detection system over encrypted traffic," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, Apr. 2017, pp. 561–574.
[72] A. C. Yao, "How to generate and exchange secrets (extended abstract)," in *Proc. FOCS*, 1986, pp. 162–167.
[73] M. Pattaranantakul, R. He, Q. Song, Z. Zhang, and A. Meddahi, "NFV security survey: From use case driven threat analysis to state-of-the-art countermeasures," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3330–3368, Jul. 2018.
[74] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1617–1634, 3rd Quart., 2014.
[75] D. Kreutz, F. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.
[76] P. Zave and J. Rexford, "Patterns and interactions

in network security," *ACM Comput. Surv.*, vol. 53, no. 6, pp. 118:1–118:37, 2021.
[77] C. Wang, X. Yuan, Y. Cui, and K. Ren, "Toward secure outsourced middlebox services: Practices, challenges, and beyond," *IEEE Netw.*, vol. 32, no. 1, pp. 166–171, Jan. 2018.
[78] E. Marku, G. Biczok, and C. Boyd, "Securing outsourced VNFs: Challenges, state of the art, and future directions," *IEEE Commun. Mag.*, vol. 58, no. 7, pp. 72–77, Jul. 2020.
[79] G. S. Poh, D. M. Divakaran, H. W. Lim, J. Ning, and A. Desai, "A survey of privacy-preserving techniques for encrypted traffic inspection over network middleboxes," *CoRR*, vol. abs/2101.04338, pp. 1–17, Jan. 2021.
[80] *The Perils of Deep Packet Inspection*. Accessed: Nov. 2021. [Online]. Available: https://www.symantec.com/connect/articles/perils-deep-packet-inspection
[81] A. Bremler-Barr, Y. Harchol, D. Hay, and Y. Koral, "Deep packet inspection as a service," in *Proc. 10th ACM Int. Conf. Emerg. Netw. Exp. Technol.*, Dec. 2014, pp. 1–6.
[82] V. Paxson, "Bro: A system for detecting network intruders in real-time," in *Proc. USENIX Secur.*, 1998.
[83] V. Paxson, "Bro: A system for detecting network intruders in real-time," *Comput. Netw.*, vol. 31, nos. 23–24, pp. 2435–2463, 1999.
[84] *Snort*. Accessed: Nov. 2021. [Online]. Available: https://www.snort.org/
[85] R. Oppliger, "Internet security: Firewalls and beyond," *Commun. ACM*, vol. 40, no. 5, pp. 92–102, 1997.
[86] M. G. Gouda and A. X. Liu, "A model of stateful firewalls and its properties," in *Proc. Int. Conf. Dependable Syst. Netw. (DSN)*, 2005, pp. 128–137.
[87] *The Netfilter.org Project*. Accessed: Nov. 2021. [Online]. Available: http://www.iptables.org/
[88] M. A. Jamshed, Y. Moon, D. Kim, D. Han, and K. Park, "mOS: A reusable networking stack for flow monitoring middleboxes," in *Proc. USENIX NSDI*, 2017, pp. 113–129.
[89] L. Zhang, "A retrospective view of network address translation," *IEEE Netw.*, vol. 22, no. 5, pp. 8–12, Sep. 2008.
[90] D. Wing, "Network address translation: Extending the internet address space," *IEEE Internet Comput.*, vol. 14, no. 4, pp. 66–70, Jul. 2010.
[91] D. E. Eisenbud *et al.*, "Maglev: A fast and reliable software network load balancer," in *Proc. USENIX NSDI*, 2016, pp. 523–535.
[92] M. Rahman, S. Iqbal, and J. Gao, "Load balancer as a service in cloud computing," in *Proc. IEEE 8th Int. Symp. Service Oriented Syst. Eng.*, Apr. 2014, pp. 204–211.
[93] J. Ma, W. Rankothge, C. Makaya, M. Morales, F. Le, and J. Lobo, "A comprehensive study on load balancers for VNF chains horizontal scaling," *CoRR*, vol. abs/1810.03238, pp. 1–9, Oct. 2018.
[94] A. Vakali and G. Pallis, "Content delivery networks: Status and trends," *IEEE Internet Comput.*, vol. 7, no. 6, pp. 68–74, Nov. 2003.
[95] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Commun. Mag.*, vol. 50, no. 7, pp. 26–36, Jul. 2012.
[96] N. T. Spring and D. Wetherall, "A protocol-independent technique for eliminating redundant network traffic," in *Proc. Conf. Appl., Technol., Architectures, Protocols Comput. Commun. (SIGCOMM)*, 2000, pp. 87–95.
