Source-Based Path Selection: The Data Plane Perspective

Taeho Lee[†*], Christos Pappas[†*], Cristina Basescu[†], Jun Han[§], Torsten Hoefler[†], Adrian Perrig[†]

[†]ETH Zürich
{kthlee, pappasch, cba, htor, adrian.perrig}@inf.ethz.ch

[§]Carnegie Mellon University
junhan@cmu.edu

Abstract

In source-based path selection, the sender chooses the path to the destination from a set of available paths and embeds the forwarding information in the packets. Future Internet proposals have employed this scheme to realize the benefits of source routing without the inherent scalability problems of path computation at the source. Furthermore, to address the security concerns of packet-carried forwarding state, these proposals leverage cryptographic primitives (e.g., Message Authentication Codes) per packet in the data plane. However, the implications on the forwarding performance of these novel routing schemes have not been studied in detail.

In this paper, we study the data plane of source-based path selection schemes. We take SCION—a future Internet proposal—as an example and sketch its data plane implementation. We implement a software switch that can forward up to 140 million minimum-sized packets per second (limited by the hardware I/O subsystem) and can achieve line-rate throughput of 120 Gbps.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: Network performance; C.2.1 [**Network Architecture and Design**]: Circuit-switching networks

Keywords

Source-based Routing; Data Plane; Future Internet

1. INTRODUCTION

Source routing is a routing technique in which the source specifies a complete or a partial path to the destination [1]. The chosen path is encoded in the packet headers, unlike other routing techniques where distributed forwarding decisions are made at each forwarding device [2].

Source routing offers several advantages [3]: 1) The data plane becomes simpler. Intermediate forwarding devices (e.g., switches and routers) perform very simple operations. 2) Traffic engineering is more flexible, allowing application-optimized path selection at the source. 3) Routing stability is improved (e.g., no transient loops) since the path computation is centralized at the source.

Despite the advantages, source routing has not been deployed in the Internet mostly due to scalability and security reasons: 1) Creating a topology map of the network that the source can use to compute paths becomes a challenge. More specifically, a scalable representation of the entire Internet and the granularity of detail for the topology are challenging problems. 2) An on-path adversary can manipulate the path or redirect traffic. This allows the adversary to perform man-in-the-middle attacks or to infiltrate a secure network.

In recent years, future Internet architecture proposals have revisited the idea of source routing for its benefits while addressing its scalability and security problems. In particular, the scalability problems have been addressed by using source-based path selection, a system in which the source obtains a list of paths that it can use to reach the destination, rather than a full topology map as in generic source routing schemes. In addition, to address the security problems of source routing, cryptographic primitives are used to protect the integrity of the forwarding information in each data packet. Although new architecture proposals [4, 5, 6] realize the benefits of source routing without the disadvantages, the effect of source-based path selection on the data plane has generally been neglected.

Future Internet architectures, such as SCION [4] and Nebula [5], use cryptographic operations to prevent forgery of path information. In this paper, we focus on the impact of cryptographic operations on packet forwarding for such routing architectures. We take SCION [4] as an example and design an efficient data plane. We design and implement a software switch that can forward up to 140M minimum-sized packets per second (limited by the hardware I/O subsystem) and achieves line-rate throughput of 120 Gbps.

2. BACKGROUND

We use SCION [4] as a use case of an architecture that leverages source-based path selection and embeds forwarding information in packet headers. In this section, we describe the basic information that is needed to understand packet forwarding in SCION.

SCION is an Internet architecture that provides route control, failure isolation, and explicit trust relationships for end-to-end communication. To achieve end-to-end communication, initially the ASes—organized in a hierarchical topological structure—explore the topology and create *half-paths*. The half-paths are joined to form end-to-end AS-level paths, which are used in the data plane to forward packets. In the following we briefly summarize the basic building blocks to achieve communication.

