SIXPACK: Securing Internet eXchange Points Against Curious onlooKers

Technical Report, 21 pages

Marco Chiesa
KTH Royal Institute of Technology
Université catholique de Louvain

Daniel Demmler
Technische Universität Darmstadt

Marco Canini
Université catholique de Louvain

Michael Schapira
Hebrew University of Jerusalem

Thomas Schneider
Technische Universität Darmstadt

ABSTRACT

Internet eXchange Points (IXPs) play an ever-growing role in Internet inter-connection. To facilitate the exchange of routes amongst their members, IXPs provide Route Server (RS) services to dispatch the routes according to each member's peering policies. Nowadays, to make use of RSes, these policies must be disclosed to the IXP. This poses fundamental questions regarding the privacy guarantees of route-computation on confidential business information. Indeed, as evidenced by interaction with IXP administrators and a survey of network operators, this state of affairs raises privacy concerns among network administrators and even deters some networks from subscribing to RS services. We design SIXPACK[1], an RS service that leverages Secure Multi-Party Computation (SMPC) to keep peering policies confidential, while extending, the functionalities of today's RSes. As SMPC is notoriously heavy in terms of communication and computation, our design and implementation of SIXPACK aims at moving computation outside of the SMPC without compromising the privacy guarantees. We assess the effectiveness and scalability of our system by evaluating a prototype implementation using traces of data from one of the largest IXPs in the world. Our evaluation results indicate that SIXPACK can scale to support privacy-preserving route-computation, even at IXPs with many hundreds of member networks.

CCS CONCEPTS

• **Networks** → **Network privacy and anonymity**; Network control algorithms; • **Security and privacy** → *Privacy-preserving protocols*;

KEYWORDS

Internet eXchange Points, privacy-preserving routing, interdomain routing, Secure Multi Party Computation.

1 INTRODUCTION

With the rise of Internet eXchange Points (IXPs) as the emerging physical convergence points for Internet traffic, new privacy concerns arise. IXPs offer centralized Route Server (RS) services for ranking, selecting, and dispatching BGP routes to their (potentially many hundreds of) member networks [76]. However, to benefit from these centralized services, IXP members need to divulge *private* information, such as peering relationships and route-export policies to the IXP or, even worse, to other IXP members. Such information can reflect sensitive commercial and operational information, and is consequently often regarded as private [44, 92]. Indeed, our interaction with IXP administrators, and our survey of network operators (§2), reveal that such privacy concerns are widespread and that some networks even refrain from subscribing to RS services for precisely this reason. Beyond privacy, our survey reveals three additional IXP members' concerns for RS usage: limited routing policy expressiveness, reliability, and insufficient value. This situation hinders RS adoption and makes it hard to provide novel valuable routing services to IXP members, as these can rely on the exposure of (even more) sensitive data. Indeed, on the one hand, advanced performance-oriented routing services are fundamental to improve performance of video and latency-critical Internet applications [19, 23, 24, 52, 58, 74, 75, 86, 88, 90]. In this regard, today's largest content providers (e.g., Google and Twitch) resort to active measurement techniques for inferring route performance [40], a difficult task in practice [24]. On the other hand, our survey reveals that a large majority of network operators (60%) is concerned about sharing network performance information such as IXP port utilization with external entities. In this regard, the goal of supporting advanced Internet routing features while protecting sensitive information is the subject of several recent studies [8, 44, 45, 56, 57, 62, 69, 93].

How should we design RSes? To increase trust in IXPs and motivate further adoption of RS services, we argue that RSes should meet the following basic requirements: *a) Easy management*, i.e., relieve IXP members from the burden of configuring numerous BGP peering sessions, *b) Policy expressiveness*, i.e., provide IXP members with highly-expressive route selection at least equivalent to having multiple bilateral BGP sessions [36], *c) Performance-driven routing*, i.e., dispatching tools that leverage the IXP's superior visibility into data-plane network conditions, *d) Efficiency*, i.e., today's RSes are required to compute and dispatch routes in the order of hundreds per second, with full routing-table transfers performed in the order or minutes [2, 27, 79], *e) Privacy preservation*, i.e., no IXP member nor the IXP itself should be able to learn information about routing policies of the other members (except for information that can be deduced from its own RS-assigned routes), *f) Reliability*, i.e., guaranteed connectivity upon failures in the IXP infrastructure.

[1]Project page available at https://six-pack.bitbucket.io. Code available at https://bitbucket.org/six-pack/six-pack.

SIXPACK: A privacy-preserving advanced RS. To accomplish the above, we advocate implementing an RS via secure multi-party computation (SMPC) [38, 91]. With an SMPC-based realization of an RS, the desired routing outcome can be computed without the IXP or IXP members gaining visibility into the "inputs" to this computation, i.e., members' private routing policies.

However, realizing our vision of a privacy-preserving RS is highly nontrivial. General-purpose SMPC machinery is excessively heavy in terms of computation and communication overheads and is thus infeasible to employ for this purpose [44]. Consequently, attaining feasible runtimes and communication costs requires devising a suitable highly-optimized, privacy-preserving scheme specifically tailored to the RS context. We present SIXPACK, the first IXP route server service that satisfies *all* the aforementioned requirements.[2] It efficiently ranks, selects, and dispatches BGP routes based on the members' expressive routing policies and any IXP-provided performance information *without* leaking any confidential business peering information.

A conceptual overview of SIXPACK is given in Fig. 1. The IXP route server service is jointly performed by two independent and non-colluding computational parties, RS_1 and RS_2, which run an SMPC protocol. Each IXP member encrypts its BGP routes, announces them to the RSes, and creates two "shares" of its (private) business peering policy that are sent to the two RSes. Each of the RSes, in turn, sends to each IXP member, upon completion of the SMPC, a share of its output, that the member can use to recover its selected routes. SIXPACK provably guarantees that as long as RS_1 and RS_2 do not collude with each other, neither the RSes nor other IXP members will learn any information regarding an IXP member's business and routing policies. We envision RS_1 and RS_2, as being run by the IXP and a neutral and well-regarded international organization (e.g., NANOG or RIPE), which is already trusted to support and operate fundamental Internet services (though not necessarily trusted for privacy confidentiality)[3]. In addition, one of the two RSes should be executed on a machine that is located outside the IXP but in very close proximity (i.e., within the same colocation data center) so as to be in a separate security domain while keeping latency of inter-RS communication at a minimum (i.e., $<$ 1ms).

We observe (§5) that a naïve application of SMPC to RS computation, in which the entire RS computation is carried out via SMPC machinery, is infeasible in practice as a result of unrealistic computation and communication overheads in SMPC. In fact, network operators rely on a highly-expressive inter-domain routing protocol to export, rank, and filter routes routes based on their own routing policies. In the Border Gateway Protocol (BGP), the standard de-facto inter-domain routing protocol, such operations often entail evaluating regular expressions based on the traversed networks, a computationally prohibitive operation in SMPC [55, §5]. To preserve the same routing expressiveness of BGP, we thus resort to the following approach. SIXPACK is carefully designed to keep complex computation outside the SMPC, to the largest extent possible without compromising privacy. Specifically, in SIXPACK, we carefully decompose the RS functionality into two simple, yet crucial, SMPC-based building blocks for *efficient* route-dispatch, called EXPORT-ALL and SELECT-BEST, while performing the most complex computation over unencrypted data outside of the SMPC without leaking any private information. Namely, all members that wish to announce a route through the IXP will first locally compute the set of members to whom each route should be exported. using any arbitrary complex "export" routing policy. This computation is performed outside of the SMPC. Using EXPORT-ALL, all available BGP routes that are exportable to an IXP member, that is, the routes that other IXP members are willing to advertise to that member, are dispatched to the member in a fully privacy-preserving manner. Then, the member locally ranks its available routes outside of the SMPC according to its arbitrary complex local preferences over routes and feeds the resulting ranking as input into SELECT-BEST. SELECT-BEST leverages this information and information from the IXP (e.g., port utilization) to select the best route for that member without leaking any information. Performing the ranking routes outside of the SMPC greatly enhances the performance of the system. In fact, to achieve the same routing expressiveness of today's interdomain routing protocols, ranking routes must support complex regular expression evaluations and requires, we move [4] Thus, SIXPACK both goes well beyond the services offered by today's RSes (by delegating route-selection from the RS to the member and incorporating performance-related information into the route selection process) and provides strong privacy guarantees to IXP members.

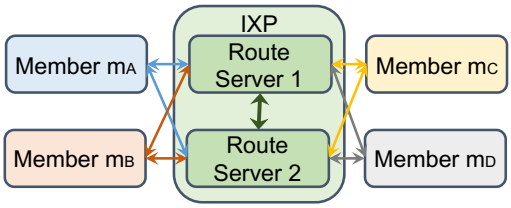


Figure 1: Conceptual overview of SIXPACK. The four IXP members communicate with the two route server entities, which run SMPC to perform ranking, selection, and dispatch of BGP routes.

We discuss SIXPACK's optimized design, underlying assumptions, threat model, deployment challenges, IXP visibility of data-plane traffic, etc., in detail in the following sections.

Paper contributions.

- An analysis of the operators' concerns about peering with RSes at IXPs through a survey with 119 responses and a measurement of RS usage at one of the largest IXP worldwide.
- The design and implementation of the first IXP route server capable of keeping the peering policies and routing preferences private, while allowing the IXP members to express arbitrary BGP routing policies *including* policies that incorporate confidential performance-related information available at the IXP (e.g., port utilization).[5]

[2] While the focus in this work is not reliability, even today, at large IXPs, members are contractually requested to set multiple peering sessions with distinct RSes for redundancy requirements [28]. Also, recently proposed Internet drafts [50] further mitigate the impact of failures in the IXP network.

[3] In addition, our survey of network operators [66] reveals RIRs enjoy the trust of a large fraction (88%) of respondents.

[4] Thanks to our modular design, a member may skip the SELECT-BEST phase at the cost of revealing to the IXP that it disregards using IXP information.

[5] We plan to release our SIXPACK prototype as open source.

- Applying an SMPC approach to the important and timely context of route dispatch at IXPs and devising *efficient and privacy-preserving* SMPC building blocks for RSes.
- An evaluation of our prototype. Through experiments with a BGP trace from one of the largest IXPs in the world, our results show that SIXPACK scales to hundreds of IXP members and achieves BGP processing times below 90ms at the 99^{th} percentile. Via microbenchmarks, we assess the online costs of SMPC-based RSes to be well within *real-time* processing requirements of large IXPs.

2 ON IXPS AND PRIVACY

We present below preliminaries on Internet exchange points and quantify, via measurements, the extent to which route server services are used. We also report on our interaction with IXPs and member network administrators, including a survey of network operators, indicating that privacy concerns are a significant factor hindering widespread RS usage and corroborating assumptions underlying past research regarding the privacy of routing policies.

Background on IXPs. IXPs are high-bandwidth physical networks located within a single metropolitan area. IXPs are typically geographically distributed [5, 25] and hosted within colocation centers [51], facilities operated by *third party providers* that offer high levels of physical security.

Heterogeneous economic entities use IXPs to exchange Internet traffic with each other [1]. To do so, each *member* connects its own network to one or more physical ports at the IXP network. After physical connectivity is established, each member announces the set of IP prefix destinations for which it is willing to receive traffic and starts receiving route announcements from the other members of the IXP.

The routes used to reach prefixes are spread and selected via the de facto standard inter-domain routing protocol of the Internet, i.e., Border Gateway Protocol (BGP). To this end, a full-mesh of BGP sessions among each pair of IXP members may be established. At medium to large IXPs, which can have over 800 members and carry over 5 Tbps, such full-meshes can be partially replaced by a *Route Server* (RS) service to ease the exchange of BGP announcements among members [76]. The RS establishes a BGP session with each IXP member, collects and distributes their BGP route-announcements. Note that data-plane traffic does not traverse the RS, which is only involved in control-plane traffic.

Each IXP member has the freedom to specify, for each destination IP prefix, an *export policy*, i.e., the set of other IXP members that are allowed to receive its route announcements. The RS selects, for each member, a route (per IP prefix) that is "exportable" to that member (according to members' export policies), and dispatches it to that member.