[97] 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. ACM SIGCOMM*, 2012, pp. 13–24.
[98] *Hulu*. Accessed: Nov. 2021. [Online]. Available: https://www.hulu.com
[99] *Radioshack Sells Customer Data After Settling With States*. Accessed: Nov. 2021. [Online]. Available: http://www.bloomberg.com/news/articles/2015-05-20/radioshackreceives-approval-to-sell-nameto-standard-general
[100] *Nearly 80% of Companies Experienced a Cloud Data Breach in Past 18 Months*. Accessed: Nov. 2021. [Online]. Available: https://www.securitymagazine.com/articles/92533-nearly-80-of-companies-experienced-a-cloud-data-breach-in-past-18-months
[101] C. Wang, Q. Wang, K. Ren, N. Cao, and W. Lou, "Toward secure and dependable storage services in cloud computing," *IEEE Trans. Services Comput.*, vol. 5, no. 2, pp. 220–232, Apr./Jun. 2012.
[102] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Trans. Comput.*, vol. 62, no. 2, pp. 362–375, Feb. 2013.
[103] Z. Ling, J. Luo, K. Wu, W. Yu, and X. Fu, "TorWard: Discovery, blocking, and traceback of malicious traffic over Tor," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 12, pp. 2515–2530, Dec. 2015.
[104] M. Nasr, A. Houmansadr, and A. Mazumdar, "Compressive traffic analysis: A new paradigm for scalable traffic analysis," in *Proc. ACM CCS*, 2017, pp. 2053–2069.
[105] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson, "Dissent in numbers: Making strong anonymity scale," in *Proc. USENIX OSDI*, 2012, pp. 179–182.
[106] C.-C. Lee, M.-S. Hwang, and I.-E. Liao, "Security enhancement on a new authentication scheme with anonymity for wireless environments," *IEEE Trans. Ind. Electron.*, vol. 53, no. 5, pp. 1683–1687, Oct. 2006.
[107] K. Seo and S. Kent, *Security Architecture for the Internet Protocol*, document RFC 4301, Internet Requests for Comments, 2005.
[108] *Tcpcrypt—Encrypting the Internet*. Accessed: Nov. 2021. [Online]. Available: http://tcpcrypt.org/
[109] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 2046–2069, Mar. 2013.
[110] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey, "Bohatei: Flexible and elastic DDoS defense," in *Proc. USENIX Secur.*, 2015, pp. 817–832.
[111] D. Goltzsche *et al.*, "EndBox: Scalable middlebox functions using client-side trusted execution," in *Proc. 48th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2018, pp. 386–397.
[112] F. Brasser, U. Müller, A. Dmitrienko, K. Kostiainen, S. Capkun, and A. Sadeghi, "Software grand exposure: SGX Cache attacks are practical," in *Proc. USENIX Workshop Offensive Technol.*, 2017.
[113] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proc. ACM CCS*, 2006, pp. 79–88.
[114] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, 2012, pp. 965–976.
[115] S. Kamara and C. Papamanthou, "Parallel and dynamic searchable symmetric encryption," in *Proc. Financial Cryptogr. Data Secur.*, 2013, pp. 258–274.
[116] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. EUROCRYPT*, 1999, pp. 223–238.
[117] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "Fully homomorphic encryption without bootstrapping," *Electron. Colloq. Comput. Complex.*, vol. 18, p. 111, Jul. 2011.
[118] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner, "More efficient oblivious transfer and extensions for faster secure computation," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2013, pp. 535–548.
[119] F. McKeen *et al.*, "Innovative instructions and software model for isolated execution," in *Proc. 2nd Int. Workshop Hardw. Architectural Secur. Privacy (HASP)*, 2013, Art. no. 10.
