Toward Secure Outsourced Middlebox Services: Practices, Challenges, and Beyond

Cong Wang, Xingliang Yuan, Yong Cui, and Kui Ren

Abstract

Modern enterprise networks heavily rely on ubiquitous network middleboxes for advanced traffic processing such as deep packet inspection, traffic classification, and load balancing. Recent advances in NFV have pushed forward the paradigm of migrating in-house middleboxes to third-party providers as software-based services for reduced cost yet increased scalability. Despite its potential, this new service model also raises new security and privacy concerns, as traffic is now redirected and processed in an untrusted environment. In this article, we survey recent efforts in the direction of enabling secure outsourced middlebox functions, and identify open challenges for researchers and practitioners to further investigate solutions toward secure middlebox services.

Introduction

Network middleboxes have become fundamental components in today's Internet infrastructure. They offer a wide range of advanced network functions, such as intrusion detection [1], firewalls [2], and load balancers [3], and bring security and performance benefits to enterprises [4]. Recent efforts from industry and academia are increasingly pushing forward software middlebox appliances, termed "network function virtualization" (NFV) [5]. Along with this trend, third-party service providers such as public clouds and ISPs have begun to offer middleboxes as virtualized services [4].

While the benefits of outsourced middleboxes are well understood, such as reduced local maintenance burdens and increased service scalability, the fact that enterprises no longer have direct control of critical network functions brings up two primary security challenges. First, redirecting traffic to outsourced middleboxes for packet processing would give service providers full access to all the traffic flows and proprietary middlebox rules [6, 7], both of which might cause the sensitive information of enterprises to be unwillingly revealed. Second, unfaithful packet processing would occur in untrusted environments due to adversarial behaviors (e.g., modifying packet payloads [5] or shirking from consuming contracted resources for profits [8]).

Standard end-to-end encryption protocols such as HTTPS guarantee the confidentiality of packet content, but restrict middelbox functions. To support encrypted traffic processing, existing middlebox services use a walk-around approach that intercepts and decrypts the traffic in the middle [4, 9]. To enable middelbox functions without revealing packets and rules, Sherry *et al*. developed a secure middlebox named "BlindBox" [7] that supports privacy-preserving deep packet inspection via randomized token matching. Following this philosophy, a number of designs [3, 10–13] based on different cryptographic primitives are proposed to cover other middlebox functions, improve security or cater to various deployment scenarios.

Compared to the rapid development of secure middleboxes with privacy protection, a few recent efforts focusing on addressing unfaithful middlebox behaviors have also been made. Fayazbakhsh *et al.* made a proposal that adopts trusted hardware to generate authenticated logs to verify middlebox processing and behavior. Very recently, Yuan *et al.* proposed a tailored, sampling based mechanism to assure pattern matching based functions in outsourced middleboxes [8].

In this survey, we summarize recent work on secure outsourced middleboxes. To facilitate easy understanding of security challenges in middlebox outsourcing, we first introduce the common architectures of middlebox services, and define the threats faced in this new service model. The main content of this survey focuses on privacy-preserving middleboxes since most known secure middleboxes are designed for privacy protection. Note that protocols [9] that intercept the encrypted traffic at the middleboxes will not be included, because the middleboxes gain access to the traffic content. Specifically, we classify the related work into two categories, i.e., patten-matching and range-matching based middleboxes, in terms of the functions they support. We explain their methodologies, and thoroughly analyze their pros and cons from the perspective of security and performance. We also identify limitations and open problems to engage researchers to explore this area in the future.

Service Model and Threats

Service Architecture of Outsourced Middleboxes

Outsourced middlebox services are deployed in public clouds, ISPs or vendors that offer virtual network function appliances. Those services bring enterprises benefits in reduced cost and improved scalability in packet processing [4]. Figure 1 illustrates a common architecture of middlebox outsourcing. When endpoints from two enterprise networks are connected, the sender gateway redirects the traffic to the service provider for processing. After that, the middlebox uses techniques such as IP redirection or DNS redirection to forward the processed traffic to the receiver gateway. If an

Digital Object Identifier:
10.1109/MNET.2017.1700060

Cong Wang and Xingliang Yuan are with City University of Hong Kong; Yong Cui is with Tsinghua University; Kui Ren is with the State University of New York at Buffalo.