Topological Structure. The Autonomous Systems (ASes) are organized in a hierarchical, tree-based structure according to the

[*]These authors contributed equally to this work

CFI '15, June 08 - 10, 2015, Seoul, Republic of Korea

DOI: http://dx.doi.org/10.1145/2775088.2775090

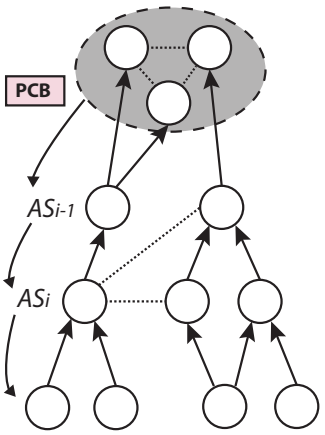

Figure 1: Hierarchical topology of ASes, with tier-1 ISPs in the core (gray circle). The arrows depict customer-provider relationships and the dashed lines depict peering relationships.

provider-customer business relationships between them (Figure 1). The root of the tree, called the *core*, consists of tier-1 ISPs. Parent-child relationships in the tree are determined by the provider-customer relationships, with the provider being the parent node.

Half-Path Construction. Each AS has to discover a set of half-paths, specifically *up-paths* and *down-paths*. The up-paths are a set of paths that each AS_i can use to reach the core. The down-paths are a set of paths that can be used to reach AS_i from the core. Next, we describe how half-paths are constructed.

The ASes in the core periodically initiate *path construction beacons* (PCBs) that traverse the hierarchical topology towards the leaves. Each AS appends information to the PCB to mark its presence on the path and forwards it to each customer. The total information that is appended by the ASes constructs an AS-level path from the core towards the leaves. This information includes the following fields:

1. Interface Field (IF): describes the constructed path at the granularity of interfaces that connect the ASes. Each AS inscribes the ingress and egress interfaces of the border routers that connect the AS with the previous and next AS hops correspondingly:
$$IF(i) = ingress(i)||egress(i)$$
This means that an AS keeps a numbered list of interfaces that connect it with other ASes, so that ingress and egress points can be identified. An AS has full control over the numbering of its interfaces—the numbering is independent for each AS.

2. Opaque Field (OF): a cryptographic marking that is used for data plane forwarding. The term "opaque" refers to the fact that the information in an OF only needs to be readable by the AS; other ASes do not access this information.
$$OF(i) = IF(i)||MAC_{K_i}(IF(i)||OF(i-1)) \quad (1)$$
The MAC is computed using a secret key K_i known only by the corresponding AS_i. A collection of OFs forms the forwarding information in each data packet.

Each AS that receives PCBs learns half-paths that connect it to the core. It can select some of these paths to be used as up-paths and some as down-paths; up and down-paths do not need to be disjoint.

Half-Path Joining. In SCION, an end-to-end path between two ASes is constructed by joining two half-paths. Specifically, the source AS combines one of its up-paths with one of the down-paths of the destination AS. Since every AS can reach the root of the tree and can be reached from the root of the tree, an end-to-end path between any two ASes can be created by combining the half-paths of the two ASes.

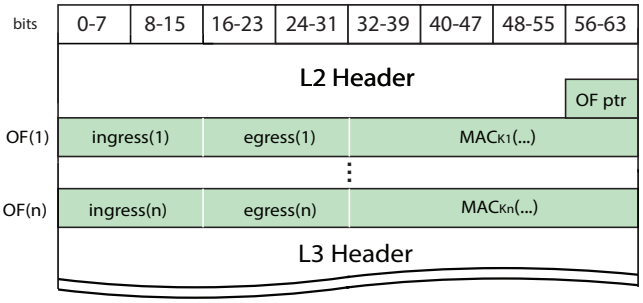

Figure 2: Header for SCION data packet

Packet Forwarding. Each data packet contains a list of OFs that specify the end-to-end AS-level paths at the granularity of border router interfaces. The border routers look at the appropriate OFs to forward the packet to the destination. In the next two sections, we describe the data plane design and provide the implementation details of a switch[1] that forwards packets based on a list of OFs.