[120] M. Naor and B. Pinkas, "Oblivious transfer with adaptive queries," in *Proc. CRYPTO*, 1999, pp. 573–590.
[121] M. O. Rabin, "How to exchange secrets with oblivious transfer," *IACR Cryptol. ePrint Arch.*, p. 187, Jun. 2005.
[122] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 26, no. 1, pp. 96–99, Jan. 1983.
[123] R. L. Rivest, L. M. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Found. Secure Comput.*, vol. 4, no. 11, pp. 169–179, 1978.
[124] D. Stehlé and R. Steinfeld, "Faster fully homomorphic encryption," in *Proc. ASIACRYPT*, 2010, pp. 377–394.
[125] N. P. Smart and F. Vercauteren, "Fully homomorphic SIMD operations," *Des., Codes Cryptogr.*, vol. 71, no. 1, pp. 57–81, Apr. 2014.
[126] C. Gentry and S. Halevi, "Fully homomorphic encryption without squashing using depth-3 arithmetic circuits," in *Proc. IEEE 52nd Annu. Symp. Found. Comput. Sci.*, Oct. 2011, pp. 107–116.
[127] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Proc. EUROCRYPT*, 2010, pp. 24–43.
[128] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-LWE and security for key dependent messages," in *Proc. CRYPTO*, 2011, pp. 505–524.
[129] S. Garg, C. Gentry, and S. Halevi, "Candidate multilinear maps from ideal lattices and applications," *IACR Cryptol. ePrint Arch.*, p. 610, Oct. 2012.
[130] J. Coron, T. Lepoint, and M. Tibouchi, "Practical multilinear maps over the integers," in *Proc. CRYPTO*, 2013, pp. 476–493.
[131] C. Gentry, S. Gorbunov, and S. Halevi, "Graph-induced multilinear maps from lattices," *IACR Cryptol. ePrint Arch.*, p. 645, Aug. 2014.
[132] Z. Brakerski and G. N. Rothblum, "Obfuscating conjunctions," in *Proc. CRYPTO*, 2013, pp. 416–434.
[133] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order preserving encryption for numeric data," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, 2004, pp. 563–574.
[134] M. Naveed, S. Kamara, and C. V. Wright, "Inference attacks on property-preserving encrypted databases," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2015, pp. 644–655.
[135] D. Cash, F. Liu, A. O'Neill, and C. Zhang, "Reducing the leakage in practical order-revealing encryption," *IACR Cryptol. ePrint Arch.*, p. 661, Jun. 2016.
[136] J. Dyer, M. Dyer, and J. Xu, "Order-preserving encryption using approximate integer common divisors," in *Proc. DPM/CBT ESORICS*, 2017, pp. 257–274.
[137] R. A. Popa, F. H. Li, and N. Zeldovich, "An ideal-security protocol for order-preserving encoding," in *Proc. IEEE Symp. Secur. Privacy*, May 2013, pp. 463–477.
[138] N. Chenette, K. Lewi, S. A. Weis, and D. J. Wu, "Practical order-revealing encryption with limited leakage," *IACR Cryptol. ePrint Arch.*, p. 1125, Nov. 2015.
[139] K. Lewi and D. J. Wu, "Order-revealing encryption: New constructions, applications, and lower bounds," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 1167–1178.
[140] D. Boneh, K. Lewi, M. Raykova, A. Sahai, M. Zhandry, and J. Zimmerman, "Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation," in *Proc. EUROCRYPT*, 2015, pp. 563–594.
[141] P. Grubbs, K. Sekniqi, V. Bindschaedler, M. Naveed, and T. Ristenpart, "Leakage-abuse attacks against order-revealing encryption," *IACR Cryptol. ePrint Arch.*, p. 895, Sep. 2016.
[142] F. B. Durak, T. M. DuBuisson, and D. Cash, "What else is revealed by order-revealing encryption?" in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 1155–1166.
[143] D. Xiaoding Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy. (SP)*, May 2000, pp. 44–55.
[144] Q. Wang, M. He, M. Du, S. S. M. Chow, R. W. F. Lai, and Q. Zou, "Searchable encryption over feature-rich data," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 3, pp. 496–510, May/Jun. 2018.
[145] M. Du, Q. Wang, M. He, and J. Weng,