0890-8044/18/$25.00 © 2018 IEEE

enterprise endpoint connects to a host in the external network, the processed traffic from the middlebox will be sent back to the gateway, then sent out. Most known secure middleboxes [3, 7, 8, 10–13] comply with this architecture with traffic redirection. Other extensions such as middlebox service chaining [8] and distributed middleboxes [12] are also considered on top of the above architectures.

Threats

Using outsourced middlebox services, traffic needs to be redirected to the third-party service provider. This context introduces new threats and concerns to enterprises.

From the perspective of confidentiality, the service provider now has full access to the traffic content sent from the enterprises, which creates a broad attack surface for both inside and outside adversaries. The employees of the service provider and the hackers could extract enterprises' sensitive information by exploiting packet headers and payloads. Also, the service provider might further access the middlebox rules, which can also be proprietary. For example, a ruleset for instruction detection could sometimes be licensed from the proprietary cyber security companies [7]. A ruleset for exfiltration prevention defined by the enterprises themselves may also contain private information, such as trade secrets [3].

From the perspective of integrity, enterprises also lose direct control of middlebox executions. As mentioned before, outsourced middleboxes are not in the same trust domain as the traditional in-house settings, and they may not perform network functions as intended. For example, the service provider might intentionally operate middleboxes not to process all the packets, so as to save resources for profits [8]. They may even modify packet payloads for spurious advertisements or drop the packets due to network congestion [5].

Goals

Security: In terms of the threats, we define the security goals of secure outsourced middleboxes as follows:

- Confidentiality: the confidentiality goal includes the protection of both packet headers and payloads, and sometimes further includes the protection of middlebox rules, such as header ranges for classification or sensitive string patterns for inspection. All such information should not be divulged to the service provider.
- Integrity: the integrity goal needs to ensure that all the packets are processed as intended by outsourced middleboxes. For example, middlebox actions such as packet forwarding and legitimate dropping need to be authenticated. Also, computational functions such as packet inspection and classification should also be assured.

Performance: Networked applications have strict performance requirements. It is desirable to devise security designs with latency and throughput comparable to cleartext functions. In addition, the setup cost and bandwidth overhead introduced in security designs should also be minimized.

Deployment: The proposed protocols should be deployable in a minimally intrusive manner, and compatible with off-the-shelf outsourced middlebox service flows. Introducing additional entities or altering existing traffic flows are not preferred.

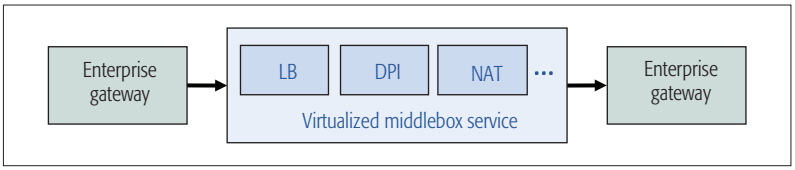

FIGURE 1. Outsourced middlebox service architecture.

Efforts on Privacy-Preserving Middleboxes

As mentioned before, most existing secure outsourced middleboxes focus on the confidentiality of rules and packets. This section summarizes these efforts. We first overview some early results that only aim at rule protection. Then we introduce a few recent representative designs that protect both rules and traffic content while still preserving middlebox functions. Those designs categorize middlebox functions into two major classes, i.e., pattern matching and range matching based functions. Based on the underlying functionality, they utilize different cryptographic primitives to design protocols that can process encrypted traffic.