Today's IXPs do not allow their members to influence the RS's route selection with the members' *import policies*. These policies comprise of the traditional BGP *local preferences* [20] and regular expressions that the RS uses to rank and filter available routes [36]. For instance, an operator may be interested in routing its traffic through a certain IXP member unless the announced route traverses a specific network.

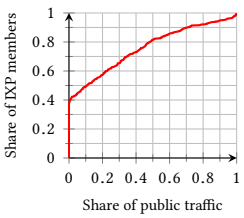


Figure 2: CDF of RS usage at a large IXP.

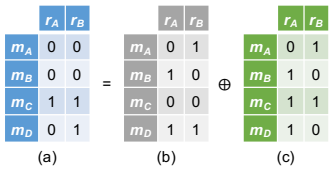


Figure 3: (a) Export policy matrix in plain text. (b) Random shares received by RS1. (c) Element-wise XOR of (a) and (b) received by RS2.

How widespread is RS usage? Despite the fact that such an RS service eases the management of BGP sessions, facilitates peering, and lowers hardware requirements on connected BGP routers, there is anecdotal evidence of its limited usage. To corroborate this, we performed an analysis of RS usage at one of the largest IXPs worldwide. We have been reported that similar values hold for at least another of the largest IXPs worldwide. We examined both data traffic and BGP control plane messages so as to quantify the fraction of traffic that is routed along the routes dispatched by the RS. Fig. 2 presents a CDF graph showing the fraction of IXP members that have less than a certain fraction of "public traffic", i.e., traffic routed along the RS-computed routes. As shown in the leftmost part of the figure, 40% of members do not route their traffic via RS-prescribed routes. About two-thirds of the remaining members route less than half of their traffic according to the RS and only 20% of members route over 50% of their traffic along RS-computed routes. In terms of absolute amount of traffic, we discovered that less than 17% of the overall traffic is routed via RS-prescribed routes. While Ager et al. [1] observed that most of the networks peer with the RS, we showed that members prefer to route the vast majority of their traffic based on the information exchanged through bilateral BGP sessions.

Are privacy concerns hindering wider RS usage and innovation? One of the main barriers facing the transition from a full-mesh of BGP peering sessions to a star topology (via an RS) is that the export policy of each member (and, to support route-selection at the RS, potentially also the import policy of each member) must be revealed to the IXP. This information is considered confidential, primarily due to commercial reasons. Indeed, our interaction with IXP administrators and network operators reveals such privacy concerns and, moreover, that some networks do not connect to RSes for precisely this reason. In particular, we circulated a survey among the network operator community [66] with the aim of exploring their perceptions about privacy at IXPs. We collected 119 responses belonging to a broad range of different networks: Tier 1 ISPs (8%), Tier 2/3 ISPs (57%) CDNs (6%), content providers (12%), and others (17%), with almost all networks connecting to an IXP and 80% of them using RS services (at least for that fraction of traffic not belonging to an established bilateral peering). According to our survey, the most critical concerns regarding RSes among respondents were: no control over best route selection, i.e., lack of import policy configuration tools (53%), reliability (40%), lack of route visibility (37%), privacy (19%), and legal restrictions (7%). Since the 1st and 3rd item require members to disclose their policies to the RS, we further investigated this privacy aspect: (i) 40-45%

of respondents consider their local preferences over BGP routes to be private, both with respect to the IXP and with respect to other members, (ii) 60% of respondents expressed concerns about sharing their IXP port utilization with other members, and (iii) 1 of every 4 respondents that do not use an RS service marked concerns about disclosing export policies to the IXP as a reason. With comments ranging from "*Nothing should be considered private*" to '"*Everything listed is supposed to be private/proprietary information*", our survey revealed the heterogeneous requirements of Internet domains. Despite the existence of many networks with open peering policies, our survey reveals that local-preferences over routes and port utilizations are still considered as a private information not to be divulged.

We point out that beyond privacy concerns, revealing sensitive information also entails the risk of triggering attacks on the weaker parts of the network [68], e.g., easier bandwidth-exhaustion attacks if port utilization is revealed [89].

3 THREAT MODEL AND ASSUMPTIONS

To provide strong privacy guarantees, SIXPACK relies on three assumptions (A), which we list and justify next.

Assumption 1: Two non-colluding RSes. We assume that the RS service consists of two distinct route servers: one is operated by the IXP and one by an independent, non-colluding entity. The latter RS is executed on a machine that is outside of the IXP domain but connects to the IXP network at the co-location center where the IXP is hosted. Since this instance lives in a separate security domain, this solution minimizes the possibility of an RS instance being compromised by the IXP, while keeping latency at a minimum [4].

We believe that neutral international organizations (e.g., RIPE), which are already trusted to *support* and *operate* fundamental Internet services such as DNS and IP allocation, should be assigned the task of running an instance of an RS or, alternatively, supervising those that do. We argue that running an RS instance is a simpler task than operating a distributed DNS system. Our survey of network operators [66] reveals that RIRs enjoy the trust of an overwhelming fraction (88%) of respondents. We point out that, even at today's large IXPs, members are requested to set up multiple peering sessions with distinct RSes for redundancy requirements [28]. In SMPC, redundancy is required for both RS instances.

Assumption 2: Honest-but-curious RSes. SIXPACK protects against so-called "honest-but-curious attackers", i.e., RSes that stick to the protocol but try to infer the members' private inputs. We argue that as today some networks refrain from peering with others via route servers because of fear of revealing private information to the RSes, this model captures an important desideratum in the IXP ecosystem. Furthermore, if cheating (e.g., deviation from the protocol by an RS) was detected, this would result in massive loss of trust in the service and, consequently, severe economic consequences for the IXP. We point out, however, that our SMPC circuits constructions (§5.4 and §5.5) could be evaluated with a framework secure against malicious adversaries (e.g., [73]) if desired, though at higher computation and communication overheads.

Assumption 3: No visibility into data traffic. SIXPACK is designed to hide control-plane information from the RSes. We view data-plane privacy-preservation, i.e., preventing the IXP from inferring routing policies from observing data traffic traversing the IXP network, as an orthogonal problem that requires further exploration. We refer the reader to §8 for an explanation of why inferring routing policies from the data plane is highly challenging even when information about BGP routes is available. We point out, however, that SIXPACK actually does make the inference of routing policies from data traffic more challenging for the IXP. Guaranteeing data-plane-level privacy can also involve other approaches such as encrypting and decrypting IP headers at IXP ingress and egress. We leave this interesting topic for future research.

4 SECURE MULTIPARTY COMPUTATION

Overview of SMPC. Secure multi-party computation (SMPC) allows multiple parties, to jointly compute the outcome of a function f while keeping their inputs to it and the outputs of f private. SMPC was established as a purely theoretical construct more than 30 years ago and was initially seen as too inefficient to be used in practice due to large communication and computational overhead. However, a recent line of research has improved SMPC primitives drastically and showed that practical implementations of SMPC are possible. See, e.g., [16, 46, 47, 61, 64, 71].

One application of SMPC is outsourcing the computation of a function f from the parties holding the private inputs, called "members" henceforth, to several external computational parties. Members' inputs are distributed to the computational parties while remaining secret, and the computational parties run a computation on the distributed inputs, without obtaining visibility into the actual inputs or outputs.

In this work, we employ the well-established SMPC protocol of Goldreich-Micali-Wigderson (GMW) [38] that operates on a Boolean circuit, consisting of logic gates such as AND and XOR, which represents the function f to be computed. We rely on the proven security of the GMW protocol [37, 38].

While the GMW protocol, in general, can also be executed by more than two parties, we consider the scenario where private inputs from the members are outsourced to exactly two computational parties $SMPC_1$ and $SMPC_2$. This decision is made mainly for performance reasons. The outsourcing of private member inputs is achieved via *XOR-based secret sharing* as follows: Each member selects, for every plaintext input bit b, a random bit b_1, and computes bit $b_2 = b \oplus b_1$. The bits b_1 and b_2, called *shares* of b, are distributed between the SMPC parties such that $SMPC_i$ obtains share b_i. Importantly, from the perspective of the two computational parties, the shares are indistinguishable from random bits.

After the input sharing phase, the circuit representing the function f to be computed is evaluated using shares as inputs to the Boolean gates. XOR gates are evaluated locally, by each computational party computing the XOR of the input shares sent to that party. AND gates, in contrast, require one round of communication between the computational parties. Specifically, AND gate evaluation can be executed via oblivious transfer (OT) [7], or using Beaver's multiplication triples [9], which can be efficiently precomputed. While AND gates on one layer of the circuit can be evaluated in parallel, AND gates on subsequent layers have to be evaluated sequentially due to the data dependency between them.

Hence, from a GMW implementation perspective, the lower the multiplicative depth (in terms of AND gates) of the circuit representing the function of interest f, the better the round complexity which determines the latency.

After the evaluation of all gates in a circuit, the output shares computed by the two parties are sent to the members, who are then able to recover their plaintext output by computing the XOR of the two output shares.

We emphasize that our circuits could be evaluated also with other SMPC protocols (e.g., Yao's garbled circuits [91]) or protocols that provide security against stronger adversaries (e.g., [73]) and/or use more than two computational parties (of which a fraction can be corrupted). However, these protocols have significantly higher communication and/or computation complexities.

5 SIXPACK PRIVACY-PRESERVING RS

We introduce SIXPACK, a privacy-preserving RS service for IXPs. SIXPACK combines the benefits of a centralized route dispatch service with the provable guarantee of privacy preservation. Through SIXPACK, IXP members can receive the best available BGP routes according to arbitrary local route preferences and auxiliary information of the IXP (e.g., knowledge of congestion level and other performance metrics [58]).

Building on recent advances in SMPC, SIXPACK employs two independent and non-colluding computational entities to correctly dispatch route announcements without gaining any visibility into the members' routing policies (i.e., export/import policies) nor leaking any private IXP performance-related information to the members.

To illustrate the non-trivial challenges facing SIXPACK's design, we first discuss a fairly naïve approach to applying SMPC to route-dispatch at IXPs, and why this approach fails. We then describe the key design ideas behind SIXPACK, present the routing policy model, and discuss in detail the two main components of the system.

5.1 A Naïve Approach

As general-purpose SMPC is capable of arbitrary computation, it would be tempting to implement SIXPACK solely within a single SMPC, thus providing arbitrary privacy-preserving policy expressiveness. This task entails devising a function that takes as input the set of members' export policies, the set of arbitrarily complex members' import policies (e.g., regular expressions on the AS networks traversed by a route), and the IXP's performance-related information (e.g., port utilization), and outputs the resulting "best" BGP routes.

However, even performing a single full regular expression string matching operation in SMPC using state-of-the-art implementations is overly prohibitive in practice [55, §5] as this operation is shown to require runtimes in the order of *minutes*. Even by restricting the members to use a single regular expression in their import policy (a fairly restrictive assumption), evaluating the import policies in a large IXP with 500 members would take days! In contrast, SIXPACK computes and dispatches routes in the order of tens of milliseconds, improving upon the naïve approach by 5 orders of magnitude. This is made possible through a combination of several ingredients, as discussed below.

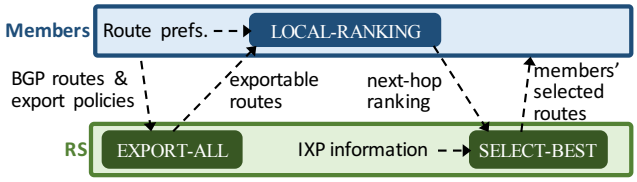


Figure 4: SIXPACK's 3-step route dispatching process.

5.2 SIXPACK Design

To achieve practical runtimes, SIXPACK is carefully designed to keep complex computation outside the SMPC, to the largest extent possible without compromising privacy (see Fig. 4). Specifically, the route dispatch computation is split into three operations to be performed sequentially, called EXPORT-ALL, LOCAL-RANKING, and SELECT-BEST. Both EXPORT-ALL and SELECT-BEST are SMPC-based components of the system that are executed within the RS by the two non-colluding entities (depicted as a single green-colored box). The LOCAL-RANKING component, in contrast, is locally executed by each member (depicted as a single blue-colored box). Next, we describe the SIXPACK pipeline for processing BGP route announcements. For readability, we assume members connect with a single BGP router (see [67] for the case with multiple routers). We observe that BGP withdrawal messages can be handled in a similar way (see [67]).