"Privacy-preserving indexing and query processing for secure dynamic cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 9, pp. 2320–2332, Sep. 2018.
[146] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: Ramification, attack and mitigation," in *Proc. NDSS*, 2012.
[147] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, "Leakage-abuse attacks against searchable encryption," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2015, pp. 668–679.
[148] Y. Zhang, J. Katz, and C. Papamanthou, "All your queries are belong to us: The power of file-injection attacks on searchable encryption," in *Proc. USENIX Secur.*, 2016, pp. 707–720.
[149] S. Patel, G. Persiano, K. Yeo, and M. Yung, "Mitigating leakage in secure cloud-hosted data structures: Volume-hiding for multi-maps via hashing," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 79–93.
[150] L. Blackstone, S. Kamara, and T. Moataz, "Revisiting leakage abuse attacks," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2020.
[151] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, "Innovative technology for CPU based attestation and sealing," in *Proc. Int. Workshop Hardw. Architectural Secur. Privacy*, 2013, pp. 1–7.
[152] Y.-H. Lin, S.-H. Shen, M.-H. Yang, D.-N. Yang, and W.-T. Chen, "Privacy-preserving deep packet filtering over encrypted traffic in software-defined networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–7.
[153] H. Wang *et al.*, "SICS: Secure and dynamic middlebox outsourcing," *IEEE/ACM Trans. Netw.*, vol. 28, no. 6, pp. 2713–2726, Dec. 2020.
[154] T. Chou and C. Orlandi, "The simplest protocol for oblivious transfer," in *Proc. LATINCRYPT*, 2015, pp. 40–58.
[155] T. Fuhr and P. Paillier, "Decryptable searchable encryption," in *Proc. Provable Secur.*, 2007, pp. 228–236.
[156] B. Fan, D. G. Andersen, M. Kaminsky, and M. D. Mitzenmacher, "Cuckoo filter: Practically better than Bloom," in *Proc. 10th ACM Int. Conf. Emerg. Netw. Exp. Technol.*, Dec. 2014, pp. 75–88.
[157] R. Pagh and F. F. Rodler, "Cuckoo hashing," *J. Algorithms*, vol. 51, no. 2, pp. 122–144, May 2004, pp. 122–144.
[158] E. Stefanov, C. Papamanthou, and E. Shi, "Practical dynamic searchable encryption with small leakage," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2014.
[159] R. Bost, "$\sum o\varphi o\varsigma$: Forward secure searchable encryption," in *Proc. ACM CCS*, 2016, pp. 1143–1154.
[160] W. Tzeng and Z. Tzeng, "A public-key traitor tracing scheme with revocation using dynamic shares," in *Proc. Int. Workshop Pract. Theory Public Key Cryptogr.*, 2001, pp. 207–224.
[161] R. A. Popa and N. Zeldovich, "Multi-key searchable encryption," *IACR Cryptol. ePrint Arch.*, p. 508, Jan. 2018.
[162] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. VLDB*, 1999, pp. 518–529.
[163] A. R. Khakpour and A. X. Liu, "First step toward cloud-based firewalling," in *Proc. IEEE 31st Symp. Reliable Distrib. Syst.*, Oct. 2012, pp. 41–50.
[164] T. Kurek, M. Niemiec, and A. Lason, "Taking back control of privacy: A novel framework for preserving cloud-based firewall policy confidentiality," *Int. J. Inf. Secur.*, vol. 15, no. 3, pp. 235–250, Jun. 2016.
[165] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.
[166] M. G. Gouda and A. X. Liu, "Structured firewall design," *Comput. Netw.*, vol. 51, no. 4, pp. 1106–1120, Mar. 2007.
[167] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill, "Order-preserving symmetric encryption," in *Proc. EUROCRYPT*, 2009, pp. 224–241.
[168] Q. Liu, S. Pei, K. Xie, J. Wu, T. Peng, and G. Wang, "Achieving secure and effective search services in cloud computing," in *Proc. IEEE TrustCom*, Aug. 2018, pp. 1386–1391.
[169] P. Mohassel, S. Niksefat, S. S. Sadeghian, and B. Sadeghiyan, "An efficient protocol for oblivious DFA evaluation and applications," in *Proc. Cryptographers' Track RSA Conf.*, 2012, pp. 398–415.
[170] *Clamav*. Accessed: Nov. 2021. [Online]. Available: https://www.clamav.net/
[171] A. V. Aho and M. J. Corasick, "Efficient string matching: An aid to bibliographic search," *Commun. ACM*, vol. 18, no. 6, pp. 333–340, Jun. 1975.
[172] B. Choi, J. Chae, M. Jamshed, K. Park, and D. Han, "DFC: Accelerating string pattern matching for network applications," in *Proc. USENIX NSDI*, 2016, pp. 551–565.
[173] S. Lai *et al.*, "Result pattern hiding searchable encryption for conjunctive queries," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 745–762.
[174] N. Kushman, S. Kandula, and D. Katabi, "Can you hear me now?!: It must be BGP," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 2, pp. 75–84, 2007.