In common, these designs include two primary procedures, system setup and packet processing. In the setup procedure, the proprietary rules are encrypted in a specific scheme. During the subsequent packet processing procedure, the middleboxes execute customized protocols to process incoming encrypted traffic over the encrypted rule set. Table 1 shows an overview of these designs regarding primitives, security, and performance.

Early Results

In 2012, Khakpour and Liu envisioned that more and more enterprises will outsource their IT security services such as firewalls [6]. Along with this trend, they believed that the confidentiality of firewall rules should be protected, because these rules can be useful for attackers to compromise the enterprise network, and they may also contain sensitive information about the enterprises. To protect firewall rules, they devised a method that encodes the matching conditions of the rules into a bloom filter. The rules are first transformed into decision diagrams stored in a bloom filter tree. Each tree node is a bloom filter that encodes matching conditions for a specific inspection field, e.g., source IP port range [0, 1024]. The leaf nodes indicate the corresponding actions. Then each field of an incoming packet is scanned by the bloom filter tree, and the traversal path will lead to a matched action. Afterward, Shi *et al.* [14] adopted cryptographic program obfuscation techniques (based on the candidate multilinear map constructions) and proposed a design to enable the firewall function on obfuscated rules.

We note that the above two designs only consider the protection of middlebox rules. Traffic content is in cleartext and fully exposed to the middlebox.

Pattern Matching Based Middleboxes

Pattern matching serves as a fundamental basis for a wide range of network functions such as deep packet inspection, traffic filtering, and web application firewalls. It generally refers to how a set of rules with pre-defined patterns are used to guide the middleboxes to check whether the packets contain certain patterns of interests. Based on the results of pattern matching, mid-

Schemes	Function	Primitive	Protection	Security	Throughput	Latency
KL12 [6]	Range match	Bloom filter	Rules	Weak	> 1Gb/s	< 3.5 ms
SZZ15 [14]	Range match	Multilinear map	Rules	Strong	N.A.	6-300 ms
BlindBox [7]	Pattern match	Yao's garbled circuits, symmetric searchable encryption	Rules, packets	Strong	166 Mb/s	5–33 ms
YWLW16 [10]	Pattern match	Symmetric searchable encryption	Rules, packets	Strong	315 Mb/s	~ 10 ms
Embark [3]	Range match	Prefix encryption	Rules, packets	Strong	1–10 Gb/s	< 1 ms
SplitBox [12]	Range match	Secret sharing	Rules, packets	Weak	2-8 Gb/s	10–500 ms

Note: the throughput and latency of KL12 [6] and SZZ15 [14] are measured in terms of per packet processing. In BlindBox [7], YWLW16 [10], Embark [3], and SplitBox [12], they are measured in terms of per token (field) matching.

TABLE 1. Overview of privacy-preserving middleboxes.

Networked applications have strict performance requirements. It is desirable to devise security designs with latency and throughput comparable to cleartext functions. In addition, the setup cost and bandwidth overhead introduced in security designs should also be minimized.

dleboxes then perform further actions or traffic analytics accordingly. In the literature, encrypted pattern matching can be realized via the cryptographic primitive known as "searchable encryption," which was originally designed to find tokenized keywords in encrypted documents. In terms of this observation, designs using searchable encryption schemes are proposed. In particular, both BlindBox [7] and the work by Yuan *et al*. [10] adopt the methodology to tokenize packet payloads first and then try to match these tokens against the encrypted rule patterns.

BlindBox: In [7], Sherry *et al*. propose a secure middlebox system called "BlindBox" that is the first to enable secure deep packet inspection over encrypted traffic. They consider a deployment scenario where the rules may be intellectual property known only by the rule vendors. This scenario raises a challenge on how to encrypt rules before inspection. Specifically, they adopt a secure two-party computation protocol based on Yao's garbled circuits to realize the string pattern encryption. As a result, the middlebox does not know the enterprise's private key, while the enterprise gateway does not learn the rules.