Step I: Exporting all permissible routes. SIXPACK processes streams of BGP announcements generated by the IXP members. Through EXPORT-ALL, SIXPACK takes as input a BGP route destined to a prefix π and its associated export policy and outputs the route to the IXP members authorized to see that route. This operation is performed in SMPC, and so neither the route nor the export policy is disclosed to any unintended entity. The route and its export policy are both stored in encrypted form within the RS. We discuss EXPORT-ALL in detail later in this section (§5.4).

Step II: Ranking routes based on local preferences. At this point, each member that received the new route executes LOCAL-RANKING to rank all its available routes towards π according to its (arbitrarily complex) local import policies. A key idea embedded into SIXPACK's design is performing this computationally-heavy operation outside of the SMPC, i.e., at the member-side. Now, recall that in BGP, each IXP member only announces at most one single route towards π. Thus, a ranking of the routes destined to π corresponds to a ranking of the IXP members, where a member assigns the lowest preference to those IXP members from whom it did not receive a route to π. We call such ranking the *next-hop* ranking, and this is the output of Step II.

Step III: Incorporating IXP information into route selection. When multiple routes for a certain prefix are available, members submit the next-hop ranking received to the RS service. Upon receiving a next-hop ranking from a member, the RS proceeds to run the SMPC-based SELECT-BEST component for dispatching the best-selected route to that member. The best route is computed based on the next-hop ranking and, importantly, the performance-related information available to the IXP (e.g., port utilization). This operation is performed in SMPC, and so neither the members' rankings nor the IXP performance-related information is revealed to any

unintended entity. We discuss the Select-Best in detail later in this section (§5.5).

Benefits of our design approach. Through Export-All, an IXP member gains full visibility of the available routes and can possibly select the best one according to *any arbitrary* local import policy (e.g., next-hop preferences, shortest route, avoid specific AS networks). Then, by taking part in Select-Best, the IXP member can incorporate IXP information into its route selection process. By implementing Export-All and Select-Best via carefully optimized SMPC and keeping Local-Ranking's potentially highly complex computation outside of the SMPC framework, this pipeline preserves the member's and the IXP's privacy, and is efficiently executable.

Observe that, since BGP treats each IP prefix destination independently, multiple instances of the two RSes can be easily instantiated to dispatch routes in parallel, thus enhancing the system throughput. However, multiple route announcements towards the same IP prefix have to be processed sequentially.

Peering at multiple sites/IXPs. We have so far assumed that each organization has a single point of presence at an IXP. In practice, organizations may connect at different physical locations at the same IXP with the same goal of further reducing latencies. In sixpack, each connection to the same IXP from the same member is treated as an independent member. This allows operators to arbitrary export/rank/filter BGP routes at those locations independently.

Before we get into the details of the two SMPC-based components of sixpack, we first formalize our model of export policies (and next-hop rankings) in §5.3. We then describe Export-All in §5.4 and Select-Best in §5.5. Due to space limit, a formal description of our protocol and proofs of its security are given in Appendix F.

5.3 Routing Policies Model

Export-policy. The Export-All component of sixpack dispatches routes according to the export policies pertaining to the routes that it received from its members. Each route carries its own export policy specification, i.e., the set of members to whom that route can be exported. Since BGP computes routes independently for each destination IP prefix, w.l.o.g., we henceforth assume through this section that there only exists a single destination IP prefix π. Moreover, the BGP route computation only depends on the last route announced by a neighbor. Hence, our model only needs to store the set of routes *currently* available at the RS and the *currently* specified export policies. To achieve efficient SMPC computation, we model BGP export policies as follows. Let $M = \{m_1, \ldots, m_{|M|}\}$ be the set of IXP members and $R = \{r_1, \ldots, r_{|R|}\}$ be the set of available routes. We define the *export policy matrix* P, with $|M|$ rows and $|R|$ columns. Entry $P_{i,j}$ in the matrix, for $1 \leq i \leq |M|$ and $1 \leq j \leq |R|$, is 1 if route r_j is exportable to member m_i and 0 otherwise.

As an example of an export policy matrix, consider Fig. 3(a) on p. 3 where m_A, m_B, m_C, m_D are IXP members and r_A, r_B are routes announced by m_A and m_B, respectively. While route r_A is exported to m_C only, route r_B is exported to m_C and m_D. Observe that r_A and r_B are not exported to m_A and m_B, respectively, i.e., to the IXP member they originate from.

In the Export-All component, all permissible routes are exported. Each IXP member m_i should receive each route r_j for which $P_{i,j} = 1$. In Fig. 3(a), m_C receives both r_A and r_B, and m_D receives only r_B, while m_A and m_B do not receive any route. In the Select-Best component, each member m_i is entitled to receive any route r_j with $P_{i,j} = 1$. The actual route that will be received depends on the ranking over routes and IXP performance-related information.

Next-hop ranking (a.k.a. local preferences over routes). Select-Best receives as input, from each participating IXP member, its next-hop ranking with respect to the destination IP prefix. Each next-hop corresponds to a route announced by a member, thus next-hop rankings model *local preferences* over the received routes. For each IP prefix π, we model preferences over the available routes at the RS as a matrix Ψ with size $|M| \times |M|$. Each element $\psi_{i,j}$ of that matrix represents member m_i's local preference (value) for routes announced by m_j, where routes announced by members with higher local preference are preferred over routes announced by members with lower local preference. Using Select-Best, each IXP member m_i thus receives a single route r announced by m_j such that $\psi_{i,j}$ is the highest priority value in row i of Ψ, for which $P_{i,j} = 1$ holds. Ties are broken deterministically.

The matrix Ψ can easily be extended to represent preferences over routes based on a combination of members' local preferences and IXP performance-related recommendations. For instance, if each preference value ψ is encoded as an $\rho = 8$-bit integer, we can use the four most significant digits of ψ to create 16 different classes of members' local preferences over routes and use the four least significant bits to create another 16 additional classes for the IXP performance-related recommendations. In this way, the IXP information is used only to break ties among routes with the same rank. Alternatively, IXP information can be given higher priority and members' local preferences used to break ties.

5.4 The Export-All Component

Through Export-All, the RSes export to each member all permissible (i.e., exportable) routes while keeping each member's export policy private. We note that this problem could also be solved by using public-key cryptography if we assume that each sending member knows the public keys of all other IXP members. However, a public-key solution alone cannot incorporate IXP performance-related information without revealing it – a concern for 60% of the surveyed operators. Furthermore, SMPC circumvents all key management challenges, protects against side-channel attacks, and easily integrates with the Select-Best SMPC component.

Observe that, in the Export-All component, not only is the computation *per-prefix independent*, i.e., the computation is executed independently for each destination IP prefix, but it is *per-route independent*, in the sense that the announcement of a specific route to a member does not depend on what other routes to the same prefix are announced to that member. Hence, w.l.o.g., Export-All is described below with respect to a single route to a single prefix π.

Fig. 5 illustrates an example of the Export-All computation. Two independent RSes, RS_1 and RS_2 (center of the figure), perform the redistribution of a route from m_A according to its export policy.

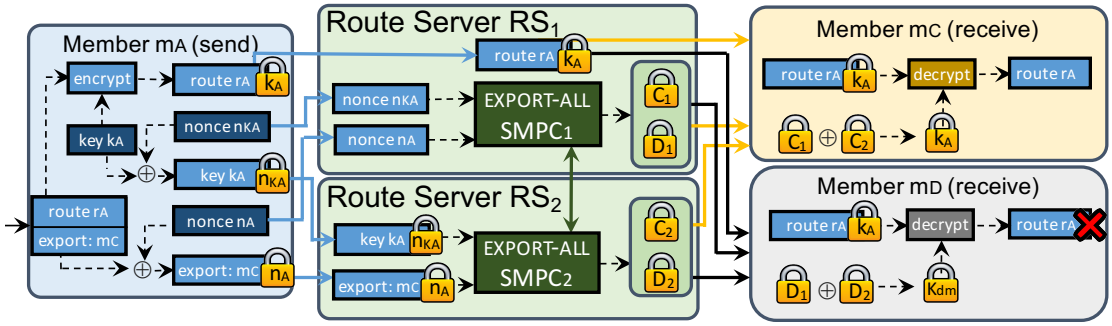
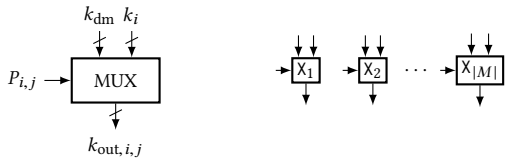


Figure 5: The EXPORT-ALL component. The RSes export to IXP members all permissible routes.



(a) Circuit Building Block X_j **(b) Circuit Structure**

Figure 6: The EXPORT-ALL circuit is a set of $|M|$ multiplexers, each of which outputs either the valid or dummy key depending on entry $P_{i,j}$, where i (j) is the member announcing (receiving) a route. Since we process the inputs in a SIMD fashion, the routes for all receiving members are processed in parallel. If only an incremental update to a single route is processed, the circuit consists only of a single MUX.

We consider the scenario presented in Fig. 3(a). The computation operates over a policy that is kept private using SMPC and the route is dispatched in such a way that neither the RSes nor the IXP members can distinguish whether a route is announced to any other member or not. Each member attempts to decrypt the information received from the RSes (right side of the figure). This operation succeeds iff the route is actually exported to that member, which then learns the route. We now discuss in more detail the different parts of EXPORT-ALL.

Encrypted routes. Each route that needs to be distributed is first encrypted. W.l.o.g., we assume that member m_A wants to send a route r_A as shown in Fig. 5. Then, m_A encrypts r_A using a route-specific key k_A and a symmetric encryption scheme (we use AES) and sends the route to RS_1, which, in turn, redistributes it to all the members. These can decrypt r_A only if they possess the route key k_A. SIXPACK guarantees that an IXP member receives the key k_A only if the route can be exported to it. We use a "dummy key" k_{dm} to notify a member when a route cannot be exported to it. Recall that a receiving member does not have visibility of the routes in plain text, so it does not know which routes are announced. Even if a member colludes with one of the two RS entities, it cannot distinguish whether a certain route is exported to any of the other IXP members.

Exporting keys via SMPC. We now leverage SMPC to dispatch the key k_A in a privacy-preserving manner. Specifically, we devise a tailored SMPC circuit (shown in Fig. 6) that is jointly executed by two SMPC entities: RS_1 and RS_2. The EXPORT-ALL circuit consists of one multiplexer X_i per member m_i, which outputs either the valid or dummy key k_{dm} depending on the export policy entry $P_{i,j}$ (see 5.3), where i (j) is the member announcing (receiving) a route. Since we process the inputs in a SIMD fashion, the routes for all receiving members are processed in parallel MUX blocks. If only an incremental update to a single route is processed, the circuit consists only of a single MUX.

To generate the SMPC input (left of Fig. 5), member m_A creates a random nonce n_A and XORs wit r_A's export policy. The result is sent to RS_2, while n_A is sent to RS_1. Observe that neither RS_1 nor RS_2 is able to decrypt the export policy as they are assumed to not collude. Both RSes store the share of the export policy received by the member. Analogously, m_A generates a nonce n_{KA} that is XORed with key k_A. The result is sent to RS_2, while n_{KA} is sent to RS_1. We show in Fig. 3 on page 3 an example of (a) the export policies of two routes r_A and r_B, (b) the nonces chosen by m_A and m_B, respectively, and (c) the resulting inputs to RS_2.

Once the two RSes receive their shares of the export policy and key k_A, $SMPC_1$ and $SMPC_2$ (center of Fig. 5) output to every member m_X two shares X_1 and X_2 of the output, respectively. XORing X_1 with X_2 (right of Fig. 5) produces a key that can be used to decrypt the encrypted route r_A if and only if $P_{X,A} = 1$, (i.e., route r_A can be exported to member m_A), or the dummy key k_{dm}, otherwise. In Fig. 5, m_C receives C_1 and C_2; when XORed, they give k_A. Similarly, m_D receives D_1 and D_2; when XORed, they give k_{dm}, leading m_D to discard the encrypted route r_A. Note that our design of the EXPORT-ALL circuit can execute on multiple routes at once from different members for more efficiency (see Fig. 6(b)). In fact, EXPORT-ALL takes as input an export policy matrix P (§5.3), which may consist of just one column (i.e., one route) as a special case.