3. DATA PLANE DESIGN

We describe the data plane design for highly efficient packet forwarding. The forwarding operations determine the required forwarding information in the packet headers; hence, they specify the packet structure. We describe the packet structure and the necessary operations for packet forwarding.

3.1 Packet Structure

The forwarding state is embedded in each data packet and specifies an AS-level path at the granularity of ingress/egress interfaces (Figure 2). As Section 2 describes, the forwarding state consists of a list of OFs. We describe in more detail each field that is part of an OF.

Ingress Field: specifies the interface at which the packet should arrive when it enters an AS. The length of this field is 16 bits, which suffices to enumerate over 65,000 interfaces.

Egress Field: specifies the interface at which the packet should be sent out towards the next AS. The length of this field is 16 bits.

MAC Field: protects the forwarding information of each AS against alteration and is computed according to Equation 1. To optimize the packet size, the length of the MAC in the OF is reduced to 32 bits. We argue that a 32-bit MAC is sufficient to detect an adversary with high probability if it forges an OF.

A 32-bit MAC is secure under a chosen plaintext attack (IND-CPA secure) [7]. This security property is defined as follows: an adversary is given a set of inputs and their MACs. Then the adversary is given two new inputs (which are not in the set), and only one MAC, which corresponds to one of the two inputs. If the adversary is not able to tell with probability better than half which of the two inputs the MAC corresponds to then the MAC is IND-CPA secure.

In our scheme, even if an adversary eavesdrops on data packets to extract path information (i.e., learning the inputs and the corresponding MACs from the path information), the adversary cannot

[1]We use the terms switch and router interchangeably since we are only concerned with the data plane (i.e., packet forwarding efficiency)

learn to distinguish or to compute MACs for new inputs. Hence, to forge a MAC, an adversary can at best attempt an exhaustive search, which has a success probability of $\frac{1}{2^{32}}$ for each OF field (one-hop path segment). However, to forge a path segment of length n, the adversary cannot guess each MAC field independently, because the adversary does not receive feedback on the validity of each generated MAC. Consequently, forging a path segment of length n requires the adversary to forge all MACs, leading to a success probability of $\frac{1}{2^{32n}}$.

OF Pointer (OF ptr) Field: specifies the OF that the border router should check, which corresponds to the OF that the AS has generated during the PCB propagation. The OF ptr is incremented when a border router sends the packet to the next hop AS; the SCION specification does not require integrity protection for the OF pointer [4]. The length of this field is 8 bits, allowing AS paths up to 256 hops; the average AS path length today is 3.9 hops for IPv4 and 3.6 hops for IPv6 [8].

3.2 Packet Forwarding

Embedding forwarding state in the packet simplifies the forwarding operations at each switch, by eliminating inter-domain forwarding tables. Namely, a switch has to check the interfaces specified in an OF, verify a MAC, and update the OF pointer. We describe the operations of an ingress and egress switch in more detail.

Ingress Switch: An ingress switch has to verify whether the packet enters the AS at the correct ingress point. Specifically:

1. The switch checks whether the incoming port for the packet corresponds to the ingress field in the OF. If the check passes it proceeds with Step 2.
2. The switch verifies the MAC in the OF, using its local secret key K_i, which was used to generate the MAC in the control plane. If the MAC is successfully verified, the switch forwards the packet towards the egress interface.

If either check fails, the packet is dropped. Note that skipping the ingress port check would enable attacks where the packet arrives at an unauthorized ingress port. Such attacks are feasible under a threat model with colluding ASes [4].

SCION does not specify intra-AS routing; hence, each AS is responsible to route the traffic from the ingress point to the egress point.

Egress Switch: An egress switch has to forward the packet to the next AS at the interconnection point that was specified by the OF. The required operations are the following:

1. The switch verifies the MAC in the OF, in the same way that the ingress switch does.
2. It updates the OF pointer to point to the next OF.
3. It forwards the packet through the egress interface.

For both switches, the computational requirements are equivalent and require symmetric-key cryptographic operations. In the next section we explain how we implement these operations in a software switch.