In an effort to maintain off-the-shelf HTTPS functionality, BlindBox implements a separate session just to transmit encrypted payload tokens for inspection, and develops a scheme that supports token matching over encrypted patterns. After the gateway tokenizes the payload, each token t is encrypted as: ($salt$, AES(AES(k, t), $salt$)), where $salt$ is used to hide the repeated tokens in the payloads. Then the middlebox holds the encrypted traffic, and for each incoming encrypted token, it computes AES(AES(k, r), $salt$) for all encrypted patterns AES(k, r).

BlindBox's protocols are deployed in an open source software middlebox called "Click" (the Click modular router, online at https://github.com/kohler/click.). Its inspection throughput is practically acceptable, compared to existing intrusion detection systems. However, the current design still faces the challenge of its setup delay in minutes not seconds.

The Work by Yuan *et al*. [10]: Later, Yuan *et al*. proposed new constructions for privacy-preserving deep packet inspection in outsourced middleboxes [10]. They observe that BlindBox only protects the content of rule patterns, while leaving other meta information such as inspection fields, offsets and the number of patterns in cleartext for compatibility. In certain cases, revealing such meta information would also compromise privacy. For example, certain rules might inspect specific positions of the packet payload to check the vulnerability of targeted network protocols. If one keeps the inspection positions in cleartext, the middleboxes will learn the network protocols and related applications that the endpoints run.

In light of the shortcomings from prior arts, the design by Yuan *et al*. enables the support of various kinds of inspection rules with strict rule protection. Its idea is to transform the rules into pattern-action pairs, which are then inserted into a highly customized encrypted filter. By itself, the encrypted filter leaks no information about the rules inside. The design embeds the meta information of rules into the secure tokens and protects pattern-action pairs via secure one-way functions. It also adopts secret sharing for action encryption in rules with multiple pattern matching conditions. Such a technique ensures the protection of the action, until all patterns in the corresponding rules are matched.

As the design greatly exploits symmetric cryptographic tools, it exhibits inspection throughput and latency superior to prior arts, as shown in Table 2.

Range Matching Based Middleboxes

Another common class of middlebox functions is based on range matching such as load balancers, NAT, and traffic classification. Those functions check whether packet headers such as IP addresses and port numbers are in pre-defined ranges. There are two representative designs that support secure range matching based middlebox functions over encrypted traffic. The first is the follow-up work done by the team of BlindBox, named "Embark" [3], which describes a new primitive prefix encryption for encrypted prefix matching. This primitive is efficient and compatible with known packet classification algorithms. The other work is named "SplitBox." It models range matching computations as bitwise operations between rules and packets, and then leverages a secret

sharing based secure multi-party computation protocol to allow multiple middlebox instances to jointly process the split packet shares.

Embark: The primary innovation of Embark [3] is an encryption scheme called PrefixMatch. It is designed in a way such that an encrypted packet field can be matched against encrypted prefixes or ranges using the same operators just as in the unencrypted domain, such as greater/less than and equal to. Therefore, PrefixMatch can be run over existing software and hardware implementations, achieving good backward compatibility and appealing performance.

The security of PrefixMatch is guaranteed by the randomized rules and packet headers. Although the header contents are always hidden from the cloud provider, some information leakage seems inevitable due to the desired functionality and backward compatibility. For instance, PrefixMatch supports the rules that share overlapped prefixes. Specifically, the number of identified non-overlapping intervals, and which (encrypted) prefixes overlap in each (encrypted) interval, are learned by the cloud provider; yet, the size, order or endpoints of these intervals are not revealed. We note that the impact of the above meta information leakage remains to be investigated. Whether an adversary could exploit the leakage to learn the rule content and packet headers should be answered, especially if they gained partial knowledge of the rule set.

SplitBox: There is one attempt to apply secure multi-party computation (SMC) to securely outsource middleboxes [12]. From a high-level point of view, an entry middlebox residing in one cloud handles incoming packets and splits them into secret shares to the middleboxes spanning multiple servers in another cloud. Those middleboxes process packet based on a network function abstraction via secure multi-party computation, and the partial processing results are later assembled at the local gateway for final packet transformation.