5.5 The SELECT-BEST Component

To leverage the superior IXP's visibility into dataplane conditions, we design SELECT-BEST, a privacy-preserving component that allows IXP members to select the best permissible route according to both *their own* local preferences and the *IXP* performance-related information.

To execute SELECT-BEST, each IXP member will first translate its ranking of available routes (e.g., prefer shortest routes) to a ranking of the corresponding IXP members that announced them, where members that are neither exporting nor announcing a route to this member are assigned the lowest preference. Analogously, the IXP translates its sensitive performance-related information, such as

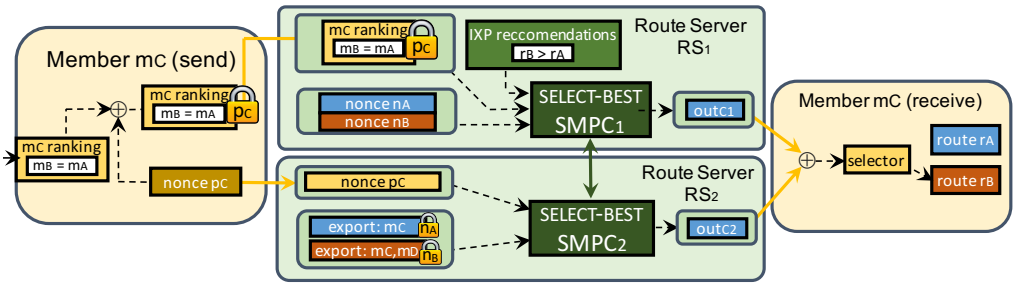
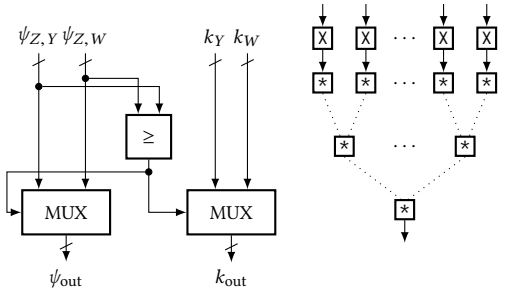


Figure 7: The SELECT-BEST component. The RSes export to each IXP member the best preferred permissible route.

k	10010101	=	m_C	01001000	⊕	m_C	11011101
k_{dm}	00000000		m_D	11100101		m_D	11100101
(a)			(b)			(c)	

Figure 8: Example keys (a) used for encrypting routes, (b) output from RS_1, and (c) output from RS_2. In our implementation route keys are 128 bit AES keys.



(a) Route comparison circuit ⋆ (b) Tree structure circuit

Figure 9: The SELECT-BEST circuit for exporting the single highest ranked route key to a member m_Z. Each ⋆ circuit compares 2 routes r_Y and r_W announced by m_Y and m_W, resp., where $\psi_{Z,Y}$ and $\psi_{Z,W}$ are the m_Z's preferences over r_Y and r_W, resp., while k_Y and k_W are the keys of routes r_Y and R_W, resp. The multiplicative depth of the SELECT-BEST circuit, for a priority value of ℓ bit, is $\lceil \log_2(\#\textbf{Routes}) \rceil \cdot (\lceil \log_2(\ell) \rceil + 2) + 1$.

members' port utilization, to preference values. As explained in §5.3, these preferences values can be combined into a single preference value ψ per route, where we envision the members' local preference to be given precedence over the IXP's inputs. This information is provided as input to SELECT-BEST, which computes the best route in a privacy-preserving manner. We note that this functionality *cannot* be realized with public-key cryptography alone and is thus an interesting and practical real-world application of SMPC.

Refer to Fig. 7 as an example of the SELECT-BEST computation, where we consider the export policy scenario of Fig. 3(a). Namely, two members m_A and m_B announced two routes r_A and r_B, respectively. Through EXPORT-ALL, m_C received both r_A and r_B while m_D only received r_B, with the latter deciding not to execute SELECT-BEST. Based on port utilization levels and assuming m_C equally ranks r_A and r_B, the IXP gives route r_B a higher preference.

Choosing the best route via SMPC. We now leverage SMPC to select the best route of each IXP member in a privacy-preserving manner. To this end, we devise a tailored SMPC circuit (shown in Fig. 9) that, for each member m, takes as input the next-hop ranking (i.e., preferences over members) and the IXP route recommendations, and outputs to m the identifier of the best route.

The first step (left of Fig. 9b) is similar to the EXPORT-ALL circuit: based on the export policy, the preference of all non-exportable routes is set to zero. After that (right of Fig. 9b), we feed the resulting priorities as well as the route keys (which are used as identifiers) into a *MaxIdx* tree circuit [60, §3.3] and thus determine the best route, i.e., output the route key with the highest preference. We also input the dummy key k_{dm}, which is returned if the receiving member does not have any permissible route. The comparison among two routes r_Y and r_W (announced by members m_Y and m_W, resp.) is performed by the $\star$ circuit (Fig. 9a), where $\psi_{Z,Y}$ and $\psi_{Z,W}$ are the m_Z's preferences over r_Y and r_W, resp., while k_Y and k_W are the keys of routes r_Y and r_W, resp. The selection bit of the MUX is chosen such that the route key with the higher preference value gets propagated to the next level of the tree. The multiplicative depth of the SELECT-BEST circuit, for a priority value of ℓ bit, is $\lceil \log_2(\#\text{Routes}) \rceil \cdot (\lceil \log_2(\ell) \rceil + 2) + 1$.

Both input and output are considered private information, and are not visible to the RS in clear. The IXP members are responsible for generating and then reconstructing the SMPC's input and output through secret-sharing.

The input to the SMPC (i.e., the next-hop ranking) is generated similarly to the input of the EXPORT-ALL component. In the example (left side of Fig. 7), m_C generates an $|M| \cdot \rho$-bits nonce p_C that is XORed with its next-hop ranking (i.e., row ψ_C of the ranking matrix Ψ, §5.3). The XORed result is sent to RS_2, while p_C is sent to RS_1. Neither RS_1 nor RS_2 are able to decrypt the ranking as they are assumed to not collude.

At this point, the two RSes combine the member's preferences with the IXP preferences, where only RS_1, which runs at the IXP supplies the preferences based on performance information; RS_2 uses a vector of zeros. The two RSes execute the SELECT-BEST circuit on the given inputs (center of Fig. 7). Then, m_X receives two values out_{X1} and out_{X2}; when XORed, they produce an identifier of the best route. In Fig. 7, m_C receives r_B as the best route based on the IXP performance-based preference. Observe that if m_C had preferred r_A over r_B, it would have received an identifier to r_A.

As for the other circuit, it is worth noting that our design of the SELECT-BEST circuit can process multiple next-hop rankings at once from different members for improved efficiency. In fact,

SELECT-BEST takes as input a ranking matrix Ψ (§5.3), which may consist of just one row as a special case.

6 IMPLEMENTATION

We implemented the two SMPC components shown in Fig. 5 and Fig. 7 in C++ using the ABY framework [29] in 700 LoCs. ABY is written in C++ and provides low-level primitives for building Boolean circuits that are evaluated efficiently with the GMW protocol. We rely on the GMW protocol, which was shown to be beneficial in several practical cases [80], for two main reasons: (1) GMW precomputes *all* cryptographic operations in an input- and even function-independent setup phase, which can be parallelized and computed at any time before the private inputs are known; and (2) GMW also offers performance benefits in certain scenarios, such as SIMD (single instruction multiple data) processing and very efficient vector gates. This allows us to process multi-input AND and MUX gates very efficiently and can work on an identical sub-circuit in parallel, cf. [80]. Currently, we do not minimize the BGP update processing time by precomputing the SMPC setup phase. We however make heavy use of the features of SIMD processing.

With ABY's SIMD evaluation, we can run our two SMPC circuits for inputs from arbitrarily many members in parallel while vector operations allow the efficient processing of long bit strings, such as the route keys. The circuits that we built are optimized to process multi-bit values efficiently, which benefits the processing of route keys and comparing preference values. The SELECT-BEST circuit is evaluated in a tournament fashion by arranging the preference comparison gates in a tree. Thereby we achieve a circuit depth that grows logarithmically in the number of members, thus resulting in optimized latency for GMW. We designed our circuits to be as minimalistic and performant as possible, while still being as expressive as needed for our use cases. Our circuits have a low depth, thus achieving good runtimes for evaluation with GMW. Additionally, we extended the ABY framework with native input and output operations for outsourced SMPC, which saves two rounds of communication for input sharing and output reconstruction.

We rely on the proven security of a symmetric cipher for encrypting/decrypting routes, which we instantiate with 128-bit AES in CTR mode. We verified that AES adds negligible overhead compared to the SMPC, which holds in general and especially on machines with the AES-NI instruction set.

Our route server service, which wraps the SMPC components and handles the distribution and processing of all the BGP update information among the IXP members, is implemented as 1,800 LoCs in Python. The RSes run as independent processes, each executing its own instance of SMPC as a daemon subprocess, and communicate via TCP sockets. To improve the efficiency of SELECT-BEST, we observe that, in practice, it is convenient to batch several next-hop ranking messages together before executing SELECT-BEST as shown in our evaluation (§7.2).

Although we did not optimize our implementation to the fullest extent possible, our evaluation (§7) shows that our approach already scales to the size of the largest IXPs in the world. We discuss deployment consideration in §8. Further details on the implementation are provided in our full paper [67].

7 EVALUATION

We evaluate our SIXPACK prototype to demonstrate that our approach is both feasible and practical. We first provide insights into the performance of the SMPC part of SIXPACK by performing micro benchmarks across a realistic range of numbers of IXP members and inputs to our circuits. We then evaluate our system by replaying a real trace of BGP announcements from one of the largest IXPs worldwide and by performing a stress test. Our results highlight the following:

(1) While SMPC is (as expected) the costliest part performance-wise, our results show that the online phase is even at worst below 38 ms. The *maximum* setup and online runtime we measured in our evaluation were 131.9 ms and 37.2 ms, respectively for 32 inputs in the SELECT-BEST component.

(2) Our still unoptimized SIXPACK prototype achieves BGP processing times below 90 ms at the 99^{th} percentile, and, specifically, below 23.6 ms and 62.8 ms for EXPORT-ALL and SELECT-BEST at the 99^{th} percentile, respectively. Furthermore, we measured negligible bandwidth requirements. Finally, SIXPACK processes a full-routing-table of 250 K prefixes in ≈11 minutes, comparable to today's RSes (see §7.3). We stress the fact that our prototype can fairly easily be improved to achieve better performance by precomputing the SMPC setup phase.

It is worth comparing these numbers with the convergence time of BGP on the Internet, which can be in the order of minutes [72] , that is, several order of magnitude higher than time overhead due to dispatching routes with SIXPACK.

7.1 IXP Dataset

We assess our system using a two-hour trace of BGP updates from one of the largest IXPs worldwide, which interconnects more than 600 members. It contains 25,676 BGP update messages, consisting of 76,506 IP prefix announcements and withdrawals. The average number of BGP updates per second is 3.57, the first and third quartiles are 2 and 4, respectively, while the minimum and maximum numbers per second are 1 and 29, respectively. The average number of IP prefixes announcements or withdrawals per second is 10.62, the first and third quartiles are 6 and 12, respectively, while the minimum and maximum numbers per second are 2 and 379, respectively. In addition to that trace, our data also contains a snapshot of the RS routing table at the beginning of the trace of updates. The routing table contains roughly 400,000 routes towards ≈240,000 IP prefixes. In Fig. 10(a), we use a CDF to show what fraction of the announced IP prefixes (y-axis) are reachable through no more that a certain number of routes (x-a xis). We observed that for more than half of the IP prefixes there exists a single available route, with an average of 1.9, the 95^{th} percentile of 5 and a maximum of 25 routes per prefix. In Fig. 10(b), we use a CDF to show what fraction of the IXP members (y-axis) announced no more than a certain number of routes (x-axis). We observed that on average each member announces 626.5 routes, the 95^{th} percentile is 1581 and the maximum is roughly 150,000.