4. SWITCH IMPLEMENTATION

The SCION architecture requires cryptographic operations that can potentially harm the high forwarding performance that is crucial in the data plane. We describe how we combine a state-of-the-art packet I/O engine, a hardware cryptographic engine, and optimizations that achieve line-rate forwarding performance.

Packet I/O. We implement the software switch using Data Plane Development Kit (DPDK) [9], a recent software packet processing platform. The advantages of DPDK are three-fold: 1) It performs packet processing in userspace, which provides flexibility to implement SCION forwarding operations as a userspace application. 2) It adopts a pure polling strategy to avoid unnecessary interrupts and context switches. 3) It modifies the NIC drivers to map packet buffers in userspace such that only a single copy is needed from NIC to main memory. The NIC performs a Direct Memory Access (DMA) via the PCIe bus. This process avoids the overhead of an additional packet copy from kernel to userspace.

Hardware Cryptographic Engines. Each switch needs to perform a MAC verification. To construct the MACs, we use Cipher Block Chaining mode (CBC-MAC) with AES as the underlying block cipher. The value for the Initialization Vector (IV) is 0 and the size of the input and output blocks is 128 bits (16 bytes).

In SCION, the length of the input message to the AES operation is 96 bits (32 bits for ingress and egress interfaces, and 64 bits for the previous OF). The constant 96-bit inputs feature two advantages: 1) The input fits into one AES block. 2) It offers resilience against variable-length input message attacks on CBC-MAC [10].

For computing and verifying the CBC-MACs, we use Intel AES-NI [11], an instruction set that uses hardware support to speed up AES operations. Intel reports that AES encryption of a single 16-byte block consumes 2.01 cycles per byte (CPB) when executed on Intel Westmere running at 2.67 GHz [11].

Performance Optimizations. To achieve high forwarding performance, we implement the following optimizations:

- *Multi-core Parallelization:* First, for each NIC port, we assign a dedicated CPU core that handles all packet I/O operations. Second, we take advantage of the dedicated AES-NI engine that exists on each physical core. Hence, packets received on different ports are processed by the AES-NI engines of the physical core assigned to each port.

- *Elimination of Concurrent Access to Same Memory Area:* To realize the processing speed-up of multi-core parallelization, we create per-core data structures for read and write accesses; this approach eliminates unnecessary data cache misses. Specifically, each NIC is linked with a receive queue and a transmit queue; these queues are then assigned to a CPU core to handle the NIC's traffic. Depending on the system's hardware, we load balance traffic from one NIC over multiple cores. To this end, we assign multiple queues per NIC and each queue can be handled by another core. To distribute the traffic among the queues of a NIC, we use Receive Side Scaling (RSS) [12].

4.1 Packet Life Cycle

To provide more insight to the operation of the switch, we describe the life cycle of the packet from the time it enters the switch until it exits the switch.

First, when the NIC receives packets from the network, the NIC copies the packets to the userspace buffers and updates the receive descriptor ring. This process is done by performing a Direct Memory Access (DMA) from the NIC to the host memory, leveraging the PCIe bus. Second, through the polling mechanism, a thread running in the CPU core detects the reception of packets and loads the packets into the cache from the memory. Then it reads from the cache the OF pointer to identify the OF corresponding to the current AS, and extracts the corresponding OF and the previous

OF to perform the MAC verification. Third, the AES-NI instruction set is used to perform the MAC computation over the ingress and egress interfaces of the current OF and the previous OF. The OF verification succeeds if the resulting MAC value equals to the MAC value in the current OF. The packet is dropped if the MAC is incorrect. When the OF verification is successful, the thread updates the transmit descriptor ring for the output port and notifies the NIC of a pending transmission.

5. EVALUATION

We compare the SCION switching performance with traditional IP forwarding and demonstrate the efficiency of the switch, despite the cryptographic operations.