We note that SplitBox does not consider a strong threat model, and only considers external adversaries. It assumes that the cloud is not interested in the rules and packets. As long as not all the middlebox instances are compromised, the rules and packets are protected. Besides, all middleboxes essentially do the same work, which incurs undesirable bandwidth and computation resource redundancy. As such, whether the extra cost can still preserve the benefits of outsourcing remains to be further justified.

Other Studies

There exist several other studies also focusing on privacy-preserving outsourced middleboxes. In [11], Melis *et al.* propose an abstract model of network functions based on bitwise cross product, and then adopt homomorphic encryption to enable range matching in the encrypted domain. However, homomorphic encryption slows down packet inspection by orders of magnitude. To improve performance, they propose another scheme based on a public-key based searchable encryption scheme, but the security is weaker than symmetric-key based schemes. If the adversary had access to the MB processing, the scheme adopted would suffer from a brute-force attack that compromises the packet payloads [11].

Another common class of middlebox functions is based on range match such as load balancers, NAT, and traffic classification. Those functions check whether packet headers such as IP addresses and port numbers are in pre-defined ranges.

Schemes	FRS13 [5]	YDW16 [8]
Function correctness	√	√
Malicious action	X	X
Hardware-assisted	√	X
Service chain	√	√
Function scope	General	Pattern matching

TABLE 2. Comparison of verifiable middleboxes.

Apart from applying cryptographic primitives, Schiff and Schmid describe a proposal to leverage the latest trusted computing technique (Intel SGX) to enable privacy-preserving packet inspection [13]. The high-level idea is to decrypt and process the traffic inside the SGX enclave. Although this proposal is promising, it does not consider the performance challenges and limitations of SGX in practical implementations of networked contexts.

Efforts on Verifiable Middleboxes

Besides privacy-preserving middleboxes, there are also a few studies that investigate how to verify traffic processing in middleboxes. In 2013, Fayazbakhsh *et al.* wrote a position paper that identifies the challenges in achieving verifiable network function outsourcing. They also proposed a framework that leverages trusted modules to generate attested logs for replay verification [5]. To push forward the development, Yuan *et al.* [8] introduced a lightweight cryptographic sampling based mechanism to check the correctness of middlebox and middlebox service chain processing without relying on trusted hardware. Table 2 overviews the differences between these two studies. In the following subsections, we will briefly introduce their designs.

Approach Based on Trusted Hardware

The problem of verifiable network function outsourcing (NFO) is first formulated in [5]. Motivated by the need to encourage wary customers to adopt NFO, Fayazbakhsh *et al.* formally define three correctness properties characterizing functionality, performance, and accounting, which should be fulfilled by the service provider. For example, the functional correctness over a chain of middleboxes is defined as $p^{out} = f_n(\ldots f_2(f_1(p^{in}, s_1), s_2), \ldots , s_n))$, where p^{in} and p^{out} are input and output packets of the chain. The f_i refers to the i-th function which takes as input a packet and outputs another (possibly modified) packet with some valid state s_i. They also highlight and foresee challenges in achieving each of the three properties, including the enterprise's lack of visibility into the outsourced middlebox processing, dynamic and stateful nature of the middlebox, and unpredictable network effects.

Being the initial effort, they attempt the problem in a simple but sound way. At the core, a piece of trusted hardware is assumed at the service servers to log the intact packet processing events. Given the attested logs and identical local implementa-

Such overhead will greatly increase the setup time and end-to-end latency between the gateway and middlebox. In short, how to reduce the communication overhead and speed up the setup process is a demanding problem, and should carefully be solved for realistic deployment in practice.

tion of network functions, the enterprise is surely empowered to verify the correctness properties by just replaying the traffic processing events. To be practical, the authors suggest using sampling techniques to reduce logging cost and provide a tunable degree of assurance.