7.2 SMPC Microbenchmarks

To benchmark the SMPC circuits, we consider the scenario of an IXP with 750 members, which is ≈ 1.5 times the number of members

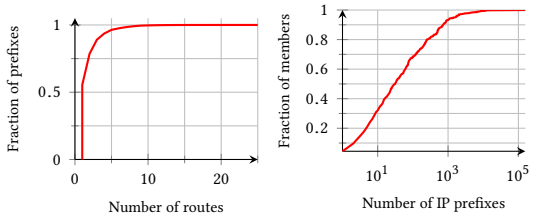


Figure 10: (a) CDF of the number of routes announced for each IP prefix (b) CDF of the number of routes announced by each member.

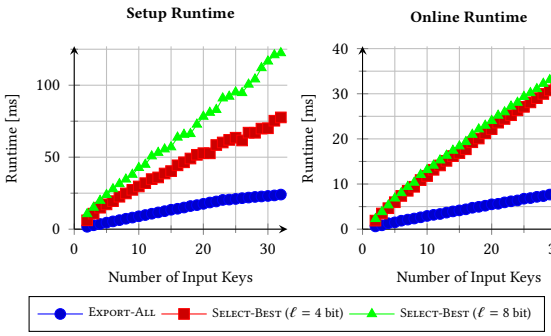


Figure 11: SMPC micro benchmarks: Median runtimes for setup and online phase of both approaches from §5.

Table 1: SMPC micro benchmark overview.

Approach	# Inputs	Setup phase Runtime[ms]	Setup phase Comm.[KiB]	Online phase Runtime[ms]	Online phase Comm.[KiB]
Export-All	2	1.7 (±8 %)	28	0.6 (±17 %)	25
Select-Best	4	19.8 (±4 %)	1,492	5.3 (±4 %)	106
	8	34.6 (±6 %)	3,469	10.3 (±3 %)	247
	16	63.7 (±9 %)	7,409	19.3 (±3 %)	528
	32	122.4 (±8 %)	15,286	35.9 (±1 %)	1,091

connected to the RS encountered in the IXP dataset we analyzed. From the dataset, we observe that at most 27 members export a route for the same IP prefix. Thus, we run benchmarks for a number of route keys up to 32.

We measure the runtime of setup and online phase of the SMPC. The setup phase is independent of both the private inputs and the function being evaluated and can be precomputed at any time before the actual circuit evaluation. One invocation of SMPC corresponds to processing a single IP prefix announcement. For all experiments, we perform 50 executions of the circuit and report median values of the runtimes and their standard deviation. Measurements were performed on two servers with a 2.6 GHz CPU and 128 GiB RAM, connected via a local 10 Gbps network. We evaluate with a preference value bit length of 8 bits. We use random preferences as the SMPC performance does not depend on the actual input (otherwise it would be vulnerable to side-channel attacks). The reported communication is the sum of sent and received data by one party.

Runtimes. Tab. 1 shows the setup and online runtimes as well as the required communication. More detailed results, covering circuit gate counts, circuit depth, and required communication are shown in our technical report [67].

Our results show that the setup phase takes between 1.7 ms and 122.4 ms, and depends on the number of inputs processed and the circuit used. In the best case, an IP prefix is announced by a single member and the Export-All component can be used to dispatch the announcement to the legitimate members since no route comparison is needed.

In the Export-All component, each route is processed independently, even those towards the same destination IP prefix. This case corresponds to the computation with just 2 inputs (i.e., a given route and the dummy one) and requires an online computation of only 0.6 ms and an amount of transferred data of 25 KiB. However, as both setup and online runtimes grow sub-linear with the number of routes, it is beneficial to compute on many routes in parallel, e.g., at times where several BGP announcements happen simultaneously. The runtimes of Select-Best are large due to the deeper SMPC circuit. Note that Select-Best may not be used when an IP prefix is announced by a single route (i.e., 2 inputs).

We also verified that our SMPC circuits scale linearly with respect to the number of members, i.e., with 1,500 members, Select-Best takes less than 70 ms to process 32 routes.

Memory Consumption. We further measure the memory consumption of our SMPC implementation. Even when processing large inputs of 32 routes and 2,000 members, the memory consumption of the dispatching operation, which is consumed only during a route dispatch, remains below 15 MiB. We determined that memory consumption grows sub-linear with increasing parameters, which shows that our implementation will remain practical in the future.

In summary, these performance numbers confirm the practicality of our SMPC implementation and SIXPACK.

7.3 Prototype Evaluation

To assess the feasibility of SIXPACK, we focus on evaluating its two main building blocks, i.e., Export-All and Select-Best, against a real-world trace of BGP updates collected from one of the largest IXPs in the world. To assess SIXPACK's scalability, we performed a stress test of Export-All and Select-Best and considered edge case scenarios such as the connection of a new member to the IXP network.

Experimental setup. We performed our experiments on three servers with 16 hyper-threaded cores at 2.6 GHz with 128 GiB of RAM and Ubuntu Linux 14.04, each two connected through 10 Gbps links. The average latency is 100 μs, similarly to latencies reported in co-location data centers [4]. We use one server to replay the stream of BGP updates from our dataset and to handle the receiving part of the SIXPACK mechanism. In each experiment, we load the full RS routing table contained in our dataset into the two RS instances.

Each RS instance runs on a server and consists of a set of *worker* processes orchestrated by a single *handler* process. The latter one (i) redistributes received BGP updates to the workers, (ii) guarantees that the other RS instance also knows which worker must be used to process a BGP update, (iii) synchronizes SIXPACK's operations, (iv) redistributes the SMPC outputs to the members, and (v) guarantees that two BGP updates for the same IP prefix are not processed simultaneously by two different workers. Each worker runs an instance of the SMPC party and computes its inputs.

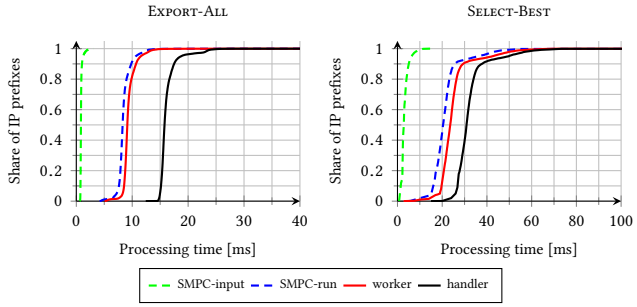


Figure 12: Processing time CDFs for Export-All and Select-Best components with 8 workers per RS instance.

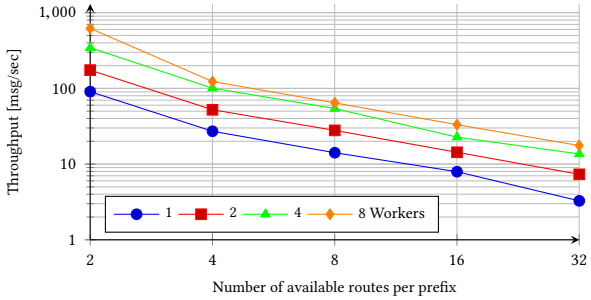


Figure 13: Throughput test of sixpack for different number of parallel workers.

Each BGP update can be either an announcement or a withdrawal of a set of IP prefix destinations that share the same export policy. In order to evaluate Select-Best, we assign random local preferences among all the IXP members.

Performance analysis on real-world BGP trace. For each BGP route announcement r, we measure the amount of time required to create the input values for the SMPC processing, the time required to run the SMPC computation, the amount of time the announcement is handled by a worker process, and the total time between the reception of r and the transmission of its output to the IXP members.

We plot the processing time of each IP prefix from our dataset in Fig. 12. We observe similar trends in both Export-All and Select-Best (i.e, Step I and Step III of sixpack), with the latter one being 2 times slower than the former one. In both graphs, we see a large gap between worker and handler times. This is mostly due to the unoptimized utilization of shared data structures in Python. Second, worker execution is dominated by the SMPC processing, with SMPC input creation requiring only a negligible amount of computational resources. We also measure the time required to encrypt and decrypt routing announcements, finding AES operations to be a negligible amount of time compared to the handler.

To summarize, the average processing times are 15.9 ms and 32.7 ms for Export-All and Select-Best, respectively, while the 99^{th} percentiles are 23.6 ms and 62.8 ms, respectively. This low running time reflects the fact that most of the routes are announced by few members (see §7.1),The performance of Select-Best is dominated by the SMPC computation. We recall that the SMPC could be further optimized by precomputing the setup phase ahead of time.

We also measured the amount of communication required by sixpack during the experiment. We found that for both Export-All and Select-Best the average bandwidth requirements from an IXP member to the RSes are negligible (i.e., 20 kbps). The requirements are higher for the communication between the two RSes. In the Export-All component, the average bandwidth requirement is less than 2.79 Mbps while in Select-Best it is no more than 10.9 Mbps. These figures show that even a 1 Gbps link between the two RSes would be more than sufficient for supporting sixpack.

Stress test. To assess the scalability of our system, we flooded sixpack with announcements and counted the number of routes dispatched per second. Our evaluation follows two dimensions: number of routes announced for the same IP prefix (including a dummy route) and number of parallel workers and plotted our results in Fig. 13. We observed that sixpack processes 619.28 route announcements per second in the Export-All component and 64.33 announcements per second for IP prefixes announced through 8 routes. We observe that eight workers in parallel provide limited improvements for computations of more than a single announced route. We observed that sixteen parallel workers allow sixpack to process over 1,010 routes per second with Export-All (not shown in the graph) but do not improve performance for routes announced by more than one member (i.e., in Select-Best). The bandwidth requirements at full-speed are always below 1.5 Gbps.

Connecting a new member: comparing with today's RSes. When a member M connects to the RS, either because of a newly established connection or after recovering from a failure, two operations are performed: (i) the RS propagates to the new member all the best permissible routes towards the IP prefixes that were previously announced by the other members and (ii) the RS recomputes and propagates to all the members the best route for each IP prefix announced by M. As for operation (i), at large IXPs, ≈250 K best-route computations must be performed, one for each prefix known to the RS. While this may sound problematic, as observed in Fig. 10(b), most of the IP prefixes are announced by a single member, hence enabling us to consistently leverage the Export-All component. Moreover, each best route is computed only for one IXP member and not for all of them. This allows us to considerably speed up the SMPC execution. For instance, while executing the Select-Best component for 500 members with 32 routes takes on average 158 ms, the same computation for a single member takes just 9 ms. We verified that operation (i) takes on average 92.8 s with our dataset. As for operation (ii), in Fig. 10(b), we observed that most of the customers do not announce more than 1K BGP routes. Our stress test shows that such operations would not take more than a few seconds. For large customers announcing ≈250K prefixes, we verified that the announcement operation takes roughly 11 minutes. In comparison, today's RSes report convergence times ranging between 3 and 10 minutes [2, 27, 79], even without incorporating import policies, performance-driven information, and privacy functionality.

8 FREQUENTLY ASKED QUESTIONS

Is sixpack ineffective since routing policies can be easily inferred? No. While many techniques have been designed to infer peering relationships, and even routing policies, in the Internet, using control- and data-plane traffic information [21, 30, 31, 54, 70,

82, 87], such solutions not only have limited accuracy [77] (e.g., globally visible AS-paths neither reveal local preferences nor "negative" export policies, i.e., not exporting a route) but might also be detectable (e.g., BGP AS-path spoofing [54]). Finally, SIXPACK supports ranking routes based on the IXP members' port utilization, a fundamental performance metric that is challenging to infer in practice [24].

How do you envision SIXPACK being adopted by IXPs? Traditional RS service and SIXPACK can coexist. Each member can choose whether to share its routes in a privacy-preserving manner or not. Observe that the routes shared via the traditional RS can be forwarded to SIXPACK so that the early adopters have the same route visibility of the members peering at the traditional route server. We argue that the desire to maintain peering relationships private will incentivize adoption. Moreover, SIXPACK only requires small modifications at both the IXP and member side (i.e., route servers and BGP border routers). Members could run SIXPACK as a BGP proxy that receives updates intended for the RS from its BGP border router and performs the SMPC input sharing process. Similarly, the BGP proxy will receive the output of the SMPC from the two SMPC-enabled route servers and translate it into BGP updates that will, in turn, be forwarded to the member's BGP border router. Members not using SIXPACK require no changes to their infrastructure.