5.1 Experimental Setup

To evaluate IP lookup, we create a forwarding table based on a recent RouteViews snapshot [13] with 500k unique entries, adding up to 2.5 MB for our Forwarding Information Base (FIB). The IP lookup implementation is based on DIR-24-8-BASIC [14], which is a popular software implementation for the Longest Prefix Match algorithm; it requires only one memory lookup in most cases.

We implement the SCION switch on a commodity server machine, consisting of two 8-core Intel Xeon E5-2680 2.7 GHz CPUs (20 MB L3 cache per CPU), equipped with 6 dual-port Intel 82599EB X520-DA2 10GbE NICs (PCIe Gen2x8), providing a total capacity of 120 Gbps. Note that the cache size can store the whole FIB and hence, the traffic pattern does not influence the forwarding performance (no cache misses). Furthermore, we utilize a Spirent SPT-N4U-220 traffic generator [15] for generating load on the switch. We connect the packet generator back-to-back with the switch and generate traffic. The switch receives the generated traffic, processes it, and sends it back to the generator.

5.2 Evaluation Results

We conduct two experiments, comparing SCION and IP forwarding. At first, we enable only one port for forwarding and demonstrate that the switching performance saturates line-rate. Next, we enable all ports and show how performance scales. The processing load for SCION switches is independent of the path length: each switch performs the same operations, and the SCION header fits in a single cache line.

For a given link capacity, the packet size determines the packet rate and hence the load on the switch. For example, Ethernet frames have a minimum size of 84 Bytes (64 bytes data + 7 bytes preamble + 1 byte frame delimiter + 12 bytes interframe gap). Thus, the maximum rate of 10 Gigabit Ethernet is 14.88 Million packets per second (Mpps) per NIC port and 178.56 Mpps for all 12 ports. For 1518-byte packets, the packet rate for 1 port is 0.81 Mpps and for 12 ports 101.4 Mpps. These values are the physical limits and represent the theoretical peak throughput. We use the packet I/O forwarding performance without any processing (i.e., no state processing and no memory lookup) as the performance baseline (Packet I/O Baseline).

5.2.1 *Single-port Experiment*

We analyze the switching performance for various packet sizes, ranging from 64 to 1518 bytes. In Figure 3, for clarity, we only present two cases, 64 and 1518 byte-packets, which correspond to the worst and best cases, respectively; however, the results are the same for the other packet sizes.

When the traffic saturates the bandwidth of the link, the switch receives packets at the rate of 14.88 and 0.81 Mpps (Million of packets per second) for 64 and 1518-byte packets, respectively;

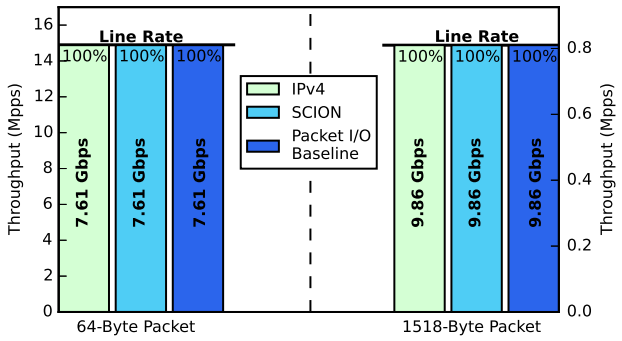

Figure 3: Switching performance of 64- and 1518-byte packets on a single port; different scales (i.e., y-axis values) are used for the two packet sizes.

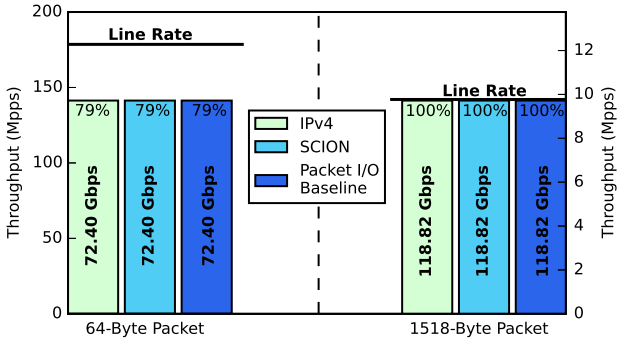

Figure 4: Switching performance of 64- and 1518-byte packets on all ports; different scales (i.e., y-axis values) are used for the two packet sizes.

these are the line-rate performances. For the baseline case, the switching performance is the same as the line-rate performance. In addition, the figure shows that SCION forwarding performance is on par with both IPv4 and baseline performances for both packet sizes, which means that the optimal switching performance is achieved for the single-port experiment.