Approach Based on Cryptographic Primitive

Though promising, the low market penetration rate of conventional trusted hardware still hinders the adoption of hardware-assisted outsourced middleboxes. Thus, a recent design proposed by Yuan *et al*. [8] resorts to a practical cryptographic tool to assure middlebox execution. As the first effort in this direction, this work focuses on a class of computation-intensive middlebox functions (i.e., pattern matching), and a threat from incentivized service providers who perform partial traffic processing for profits. The authors introduce a lightweight mechanism to verify the correctness of these functions via a cryptographic sampling based primitive called "Ringer."

The core idea of this design is to secretly insert a few pre-computed packet processing outputs (a.k.a. ringers) in a batch of packets, before passing them to delegated network functions. Ensured by the non-invertibility of cryptographic hash functions, the ringers enable the enterprise gateway to detect cheating or lazy processing of middleboxes with tunable probability. However, a direct adoption of the ringer scheme is not sufficiently secure, since the native ringer scheme is vulnerable to a known dictionary attack if the output space of the target function is small. The authors use empirical evaluations to identify that the above vulnerability indeed exists in pattern matching based middlebox functions. To address this problem, they propose a generic approach to authenticate intermediate results of pattern matching, which enlarges their output size and exponentially increases the efforts of launching dictionary attacks. Apart from one single middlebox, the authors further propose an efficient protocol to assure the execution of ordered middleboxes in a service chain.

Challenges and Research Directions

Privacy-Preserving Middleboxes

Performance: In pattern-matching based secure middleboxes such as BlindBox and the work by Yuan *et al.* [10], tokenization introduces considerable bandwidth overhead, which can be up to 24 times larger than the size of the original traffic [7]. In BlindBox, rules are only known to the rule vendors. Thus, rule encryption based on secure two-party computation protocols also introduces large bandwidth consumption. For example, the rule encryption setup based on garbled circuits would cost 1.8 GB when encrypting just 3000 patterns, where each circuit would cost 599 KB [7]. The oblivious transfer protocol between the gateway and the middlebox further increases the communication overhead. Such overhead will greatly increase the setup time and end-to-end latency between the gateway and middlebox. In short, how to reduce the communication overhead and speed up the setup process is a demanding problem, and should carefully be solved for realistic deployment in practice.

Functionality: The encrypted token matching protocol does not support regular expression inspection, which is an important function in today's deep packet inspection engines. Embark converts some regular expressions into exact token matching, but such approaches still can hardly support all the regular expression operations. BlindBox's current treatment is to allow the middlebox to decrypt the payload for regular expressions if a match is found, while the work by Yuan *et al.* [10] requires the matched traffic to be sent back to the gateway for inspection. Regarding range matching based functions, one potential limit of Embark is that prefix matching does not serve for all the range match based network functions. For example, some classifiers have a rule with a non-contiguous mask (i.e., not a simple prefix).

In addition to pattern matching and range matching based functions, other complicated processing also emerges to handle increasingly hostile network environments, such as collaborative intrusion detection and machine learning based traffic analysis. How to achieve those functions while protecting the contents of rules and packets is also challenging.

On the other hand, most existing designs only consider stateless middleboxes, while real-world middleboxes commonly need to handle a large amount of state information during traffic processing. In [11], an interactive protocol is devised to enable the middlebox to update the state table. To prevent the middlebox from learning the states and packet contents, this preliminary protocol adopts a gateway-assisted approach. The creation and update of states need to be completed at the trusted gateway. Thus, how to design a more efficient approach is another meaningful research direction.

Verifiable Middleboxes

Although conceptually correct, the preliminary solution proposed by Fayazbakhsh *et al.* [5] does not take into consideration many practical issues. From the perspective of security, an active adversary may tamper with the packets before they touch the trusted hardware and middlebox functions, which unfortunately cannot be detected by the local verifier. In terms of performance, the attested logging at the remote server requires extensive cryptographic operations, and thus significantly slows down packet processing. Also, this approach leads to considerable local cost for packet replay and verification, undermining the benefits for middlebox outsourcing. How to address these practical issues remains to be fully explored.