[175] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, "Delayed internet routing convergence," *IEEE/ACM Trans. Netw.*, vol. 9, no. 3, pp. 293–306, Jun. 2001.
[176] V. Giotsas and S. Zhou, "Inferring AS relationships from BGP attributes," *CoRR*, vol. abs/1106.2417, Jun. 2011.
[177] S. Machiraju and R. H. Katz, "Leveraging BGP dynamics to reverse-engineer routing policies," Citeseer, Tech. Rep., 2006.
[178] P. Richter, G. Smaragdakis, A. Feldmann, N. Chatzis, J. Boettger, and W. Willinger, "Peering at peerings: On the role of IXP route servers," in *Proc. Conf. Internet Meas. Conf.*, Nov. 2014, pp. 31–44.
[179] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game or a completeness theorem for protocols with honest majority," in *Proc. ACM STOC*, 1987, pp. 218–229.
[180] P. Gill, M. Schapira, and S. Goldberg, "Let the market drive deployment: A strategy for transitioning to BGP security," in *Proc. ACM SIGCOMM*, 2011, pp. 14–25.
[181] D. Demmler, T. Schneider, and M. Zohner, "ABY—A framework for efficient mixed-protocol secure two-party computation," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2015.
[182] V. Kolesnikov, A. Sadeghi, and T. Schneider, "Improved garbled circuit building blocks and applications to auctions and computing minima," in *Proc. CANS*, 2009, pp. 1–20.
[183] *DPDK*. Accessed: Nov. 2021. [Online]. Available: https://www.dpdk.org
[184] P. Golle and I. Mironov, "Uncheatable distributed computations," in *Proc. Cryptographers' Track RSA Conf.*, 2001, pp. 425–440.
[185] W. Ogata and K. Kurosawa, "Efficient no-dictionary verifiable searchable symmetric encryption," in *Proc. Financial Cryptogr. Data Secur.*, 2017, pp. 498–516.
[186] D. Kuvaiskii, S. Chakrabarti, and M. Vij, "Snort intrusion detection system with Intel software guard extension (Intel SGX)," *CoRR*, vol. abs/1802.00508, Feb. 2018.
[187] M.-W. Shih, M. Kumar, T. Kim, and A. Gavrilovska, "S-NFV: Securing NFV states by using SGX," in *Proc. ACM Int. Workshop Secur. Softw. Defined Netw. Netw. Function Virtualization*, Mar. 2016, pp. 45–48.
[188] D. S. Roche, D. Apon, S. G. Choi, and A. Yerukhimovich, "POPE: Partial order preserving encoding," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 1131–1142.
[189] T. Boelter, R. Poddar, and R. A. Popa, "A secure one-roundtrip index for range queries," *IACR Cryptol. ePrint Arch.*, p. 568, Jun. 2016.
[190] D. Cash, S. Jarecki, C. S. Jutla, H. Krawczyk, M. Rosu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for Boolean queries," in *Proc. CRYPTO*, 2013, pp. 353–373.
[191] S. Faber, S. Jarecki, H. Krawczyk, Q. Nguyen, M. Rosu, and M. Steiner, "Rich queries on encrypted data: Beyond exact matches," in *Proc. ESORICS*, 2015, pp. 123–145.
[192] S. Kamara and T. Moataz, "Boolean searchable symmetric encryption with worst-case sub-linear complexity," in *Proc. EUROCRYPT*, 2017, pp. 94–124.
[193] M. Chase and S. Kamara, "Structured encryption and controlled disclosure," in *Proc. ASIACRYPT*, 2010, pp. 577–594.
[194] D. J. Wu, J. Zimmerman, J. Planul, and J. C. Mitchell, "Privacy-preserving shortest path computation," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2016.
[195] Q. Wang, K. Ren, M. Du, Q. Li, and A. Mohaisen, "SecGDB: Graph encryption for exact shortest distance queries with efficient updates," in *Proc. Financial Cryptogr. Data Secur.*, 2017, pp. 79–97.
[196] M. Du, S. Wu, Q. Wang, D. Chen, P. Jiang, and A. Mohaisen, "GraphShield: Dynamic large graphs for secure queries with forward privacy," *IEEE Trans. Knowl. Data Eng.*, early access, Sep. 18, 2020, doi: 10.1109/TKDE.2020.3024883.
[197] J. H. Cheon, K. Han, C. Lee, H. Ryu, and D. Stehlé, "Cryptanalysis of the multilinear map over the integers," in *Proc. EUROCRYPT*, 2015, pp. 3–12.
[198] Y. Zhang *et al.*, "StEERING: A software-defined networking for inline service chaining," in *Proc. 21st IEEE Int. Conf. Netw. Protocols (ICNP)*, Oct. 2013, pp. 1–10.
[199] S. Hu, C. Cai, Q. Wang, C. Wang, X. Luo, and K. Ren, "Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, Apr. 2018, pp. 792–800.
[200] K. Bock, A. Alaraj, Y. Fax, K. Hurley, E. Wustrow, and D. Levin, "Weaponizing middleboxes for TCP reflected amplification," in *Proc. USENIX Secur.*, 2021, pp. 3345–3361.
[201] Y. Xu, W. Cui, and M. Peinado, "Controlled-channel attacks: Deterministic side channels for untrusted operating systems," in *Proc. IEEE Symp. Secur. Privacy*, May 2015, pp. 640–656.
[202] G. Pék, L. Buttyán, and B. Bencsáth, "A survey of security issues in hardware virtualization," *ACM Comput. Surv.*, vol. 45, no. 3, pp. 40:1–40:34, 2013.