5.2.2 *All-ports Experiment*

In this experiment, we analyze the forwarding performance of our switch when the traffic generator generates packets at its full capacity, using all 12 ports. For the two packet sizes, the line-rate performance increases to 178.56 Mpps and 9.71 Mpps, respectively.

Figure 4 shows that the switching performance of SCION is identical to that of both IPv4 and baseline cases. However, for 64-byte packets, we see that the performance is at 79% of the line rate. The reason for this is that the I/O subsystem of the server machine hits a bottleneck when both ports of a NIC receive packets at the maximum rate (each NIC has two 10 Gigabit ports). This bottleneck is system-dependent and not related to SCION processing: the PCIe Gen2 x8 interface of the NICs cannot handle this packet rate for two active ports and each port is capped at 11.55 Mpps. For the same NICs, Zhou et. al., [16] report the same bottleneck.

6. DISCUSSION

The advantages of carrying forwarding state in packets come at the cost of increased packet size, i.e., packet header increases linearly with the number of AS hops on the path. Specifically, each AS introduces 64 bits (8 bytes) of additional information. To put this overhead into context, we analyze three hour-long packet traces, obtained from CAIDA [17]. Based on the number of packets and the packet sizes of the trace, we calculate the bandwidth overhead,

	# Packets	Size (GB)	BW Overhead (%)
Trace 1	1012841645	704.53	4.42
Trace 2	1878632515	1609.67	3.59
Trace 3	2256208197	1546.03	4.49

Table 1: Bandwidth overhead of SCION according to three backbone link traces.

assuming an AS path of 4 hops (the average AS-path length today is less than 4 hops [8], as mentioned). This analysis assumes that there is no correlation between the distribution of AS path lengths and packet sizes. Table 1 shows the bandwidth overhead. We see that the overall bandwidth overhead is low and does not exceed 4.5%.

7. RELATED WORK

To our knowledge, this is the first attempt to design and implement a highly efficient data plane of a source-based path selection architecture that leverages cryptographic primitives. Although our work is not comparable with other software switching proposals for today's networks (e.g., IP or SDN) [16, 18, 19], our design and implementation can prove beneficial to proposals, such as ICING [20] and OPT [21], that leverage symmetric-key cryptography on the data plane.

ICING [20] is a research proposal for path verification and enforcement that relies on cryptographically protected packet-carried state. More specifically, each router on the path derives a shared key with every other router. For each packet, a router inserts a MAC for every other router on the path, hence, each router has to verify a number of MACs depending on the path length.

OPT [21] proposes light-weight protocols for source authentication and path validation. OPT also leverages symmetric-key cryptography at each router to generate a local secret key per packet and perform MAC computations. Specifically, to perform source authentication a MAC computation is required, and to perform path validation a chain of nested MACs is constructed.

8. CONCLUSION

Source-based path selection has been known to simplify data plane design. However, many future Internet architectures have used cryptographic operations to protect the path information from forgery, and the impact of cryptographic operations on data plane has not been studied previously. In this paper, taking SCION as the example architecture, we have built a software switch that is capable of forwarding packets with cryptographically protected path information. We have demonstrated that a per-packet MAC verification does not affect the forwarding performance of a switch. Our results confirms that source-based path selection allows a simple and efficient data plane design, despite the additional cryptographic operations.