We believe that bringing innovation to today's networks is no longer as prohibitive a task as it has been in the past. The rise of programmable networks with SDN is boosting innovation in different environment such as data centers [83] and, more recently, IXP networks [3, 26, 41, 42, 63, 84]. Deploying SIXPACK in a so-called Software-Defined-eXchange (SDX) could provide the opportunity for using custom interfaces and avoid the need for compatibility with BGP.

Why not use Intel SGX in place of SMPC? Software Guard eXtensions (SGX) is an instruction set [22, 53] that allows programmers to perform computation on data stored within private regions of memory that are not accessible by unauthorized processes. Despite its promise, SGX is currently the subject of many discussions regarding its real level of security. In contrast, SMPC is a well-established methodology with proven security guarantees. A major concern regarding SGX programs is that timing or memory access patterns leak information about the private inputs (SGX does not include any mechanism for coping with such leaks) [14]. While ORAM techniques can be used to mitigate these concerns [85], this comes at the price of increased complexity and non negligible obstacles to scalability. Our SMPC approach, in contrast, does not suffer from these problems. Another general concern regarding SGX is Intel's role as the centralized point of trust. SMPC allows any two entities to guarantee privacy-preservation. Because of the above SGX limitations, recent studies propose combining SGX with SMPC to strengthen the privacy of outsourced computation [43, 59]. Thus, we consider SGX as complementary to our solution. Our results establish that SMPC alone is also viable solution in the RS context.

Does SIXPACK nullify the benefits of RSes? No. SIXPACK preserves centralized route computation while tackling 3 out of 4 concerns from operators with RSes (i.e., route visibility, best route control, privacy), enhancing RS functionality with performance information, and retaining easy management.

Does SIXPACK leak information? Members only learn that (encrypted) routes are announced. The RSes also learn IP prefixes and announcing members of each encrypted route.

Does SIXPACK prevent IXPs from providing RPKI route validation services?No. Members can either reveal the IP prefix of a route and its originator so as to allow the IXP to validate that information or, alternatively, RPKI validation can easily be implemented within the SMPC framework as it only involves a simple lookup operation on a dictionary. We show a possible implementation in [67].

9 RELATED WORK

Great efforts are invested in making SMPC a practical approach with real-life applications. Indeed, SMPC has been applied to enhance privacy in a broad spectrum of applications, including auctions [13], financial data analysis [12], statistical data analysis [11], and detecting tax fraud [10].

In the networking realm, the first works to apply an SMPC approach to interdomain routing were [6, 44]. These works study how BGP routes across the whole Internet can be *computed* in a privacy-preserving manner. While a new and exciting direction, these results limit BGP expressiveness and the achieved runtimes are impractical, even on small topologies. Instead, we focus on IXPs, the crucial crossroads of the Internet that run computation on private, business-sensitive routing information. We argue that applying SMPC to this narrower context is a promising approach to privacy-preserving interdomain route-computation and we built a prototype that handles real-world traces from a large IXP. Indeed, the technical challenges studied here are very different from those in [44]; in particular, we do not *compute* routes across a multihop network. We rather solve the simpler problem of *exporting and ranking* sets of available routes, based on confidential business information. In contrast to [6, 44], this allows us to achieve substantially smaller circuit sizes that result in practical runtimes in real-world environments.

Another routing-related study is SPIDER [92], a distributed mechanism for verifying if a peering agreement between ASes (involving, e.g., a requirement to always export the shortest route available) is respected by the involved parties without revealing control-plane information (e.g., which routes are available). Applying SPIDER to our context could aid IXP members in verifying that the RS is indeed executing the protocol (in contrary to, e.g., selecting an un-optimal route for each member). Our focus in this paper is different: we guarantee that the RS does not learn anything about members' export and import policies. Finally, while SGX can be used to preserve the privacy of interdomain routing policies [56, 65], we discussed its limitations in §8.

Past studies utilized SMPC to address privacy concerns in a variety of other networking-related problems [15, 18, 69, 78]. Unlike these studies, our focus is on guaranteeing the privacy of peering policies. Additionally, the protocols proposed in these studies are either evaluated under questionable conditions, or exhibit runtimes in the order of seconds, whereas RSes are required to operate at faster runtimes.

Homomorphic encryption schemes [32, 33, 39] are crypto systems that computes over encrypted data. However, such general

schemes are slow for practical applications [34]. Even implementations tailored to simple regular expression evaluation, as those presented in BlindBox [81], are either too restrictive for supporting SELECT-BEST, i.e., by only providing exact string match, or leak information whenever a match is found, which corresponds to the IXP learning routes whenever a best route is selected by a member. Finally, [45] designs a shortest-path vector-based routing protocol that uses homomorphic encryption but is not as expressive as BGP.

We note that in dealing with the *privacy* of export policies, our work solves an orthogonal problem to that of securing BGP routing from IP-prefix hijacks and BGP path-manipulation, which is an active topic of research [17, 35, 48]. Finally, while higher visibility over routes has been proposed(e.g., BGP Add-Paths [49]), when deploying such techniques at the RS, no confidentiality about the export policies and IXP performance-related information is guaranteed.

10 CONCLUSIONS

We presented SIXPACK, a privacy-preserving IXP RS design with provable guarantees. SIXPACK dispatches routes according to highly expressive members' routing policies and IXP performance-related information. We showed that an *efficient* realization of SIXPACK with Secure Multi-Party Computation can be attained through a careful redistribution of the route dispatching responsibilities between the RS and IXP members. We devised optimized SMPC circuits tailored to RS computation. We built a SIXPACK prototype and assessed its practical feasibility with a real-world trace of BGP updates collected from one of the largest world-wide IXPs.

ACKNOWLEDGMENTS

We thank the anonymous reviewers and our shepherd, David Barrera, for their valuable feedback on our paper. We thank Josh Bailey, Xenofontas Dimitropoulos, Timothy G. Griffin, Panos Kalnis, Yatish Kumar, David Meyer, and Jennifer Rexford for their valuable feedback to the project. This research is (in part) supported by European Union's Horizon 2020 research and innovation programme under the ENDEAVOUR project (grant agreement 644960). This work has been co-funded by the German Federal Ministry of Education and Research (BMBF) and by the Hessen State Ministry for Higher Education, Research and the Arts (HMWK) within CRISP, and by the DFG as part of project S5 and E3 within the CRC 1119 CROSSING. Michael Schapira is supported by an ERC Starting Grant.

REFERENCES

[1] B. Ager, N. Chatzis, A. Feldmann, N. Sarrar, S. Uhlig, and W. Willinger. Anatomy of a large European IXP. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2012.
[2] Follow up: AMS-IX Route-Server Performance Test Euro-IX 20th, 2012. https://ripe64.ripe.net/presentations/49-Follow_Up_AMS-IX_route-server_test_Euro-IX_20th_RIPE64.pdf.
[3] AMS-IX: Megaport and AMS-IX Partner to Provide Global SDN-Enabled Elastic Interconnection and Internet Exchange Service, Jan. 2016. https://ams-ix.net/newsitems/233.
[4] AMS-IX: Real-time-statistics, Feb. 2016. https://ams-ix.net/technical/statistics/real-time-stats.
[5] Amdsterdam internet exchange infrastructure, 2017. https://ams-ix.net/technical/ams-ix-infrastructure.
[6] G. Asharov, D. Demmler, M. Schapira, T. Schneider, G. Segev, S. Shenker, and M. Zohner. Privacy-preserving interdomain routing at Internet scale. *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2017(3), 2017.
[7] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner. More efficient oblivious transfer and extensions for faster secure computation. In *ACM Conference on Computer and Communications Security (CCS)*, pages 535–548. ACM, 2013.
[8] D. Barrera, L. Chuat, A. Perrig, R. M. Reischuk, and P. Szalachowski. The SCION Internet Architecture. *Communications of the ACM*, 60(6):56–65, 2017.
[9] D. Beaver. Efficient multiparty protocols using circuit randomization. In *CRYPTO*, volume 576 of *LNCS*, pages 420–432. Springer, 1991.
[10] D. Bogdanov, M. Jõemets, S. Siim, and M. Vaht. How the Estonian tax and customs board evaluated a tax fraud detection system based on secure multiparty computation. In *Financial Cryptography and Data Security (FC)*, volume 8975 of *LNCS*, pages 227–234. Springer, 2015.
[11] D. Bogdanov, L. Kamm, S. Laur, P. Pruulmann-Vengerfeldt, R. Talviste, and J. Willemson. Privacy-preserving statistical data analysis on federated databases. In *Annual Privacy Forum*, volume 8450 of *LNCS*, pages 30–55. Springer, 2014.
[12] D. Bogdanov, R. Talviste, and J. Willemson. Deploying secure multi-party computation for financial data analysis - (short paper). In *Financial Cryptography and Data Security (FC)*, volume 7397 of *LNCS*, pages 57–64. Springer, 2012.
[13] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, M. Schwartzbach, and T. Toft. Secure multiparty computation goes live. In *Financial Cryptography and Data Security (FC)*, volume 5628 of *LNCS*, pages 325–343. Springer, 2009.
[14] F. Brasser, U. MÃijller, A. Dmitrienko, K. Kostiainen, S. Capkun, and A. Sadeghi. Software Grand Exposure: SGX Cache Attacks Are Practical. *CoRR*, abs/1702.07521, 2017.
[15] M. Burkhart, M. Strasser, D. Many, and X. Dimitropoulos. SEPIA: Privacy-preserving Aggregation of Multi-domain Network Events and Statistics. In *USENIX Security*, 2010.
[16] N. Büscher and S. Katzenbeisser. Faster Secure Computation through Automatic Parallelization. In *USENIX Security*, 2015.
[17] K. Butler, T. R. Farley, P. McDaniel, and J. Rexford. A survey of BGP security issues and solutions. *Proceedings of the IEEE*, 98(1):100–122, 2010.
[18] M. Canini, V. Jovanović, D. Venzano, G. Kumar, D. Novaković, and D. Kostić. Checking for Insidious Faults in Deployed Federated and Heterogeneous Distributed Systems. Technical Report 164475, EPFL, 2011.
[19] D. R. Choffnes and F. E. Bustamante. On the Effectiveness of Measurement Reuse for Performance-Based Detouring. In *IEEE Conference on Computer Communications (INFOCOM)*, 2009.
[20] Bgp best path selection algorithm, 2017. http://bit.ly/2slSBhV.
[21] L. Cittadini, G. Di Battista, T. Erlebach, M. Patrignani, and M. Rimondini. Assigning AS relationships to satisfy the Gao-Rexford conditions. In *International Conference on Network Protocols (ICNP)*, 2010.
[22] V. Costan and S. Devadas. Intel SGX explained. Cryptology ePrint Archive, Report 2016/086, 2016. http://ia.cr/2016/086.
[23] D. Croce, E. Leonardi, and M. Mellia. Large-Scale Available Bandwidth Measurements: Interference in Current Techniques. *IEEE Transactions on Network and Service Management*, 8(4):361–374, 2011.
[24] D. Croce, M. Mellia, and E. Leonardi. The Quest for Bandwidth Estimation Techniques for Large-scale Distributed Systems. *SIGMETRICS Perform. Eval. Rev.*, 37(3), Jan. 2010.
[25] Deutscher commercial internet exchange infrastructure, 2013. https://apollon.de-cix.net/news/blog-post/2013/07/26/de-cix-apollons-current-topology/.
[26] Project endeavour, Jan. 2015. https://www.de-cix.net/en/about-de-cix/research-and-development/endeavour.
[27] An IXP Route Server Test Framework, 2016. https://www.de-cix.net/_Resources/Persistent/fba89bc19381b6784df99d2a78d4a11ebb7583c2/DE-CIX-route-server-testframework.pdf.
[28] Deutscher Commercial Internet Exchange, 2017. https://www.de-cix.net/.
[29] D. Demmler, T. Schneider, and M. Zohner. ABY – a framework for efficient mixed-protocol secure two-party computation. In *The Network and Distributed System Security Symposium (NDSS)*, 2015.
[30] X. Dimitropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, K. Claffy, and G. Riley. AS relationships: Inference and validation. *Computer Communication Review*, 37(1):29–40, Jan. 2007.
[31] L. Gao. On Inferring Autonomous System Relationships in the Internet. *IEEE/ACM Transactions on Networking*, 9(6):733–745, Dec. 2001.
[32] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2013.
[33] C. Gentry. Fully homomorphic encryption using ideal lattices. In *ACM Symposium on Theory of Computing (STOC)*. ACM, 2009.
[34] C. Gentry, S. Halevi, and N. P. Smart. Homomorphic evaluation of the AES circuit. In *CRYPTO*, volume 7417 of *LNCS*, pages 850–867. Springer, 2012.
[35] P. Gill, M. Schapira, and S. Goldberg. Let the Market Drive Deployment: a Strategy for Transitioning to BGP Security. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2011.
[36] P. Gill, M. Schapira, and S. Goldberg. A Survey of Interdomain Routing Policies. *Computer Communication Review*, 2014.
[37] O. Goldreich. *The Foundations of Cryptography - volume 2, Basic Applications*. Cambridge University Press, 2004.