The assurance mechanism for middlebox processing proposed by Yuan *et al.* [8] introduces marginal overhead at the gateway, but it is limited to computation-intensive functions. How to extend this mechanism to other middlebox functions is a meaningful research direction. Besides, the proposed design only checks the correctness of computation results. Detecting malicious actions of middleboxes should also be tackled to further improve and refine the integrity checking of outsourced middleboxes.

Hardware-Assisted Middleboxes

With the rapid advances of trusted hardware, Intel SGX shows great potential to assist in enabling secure and practical middlebox functions. In principle, both confidentiality and integrity for middlebox processing can be ensured with relatively small overhead compared to software-based approaches built on cryptographic primitives. As indicated in [16], Kim *et al.* demonstrate that Intel SGX can be used to process HTTPS traffic without introducing additional bandwidth overhead. The idea is to leverage the SGX remote attestation protocol to share the TLS key so that the SGX enclave of the middlebox instance can decrypt and process the traffic in cleartext.

However, designing SGX-enabled middleboxes faces critical performance challenges due to the expensive enclave I/O cost and the limited size of the enclave. First, some common functions such as system calls and network I/O are prohibited in the SGX enclave, while modules in middlebox systems usually require these function calls. Therefore, how to partition middlebox modules and which modules should be ported to SGX remain to be explored. Second, how to handle a large volume of network traffic while minimizing the overhead is also challenging. Processing packets inside the enclave requires using SGX secure API (ECALL/OCALL) for packet I/O, which will involve frequent context switches, and introduce considerable cost for dealing with streamed packets [16]. This issue would further be amplified in the context of network effects such as packet loss or out-of-order. We argue that SGX-assisted middlebox systems will not be practically deployable in the near future until the above challenges are addressed.

Conclusion

In this article, we survey the recent advances in secure outsourced middleboxes. Meanwhile, we clearly identify the limitations of existing studies, and observe a number of emerging research problems in this direction. Our goal is to bring our visions to researchers and practitioners to design trustworthy outsourced middlebox services, and push forward the development of network function virtualization technologies in both academia and industry.

Our goal is to bring our visions to researchers and practitioners to design trustworthy outsourced middlebox services, and push forward the development of network function virtualization technologies in both academia and industry.

Acknowledgment

This work was supported in part by the Research Grants Council of HK (Project No. CityU 11276816 and No. CityU C1008-16G), the Natural Science Foundation of China under Project 61572412, and the Innovation and Technology Commission of HK under ITF Project ITS/307/15. Prof. Yong Cui's research is supported in part by the National Natural Science Foundation of China under Project 61422206. Prof. Kui Ren's research is supported in part by the US National Science Foundation under grant CNS-1262277.