9. ACKNOWLEDGEMENTS

We thank Yue-Hsun Lin and the anonymous reviewers for their insightful feedback and suggestions. The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement 617605. We also gratefully acknowledge support by ETH Zurich, and by Intel for their equipment donation that enabled the high-capacity experiments.

10. REFERENCES

[1] C. A. Sunshine, "Source Routing in Computer Networks," *SIGCOMM Comput. Commun. Rev.*, Jan. 1977.

[2] B. M. Hauzeur, "A Model for Naming, Addressing and Routing," *ACM Trans. Inf. Syst.*, Dec. 1986.

[3] J. H. Saltzer, D. P. Reed, and D. D. Clark, "Source Routing for Campus-wide Internet Transport," Mar. 1980.

[4] X. Zhang, H.-C. Hsiao, G. Hasker, H. Chan, A. Perrig, and D. G. Andersen, "SCION: Scalability, Control, and Isolation on Next-Generation Networks," in *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2011.

[5] T. Anderson, K. Birman, R. Broberg, M. Caesar, D. Comer, C. Cotton, M. Freedman, A. Haeberlen, Z. Ives, A. Krishnamurthy, W. Lehr, B. Loo, D. Mazières, A. Nicolosi, J. Smith, I. Stoica, R. van Renesse, M. Walfish, H. Weatherspoon, and C. Yoo, "The NEBULA Future Internet Architecture," in *The Future Internet*, 2013.

[6] X. Yang, D. Clark, and A. W. Berger, "NIRA: A New Inter-domain Routing Architecture," *IEEE/ACM Trans. Netw.*, Aug. 2007.

[7] S. Goldwasser and S. Micali, "Probabilistic encryption," *Journal of Computer and System Sciences*, 1984.

[8] RIPE Labs, "Update on AS path lengths over time." "https://labs.ripe.net/Members/mirjam/update-on-as-path-lengths-over-time". [Accessed 05/2015].

[9] "Data Plane Development Kit." "http://dpdk.org". [Accessed 05/2015].

[10] M. Bellare, J. Kilian, and P. Rogaway, "The Security of the Cipher Block Chaining Message Authentication Code," *J. Comput. Syst. Sci.*, Dec. 2000.

[11] S. Gueron, "Intel Advanced Encryption Standard (AES) New Instruction Set." "https://software.intel.com/sites/default/files/article/165683/aes-wp-2012-09-22-v01.pdf", March 2010.

[12] S. Goglin and L. Cornett, "Flexible and extensible receive side scaling," Sept. 1 2009. US Patent 7,584,286.

[13] "University of Oregon RouteViews Project." "http://www.routeviews.org". [Accessed 05/2015].

[14] P. Gupta, S. Lin, and N. McKeown, "Routing lookups in hardware at memory access speeds," in *Proceedings of IEEE INFOCOM*, 1998.

[15] "Spirent SPT-N4U-220 Chassis." "http://www.spirent.com/Ethernet_Testing/Platforms/N4U_Chassis". [Accessed 05/2015].

[16] D. Zhou, B. Fan, H. Lim, M. Kaminsky, and D. G. Andersen, "Scalable, High Performance Ethernet Forwarding with CuckooSwitch," in *Proceedings of ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, 2013.

[17] "CAIDA: Center for Applied Internet Data Analysis." "http://www.caida.org". [Accessed 05/2015].

[18] S. Han, K. Jang, K. Park, and S. Moon, "PacketShader: A GPU-accelerated Software Router," in *Proceedings of ACM Conference on SIGCOMM*, 2010.

[19] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The Click Modular Router," *ACM Trans. Comput. Syst.*, Aug. 2000.

[20] J. Naous, M. Walfish, A. Nicolosi, D. Mazières, M. Miller, and A. Seehra, "Verifying and Enforcing Network Paths with ICING," in *Proceedings of ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, 2011.

[21] T. H.-J. Kim, C. Basescu, L. Jia, S. B. Lee, Y.-C. Hu, and A. Perrig, "Lightweight Source Authentication and Path Validation," in *Proceedings of ACM Conference on SIGCOMM*, 2014.