[38] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *ACM Symposium on Theory of Computing (STOC)*, 1987.

[39] S. Goldwasser, Y. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable garbled circuits and succinct functional encryption. In *ACM Symposium on Theory of Computing (STOC)*. ACM, 2013.

[40] Cloud Native Networking, 2017. Amin Vahdat's keynote at Open Networking Summit. Available at http://bit.ly/2qIZigQ.

[41] A. Gupta, R. MacDavid, R. Birkner, M. Canini, N. Feamster, J. Rexford, and L. Vanbever. An industrial-scale software defined internet exchange point. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2016.

[42] A. Gupta, L. Vanbever, M. Shahbaz, S. P. Donovan, B. Schlinker, N. Feamster, J. Rexford, S. Shenker, R. Clark, and E. Katz-Bassett. SDX: A software defined internet exchange. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2014.

[43] D. Gupta, B. Mood, J. Feigenbaum, K. R. B. Butler, and P. Traynor. Using intel software guard extensions for efficient two-party secure function evaluation. In *Financial Cryptography and Data Security - FC International Workshops, BITCOIN, VOTING, and WAHC*, 2016.

[44] D. Gupta, A. Segal, A. Panda, G. Segev, M. Schapira, J. Feigenbaum, J. Rexford, and S. Shenker. A new approach to interdomain routing based on secure multi-party computation. In *ACM Workshop on Hot Topics in Networks (HotNets*, 2012.

[45] W. Henecka and M. Roughan. STRIP: privacy-preserving vector-based routing. In *International Conference on Network Protocols (ICNP)*, 2013.

[46] A. Holzer, M. Franz, S. Katzenbeisser, and H. Veith. Secure two-party computations in ANSI C. In *ACM Conference on Computer and Communications Security (CCS)*, pages 772–783. ACM, 2012.

[47] Y. Huang, D. Evans, J. Katz, and L. Malka. Faster secure two-party computation using garbled circuits. In *USENIX Security*, pages 539–554, 2011.

[48] G. Huston, M. Rossi, and G. Armitage. Securing BGP – a literature survey. *Communications Surveys Tutorials, IEEE*, 13(2):199–222, 2011.

[49] Advertisement of Multiple Paths in BGP, Oct. 2014. https://tools.ietf.org/html/draft-ietf-idr-add-paths-10.

[50] Making Route Servers Aware of Data Link Failures at IXPs, 2017. https://tools.ietf.org/html/draft-ietf-idr-rs-bfd-02.

[51] Interxion colocation services, 2017. http://www.interxion.com/.

[52] M. Jain and C. Dovrolis. Path Selection Using Available Bandwidth Estimation in Overlay-Based Video Streaming. *Computer Networks*, 52(12):2411–2418, 2008.

[53] P. Jain, S. J. Desai, M. Shih, T. Kim, S. M. Kim, J. Lee, C. Choi, Y. Shin, B. B. Kang, and D. Han. OpenSGX: An Open Platform for SGX Research. In *The Network and Distributed System Security Symposium (NDSS)*, 2016.

[54] U. Javed, I. Cunha, D. R. Choffnes, E. Katz-Bassett, T. E. Anderson, and A. Krishnamurthy. PoiRoot: investigating the root cause of interdomain path changes. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2013.

[55] M. Keller. The oblivious machine – or: How to put the C into MPC. *IACR Cryptology ePrint Archive*, page 467, 2015. http://ia.cr/2015/467.

[56] S. Kim, Y. Shin, J. Ha, T. Kim, and D. Han. A first step towards leveraging commodity trusted execution environments for network applications. In *ACM Workshop on Hot Topics in Networks (HotNets*, 2015.

[57] S. M. Kim, J. Han, J. Ha, T. Kim, and D. Han. Enhancing Security and Privacy of Tor's Ecosystem by Using Trusted Execution Environments. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2017.

[58] R. Kloeti, M. Rost, P. Georgopoulos, B. Ager, S. Schmid, and D. X. Stitching inter-domain paths over IXPs. In *ACM Sigcomm Symposium on SDN Research (SOSR)*, 2016.

[59] P. Koeberl, V. Phegade, A. Rajan, T. Schneider, S. Schulz, and M. Zhdanova. Time to rethink: Trust brokerage using trusted execution environments. In *Trust and Trustworthy Computing (TRUST)*, volume 9229 of *LNCS*, pages 181–190. Springer, 2015.

[60] V. Kolesnikov, A.-R. Sadeghi, and T. Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. In *CANS*, volume 5888 of *LNCS*, pages 1–20. Springer, 2009.

[61] B. Kreuter, B. Mood, A. Shelat, and K. Butler. PCF: a portable circuit format for scalable two-party secure computation. In *USENIX Security*, 2013.

[62] T. Lee, C. Pappas, D. Barrera, P. Szalachowski, and A. Perrig. Source Accountability with Domain-brokered Privacy. In *International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2016.

[63] LightReading. Pica8 powers french TOUIX SDN-driven internet exchange, June 2015. http://www.lightreading.com/white-box/white-box-systems/pica8-powers-french-touix-sdn-driven-internet-exchange/d/d-id/716667.

[64] C. Liu, X. S. Wang, K. Nayak, Y. Huang, and E. Shi. ObliVM: A programming framework for secure computation. In *S&P*, 2015.

[65] M., R. Di Lallo, G. Lospoto, H. Mostafaei, M. Rimondini, and G. Di Battista. PrIXP: Preserving the Privacy of Routing Policies at Internet eXchange Points. In *IFIP/IEEE International Symposium on Integrated Network Management, IM*, 2017.

[66] Internet Routing Privacy Survey, 2017. http://bit.ly/2rjT7Nj.

[67] M. Chiesa, D. Demmler, M. Canini, M. Schapira, T. Schneider. Securing Internet eXchange Points Against Curious onlooKers, Jan. 2017. http://bit.ly/sixpack-tech-rep.

[68] S. Machiraju and R. H. Katz. Reconciling cooperation with confidentiality in multi-provider distributed systems. Technical Report UCB/CSD-04-1345, EECS Department, University of California, Berkeley, Aug 2004.

[69] S. Machiraju and R. H. Katz. Verifying global invariants in multi-provider distributed systems. In *ACM Workshop on Hot Topics in Networks (HotNets*, 2004.

[70] H. V. Madhyastha, E. Katz-Bassett, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane Nano: Path Prediction for Peer-to-peer Applications. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2009.

[71] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay - secure two-party computation system. In *USENIX Security*, 2004.

[72] Z. M. Mao, R. Bush, T. Griffin, and M. Roughan. BGP Beacons. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2003.

[73] J. B. Nielsen, P. S. Nordholt, C. Orlandi, and S. S. Burra. A new approach to practical active-secure two-party computation. In *CRYPTO*, volume 7417 of *LNCS*, pages 681–700. Springer, 2012.

[74] P. Papageorge, J. McCann, and M. Hicks. Passive Aggressive Measurement with MGRP. *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 39(4), 2009.

[75] A. K. Paul, A. Tachibana, and T. Hasegawa. An Enhanced Available Bandwidth Estimation Technique for an End-to-End Network Path. *IEEE Transactions on Network and Service Management*, 13(4):768–781, 2016.

[76] P. Richter, G. Smaragdakis, A. Feldmann, N. Chatzis, J. Boettger, and W. Willinger. Peering at peerings: On the role of IXP route servers. In *Internet Measurement Conference (IMC)*, 2014.

[77] M. Roughan, W. Willinger, O. Maennel, D. Perouli, and R. Bush. 10 Lessons from 10 Years of Measuring and Modeling the Internet's Autonomous Systems. *IEEE Journal on Selected Areas in Communications*, 29(9):1810–1821, 2011.

[78] M. Roughan and Y. Zhang. Privacy-preserving performance measurements. In *Workshop on Mining Network Data (MineNet)*, pages 329–334. ACM, 2006.

[79] Performance Evaluation of BIRD and GoBGP, 2014. https://www.euro-ix.net/m/uploads/2016/04/24/EuroIX_GoBGP_20160419.pdf.

[80] T. Schneider and M. Zohner. GMW vs. Yao? efficient secure two-party computation with low depth circuits. In *Financial Cryptography and Data Security (FC)*, volume 7859 of *LNCS*, pages 275–292. Springer, 2013.

[81] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy. BlindBox: Deep packet inspection over encrypted traffic. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2015.

[82] R. Sherwood, A. Bender, and N. Spring. Discarte: a disjunctive internet cartographer. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2008.

[83] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, A. Kanagala, J. Provost, J. Simmons, E. Tanda, J. Wanderer, U. Hölzle, S. Stuart, and A. Vahdat. Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network. *Computer Communication Review*, 45(5):183–197, 2015.

[84] J. Stringer, Q. Fu, C. Lorier, and C. E. Rothenberg. Cardigan: Deploying a distributed routing fabric. In *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)*, 2013.

[85] S. Tamrakar, J. Liu, A. Paverd, J. Ekberg, B. Pinkas, and N. Asokan. The circle game: Scalable private membership test using trusted hardware. *CoRR*, abs/1606.01655, 2016.

[86] S. Tao and R. Guérin. On-line Estimation of Internet Path Performance: An Application Perspective. In *IEEE Conference on Computer Communications (INFOCOM)*, 2004.

[87] F. Wang and L. Gao. On inferring and characterizing internet routing policies. In *Internet Measurement Conference (IMC)*, 2003.

[88] H. Wang, K. S. Lee, E. Li, C. L. Lim, A. Tang, and H. Weatherspoon. Timing is Everything: Accurate, Minimum Overhead, Available Bandwidth Estimation in High-speed Wired Networks. In *Internet Measurement Conference (IMC)*, 2014.

[89] X. Wang and M. K. Reiter. Mitigating Bandwidth-Exhaustion Attacks Using Congestion Puzzles. In *ACM Conference on Computer and Communications Security (CCS)*, 2004.

[90] C. Xing, L. Yang, and M. Chen. Estimating Internet Path Properties for Distributed Applications. In *WiCOM*, 2009.

[91] A. C. Yao. How to generate and exchange secrets. In *Annual Symposium on Foundations of Computer Science (FOCS)*, 1986.

[92] M. Zhao, W. Zhou, A. J. T. Gurney, A. Haeberlen, M. Sherr, and B. T. Loo. Private and verifiable interdomain routing decisions. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2012.

[93] M. Zhao, W. Zhou, A. J. T. Gurney, A. Haeberlen, M. Sherr, and B. T. Loo. Private and verifiable interdomain routing decisions. *IEEE/ACM Transactions on Networking*, 24(2):1011–1024, 2016.

Table 2: SMPC micro benchmarks: Circuit properties and runtimes of both building-blocks from §5 for 750 members, and a given number of input keys, where one input is the dummy key and the remaining inputs are route keys.