ABOUT THE AUTHORS



Peipei Jiang received the B.E. degree in information security from Wuhan University, Wuhan, China, in 2019. She is currently working toward the joint Ph.D. degree at the City University of Hong Kong, Hong Kong, and the School of Cyber Science and Engineering, Wuhan University.

Her research interests include network security, AI security, and applied cryptography.

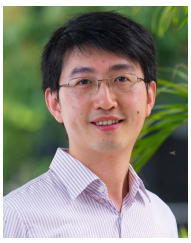
Qian Wang (Senior Member, IEEE) received the Ph.D. degree from the Illinois Institute of Technology, Chicago, IL, USA, in 2012.

He is currently a Professor with the School of Cyber Science and Engineering, Wuhan University, Wuhan, China. His research interests include AI security, data storage, search and computation outsourcing security and privacy, wireless systems security, big data security and privacy, and applied cryptography.

Dr. Wang is also a member of the Association for Computing Machinery (ACM). He received the National Science Fund for Excellent Young Scholars of China in 2018. He is also an Expert in the National 1000 Young Talents Program of China. He was a recipient of the 2016 IEEE Asia-Pacific Outstanding Young Researcher Award and the 2018 IEEE TCSC Award for Excellence in Scalable Computing (Early Career Researcher). He was a co-recipient of several best paper and best student paper awards from International Conference on Information and Communications Security (ICICS)'21, IEEE Conference on Dependable and Secure Computing (DSC)'19, IEEE International Conference on Distributed Computing Systems (ICDCS)'17, IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)'16, International Conference on Web Age Information Management (WAIM)'14, and IEEE International Conference on Network Protocols (ICNP)'11. He also serves as an Associate Editors for the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING (TDSC), the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY (TIFS), and the IEEE INTERNET OF THINGS JOURNAL (IoT-J).

Muqi Huang received the B.E. degree in computer science from Wuhan University, Wuhan, China, in 2021, where she is currently pursuing the master's degree at the School of Computer Science.