References

[1] H. Nakayama, A. Jamalipour, and N. Kato, "Network-Based Traitor-Tracing Technique Using Traffic Pattern," *IEEE Trans. Parallel Distrib. Syst.*, vol. 5, no. 2, 2010, pp. 300–13.
[2] H. Nishiyama *et al.*, "Traffic Pattern-Based Content Leakage Detection for Trusted Content Delivery Networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, 2014, pp. 301–09.
[3] C. Lan *et al.*, "Embark: Securely Outsourcing Middleboxes to the Cloud," *Proc. USENIX NSDI*, 2016.
[4] J. Sherry *et al.*, "Making Middleboxes Someone Else's Problem: Network Processing as a Cloud Service," *Proc. ACM SIGCOMM*, 2012.
[5] S. Fayazbakhsh, M. Reiter, and V. Sekar, "Verifiable Network Function Outsourcing: Requirements, Challenges, and Roadmap," *Proc. ACM HotMiddlebox*, 2013.
[6] A. Khakpour and A. Liu, "First Step Toward Cloud-Based Firewalling," *Proc. IEEE SRDS*, 2012.
[7] J. Sherry *et al.*, "Blindbox: Deep Packet Inspection for Encrypted Traffic," *Proc. ACM SIGCOMM*, 2015.
[8] X. Yuan, H. Duan, and C. Wang, "Bringing Execution Assurances of Pattern Matching in Outsourced Middleboxes," *Proc. IEEE ICNP*, 2016.
[9] D. Naylor *et al.*, "Multi-Context TLS (MCTLS): Enabling Secure In-Network Functionality in TLS," *Proc. ACM SIGCOMM*, 2015.
[10] X. Yuan *et al.*, "Privacy-Preserving Deep Packet Inspection in Outsourced Middleboxes," *Proc. IEEE INFOCOM*, 2016.
[11] L. Melis *et al.*, "Private Processing of Outsourced Network Functions: Feasibility and Constructions," *Proc. ACM Int'l. Wksp. Security Software Defined Networks & Network Function Virtualization*, 2016.
[12] H. Asghar *et al.*, "Splitbox: Toward Efficient Private Network Function Virtualization," *Proc. ACM HotMiddlebox*, 2016.
[13] L. Schiff and S. Schmid, "PRI: Privacy Preserving Inspection of Encrypted Network Traffic," *Proc. IEEE S&P Wksp. Research for Insider Threats*, 2016.
[14] J. Shi, Y. Zhang, and S. Zhong, "Privacy-Preserving Network Functionality Outsourcing," arXiv preprint arXiv:1502.00389, 2015.
[15] S. Kim *et al.*, "A First Step Towards Leveraging Commodity Trusted Execution Environments for Network Applications," *Proc. ACM HotNet*, 2015.

Biographies

Xingliang Yuan is a research fellow in the Department of Computer Science at City University of Hong Kong. He received his B.S. degree from Nanjing University of Posts and Telecommunications in 2008, the M.S. degree from Illinois Institute of Technology in 2009, both in electrical engineering, and the Ph.D. degree in computer science from City University of Hong Kong in 2016. His research interests include cloud security and network security with a current focus on search encrypted data and NFV seccurity.

Cong Wang is an assistant professor in the Computer Science Department at City University of Hong Kong. He received his B.E. and M.E. degrees from Wuhan University in 2004 and 2007, and the Ph.D. degree from Illinois Institute of Technology in 2012, all in electrical and computer engineering. He worked at the Palo Alto Research Center in the summer of 2011. His research interests are in the areas of cloud and network security, computer and mobile social networks, and secure and practical systems.

Yong Cui received the B.E. and Ph.D. degrees both on computer science and engineering from Tsinghua University. He is currently a full professor with the Computer Science Department at Tsinghua University. Co-chairing an IETF WG, he served or serves on the editorial boards on IEEE *IEEE Trans. Parallel Distrib. Syst.*, *IEEE Trans. Cloud Computing*, *IEEE Network* and *IEEE Internet Computing*. He has published over 100 papers, with several Best Paper Awards and several Internet standard documents (RFC). His research interests include mobile cloud computing and network architecture.

Kui Ren is a professor of computer science and engineering and the director of UbiSeC Lab at the State University of New York at Buffalo (UB). He received his Ph.D. degree from Worcester Polytechnic Institute. Kui's current research interests span cloud and outsourcing security, wireless and wearable systems security, and mobile sensing and crowdsourcing. His research has been supported by NSF, DoE, AFRL, MSR, and Amazon. Kui has published extensively in peer reviewed journals and conferences and received several Best Paper Awards, including at ICDCS 2017, IWQoS 2017, and ICNP 2011. He currently serves as an associate editor for *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Service Computing*, *IEEE Transactions on Mobile Computing*, *IEEE Wireless Communications*, *IEEE Internet of Things Journal*, and as an editor for *SpingerBriefs on Cyber Security Systems and Networks*. Kui is a Fellow of IEEE, a distinguished lecturer of IEEE, a member of ACM, and a past board member of the Internet Privacy Task Force, State of Illinois.