Her research interests include network security and machine learning security.



Cong Wang (Fellow, IEEE) is currently a Professor with the Department of Computer Science, City University of Hong Kong, Hong Kong. His research interests include data and network security, blockchain and decentralized applications, and privacy-enhancing technologies.

Dr. Wang is also a member of the Association for Computing Machinery (ACM). He has been one of the Founding Members of the Young Academy of Sciences of Hong Kong since 2017. He has been conferred the RGC Research Fellow in 2021. He received the Outstanding Researcher Award (Junior Faculty) in 2019, the Outstanding Supervisor Award in 2017, and the President's Awards in 2019 and 2016, all from the City University of Hong Kong. He was a co-recipient of the Best Paper Award of IEEE International Conference on Distributed Computing Systems (ICDCS)'20, International Conference on Parallel and Distributed Systems (ICPADS)'18, and International Conference on Mobile Ad-hoc and Sensor Networks (MSN)'15, the Best Student Paper Award of IEEE ICDCS'17, and the IEEE INFOCOM Test of Time Paper Award 2020. He has served as the TPC co-chair of a number of IEEE conferences and workshops. His research has been supported by multiple government research fund agencies, including the National Natural Science Foundation of China, the Hong Kong Research Grants Council, and the Hong Kong Innovation and Technology Commission. He has served as an Associate Editor for IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING (TDSC), IEEE TRANSACTIONS ON SERVICES COMPUTING (TSC), IEEE INTERNET OF THINGS JOURNAL (IoT-J), IEEE NETWORKING LETTERS, and the *Journal of Blockchain Research*.

Qi Li (Senior Member, IEEE) received the Ph.D. degree from Tsinghua University, Beijing, China.

He is currently an Associate Professor with the Institute for Network Sciences and Cyberspace, Tsinghua University. His research interests are in network and system security, particularly in the Internet and cloud security, mobile security, and big data security.

Dr. Li is also an Editorial Board Member of IEEE Transactions on Dependable and Secure Computing (TDSC) and *ACM Digital Threats: Research and Practice* (DRTAP).

Chao Shen (Senior Member, IEEE) was a Research Scholar with Carnegie Mellon University, Pittsburgh, PA, USA, from 2011 to 2013. He is currently a Professor with the School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, China. He also serves as the Associate Dean of the School of Cyber Security, Xi'an Jiaotong University. He is also with the Ministry of Education Key Laboratory for Intelligent Networks and Network Security, Xi'an Jiaotong University. His research interests include network security, human–computer interaction, insider detection, and behavioral biometrics.

Kui Ren (Fellow, IEEE) was the SUNY Empire Innovation Professor with the State University of New York at Buffalo, Buffalo, NY, USA. He is currently a Professor and the Associate Dean of the College of Computer Science and Technology, Zhejiang University (ZJU), Hangzhou, China, where he also directs the Institute of Cyber Science and Technology. He has published extensively in peer-reviewed journals and conferences. His H-index is 80, with a total citation exceeding 37 900 according to Google Scholar. His research interests include data security, IoT security, AI security, and privacy.

Dr. Ren is also a Fellow of the Association for Computing Machinery (ACM). He also serves as a member of the ACM ASIACCS Steering Committee and the S&T Committee of the Ministry of Education of China. He received many recognitions, including the Guohua Distinguished Scholar Award of ZJU, the IEEE CISTC Technical Recognition Award, the SUNY Chancellor's Research Excellence Award, the Sigma Xi Research Excellence Award, and the NSF CAREER Award. He received the IEEE INFOCOM Test of Time Paper Award in 2020 and many best paper awards from IEEE and ACM, including the ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)'20, IEEE International Conference on Distributed Computing Systems (ICDCS)'20, IEEE Global Communications Conference (Globecom)'19, ACM Asia Conference on Computer and Communications Security (AsiaCCS)'18, IEEE International Conference on Parallel and Distributed Systems (ICPADS)'18, International Workshop on Quality of Service (IWQoS)'17, and International Conference on Network Protocols (ICNP)'11. He also serves as the Chair of the Special Interest Group on Security, Audit and Control (SIGSAC) of the ACM China Council. He also serves on the editorial boards of many IEEE and ACM journals.