		Circuit			Benchmarks			
					Setup Phase		Online Phase	
Approach	#Inputs	Total ANDs	Vector ANDs	AND Depth	Runtime [ms]	Comm. [KiB]	Runtime [ms]	Comm. [KiB]
Export-All	2	102,000	750	1	1.7 (±8 %)	28	0.6 (±17 %)	25
	4	306,000	2,250	1	3.6 (±5 %)	76	1.3 (±11 %)	75
	8	714,000	5,250	1	7.2 (±3 %)	172	2.4 (±7 %)	176
	16	1,530,000	11,250	1	14.2 (±2 %)	360	4.4 (±5 %)	377
	32	3,162,000	23,250	1	24.0 (±3 %)	732	8.3 (±4 %)	778
Select-Best ($\ell = 4$ bit)	2	114,000	7,500	5	6.2 (±5 %)	248	1.9 (±8 %)	30
	4	342,000	22,500	9	14.9 (±3 %)	720	4.6 (±4 %)	89
	8	798,000	52,500	13	24.8 (±3 %)	1,652	9.4 (±5 %)	208
	16	1,710,000	112,500	17	44.4 (±5 %)	3,540	17.7 (±2 %)	445
	32	3,534,000	232,500	21	77.6 (±8 %)	7,293	34.0 (±1 %)	920
Select-Best ($\ell = 8$ bit)	2	128,250	15,750	6	10.6 (±4 %)	504	2.3 (±5 %)	35
	4	384,750	47,250	11	19.8 (±4 %)	1,492	5.3 (±4 %)	106
	8	897,750	110,250	16	34.6 (±6 %)	3,469	10.3 (±3 %)	247
	16	1,923,750	236,250	21	63.7 (±9 %)	7,409	19.3 (±3 %)	528
	32	3,975,750	488,250	26	122.4 (±8 %)	15,286	35.9 (±1 %)	1,091

A DETAILED PERFORMANCE EVALUATION

We provide comprehensive performance numbers for the evaluation that is described in §7.

The number of total AND gates reported in Tab. 2 is the number of AND gates we would have to evaluate if we would use Yao's garbled circuits protocol for our SMPC implementation. By using vector gates, that precompute multi-bit AND gates for the cost of 1-bit AND gates and that are only possible in the GMW protocol, we can save more than two orders of magnitude for the runtime of the setup phase.

B OPTIMIZATIONS

We believe that the running time of SIXPACK can be drastically reduced through the following two improvements:

First, the current implementation of the SMPC does not allow to separate the setup and online phases. For example, the runtime of the SMPC part for the Select-Best approach, when processing 32 inputs and using 4 bits for representing preferences, currently takes a total time of 158.0 ms, while the online phase only amounts to 35.9 ms of that. Since for all our results the setup phase accounts for at least 77% of the total SMPC processing time, we believe that the per-prefix processing time can be improved significantly as soon as our implementation becomes capable of precomputing the setup phase. We stress that this is possible in theory and the current situation is merely a limitation of our implementation.

Third, our Python code could be rewritten in a more performance-oriented programming language such as C/C++, thus gaining an additional decrease in runtimes.

C DEPLOYMENT

SIXPACK can easily and incrementally be deployed at IXPs since it needs not replace the traditional RS service, i.e., the two can coexist. Each member can independently decide whether to share its routes in a privacy-preserving manner or not. We believe that the desire to preserve peering relationships private can incentivize adoption.

SIXPACK only requires small modifications at both the IXP side and IXP member side (i.e., route servers and BGP border routers). Each member could run SIXPACK as a BGP proxy that receives the BGP update messages intended for the RS from its BGP border router and performs the SMPC input sharing process. Similarly, the BGP proxy will receive the output of the SMPC from the two SMPC-enabled route servers and translate it into BGP updates that will, in turn, be forwarded to the member's BGP border router.

We believe that bringing innovation to today's networks is no longer as prohibitive a task as it has been in the past. The rise of programmable networks with Software-Defined-Networking (SDN) is boosting innovation in different environment such as data centers [83] and, more recently, IXP networks [3, 26, 42, 63, 84]. Deploying SIXPACK in a so-called Software-Defined-eXchange (SDX) could provide the opportunity for using custom interfaces and avoid the need for compatibility with BGP. Advanced route dispatch services, based for instance on port congestion metrics, should obliviously be designed to avoid route oscillations (see Appendix D).

D ROUTE OSCILLATIONS

As with any algorithm that dynamically adapts the routing paths based on the network performance state, there exists a risk of not converging to a stable routing configuration. For instance, when an IXP ranks routes based on congestion levels, traffic might move back and forth between IXP ports as these become more or less congested and so are assigned new priorities. We propose addressing this in Select-Best by ensuring that when a member is offered a new route this is either because (1) its old route was withdrawn, or (2) a new route that *the member's local preference* ranks higher than the old presents itself. We discuss below two different ways of accomplishing this: one on the member side and one on the RS side.

Importantly, in both schemes, members' local preferences over routes should be given higher preference than the IXP's preference. When a member receives its best route from the SMPC, it could temporarily (slightly) increase the priority of that route to ensure that it will be selected in successive iterations, unless a better route is available, and communicate its new preferences to the RSes. Alternatively, this could also be realized within the SMPC itself, thus

minimizing communication between members and RSes, though at the price of higher circuit complexity.

E SECURITY AND PRIVACY

The overall security and privacy of SIXPACK stems from the security of the used symmetric encryption and the proven security and privacy of the GMW protocol (cf. [37, 38] for proofs).

We employ a symmetric encryption scheme to encrypt the routes that are transferred from sending to receiving members. In our implementation we use AES with a key size of 128 bit in counter-mode (CTR). We are not aware of any significant attacks on AES in this mode of operation and thus believe it to be a viable choice for our purpose.

We ensure that the symmetric keys that are required to decrypt a route are only provided to those members who are explicitly allowed to receive them. This is achieved by relying on the GMW protocol that provably guarantees correctness, security and privacy. The correctness property of GMW together with the correctness of our circuits we evaluate with it ensure that only those members who are explicitly allowed by the export policy of a sending member will receive the correct key to decrypt an encrypted route. All other members receive a dummy key, that will not successfully decrypt the route. We describe the circuits that implement this behavior in §5.4 and §5.5.

The privacy property of GMW ensures that the IXP, who carries out the operations on the routes and secret-shared export policies cannot gain access to this data unless he breaks the non-collusion assumption. This is guaranteed by using information theoretic XOR-based secret sharing that masks plaintext inputs with random data and splits it between the computational parties. These parties evaluate our circuits gate by gate using the GMW protocol, while maintaining the invariant that the values on each wire in the circuit is secret-shared among the parties. The circuits' outputs are sent to the receiving members who are able to reconstruct the plaintext output values. We summarize our observations above in the following theorem.

Theorem (Correctness, Security, and Privacy): The EXPORT-ALL protocol (Fig. 6) and the SELECT-BEST protocol (Fig. 9) correctly, securely, and privately compute the respective functionality described in §E in the presence of a semi-honest adversary, corrupting either SMPC_1 or SMPC_2, but not both.

F PROTOCOL DESCRIPTIONS

We provide formal protocol descriptions of our protocols in this section. We describe the EXPORT-ALL protocol in Fig. 14. For the SELECT-BEST approach we detail how new local preferences are shared in Fig. 15 and the SELECT-BEST protocol in Fig. 16.

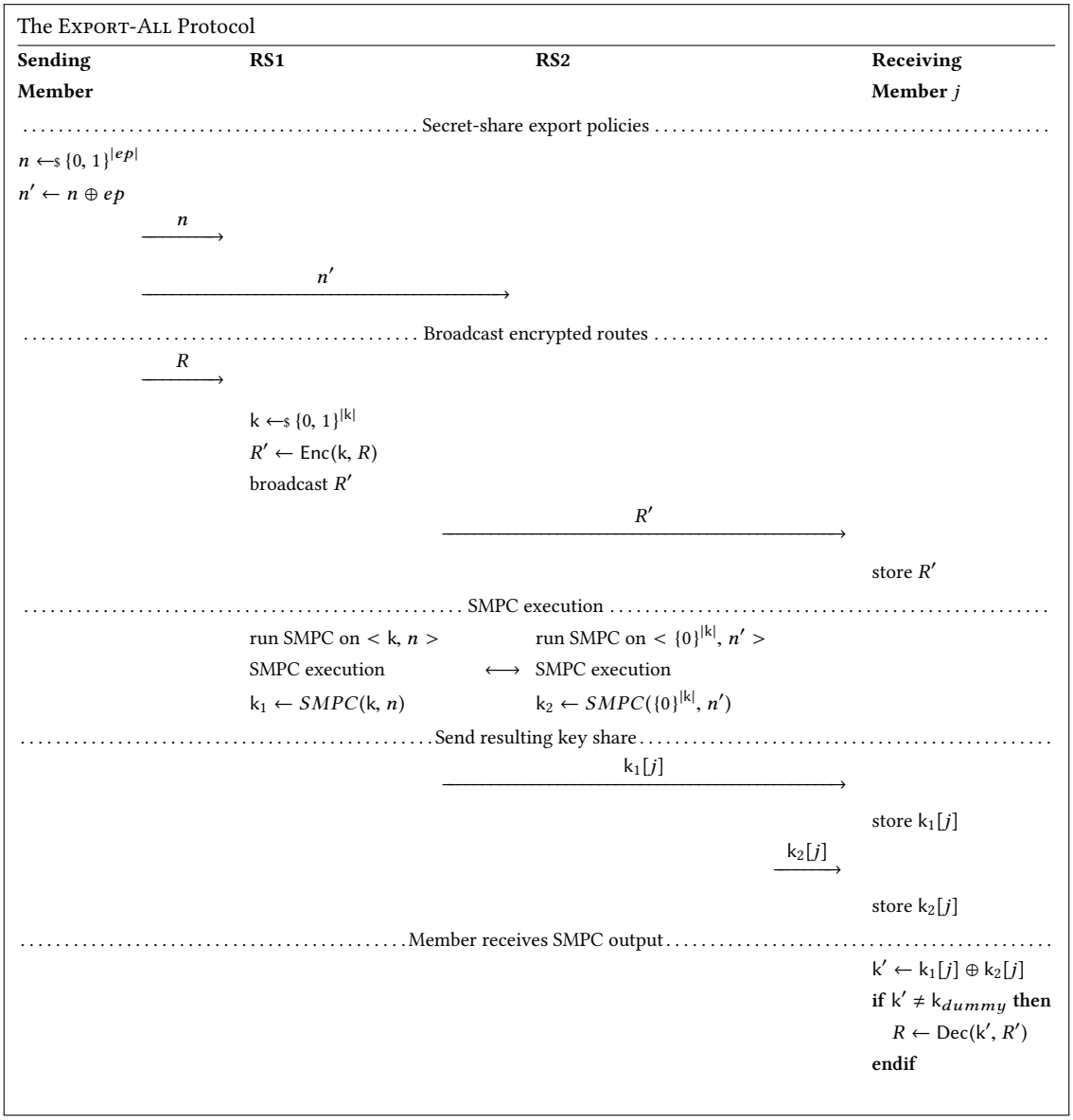


Figure 14: The Export-All Protocol. Inputs: R: a BGP route stripped off its export policy ep; $|ep|$: the length in bits of ep; $|k|$: the bit length of key k

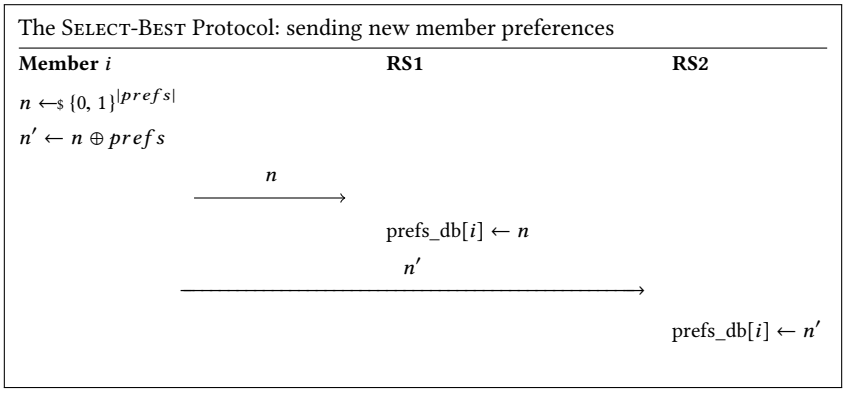


Figure 15: Preference secret-sharing in the SELECT-BEST protocol.

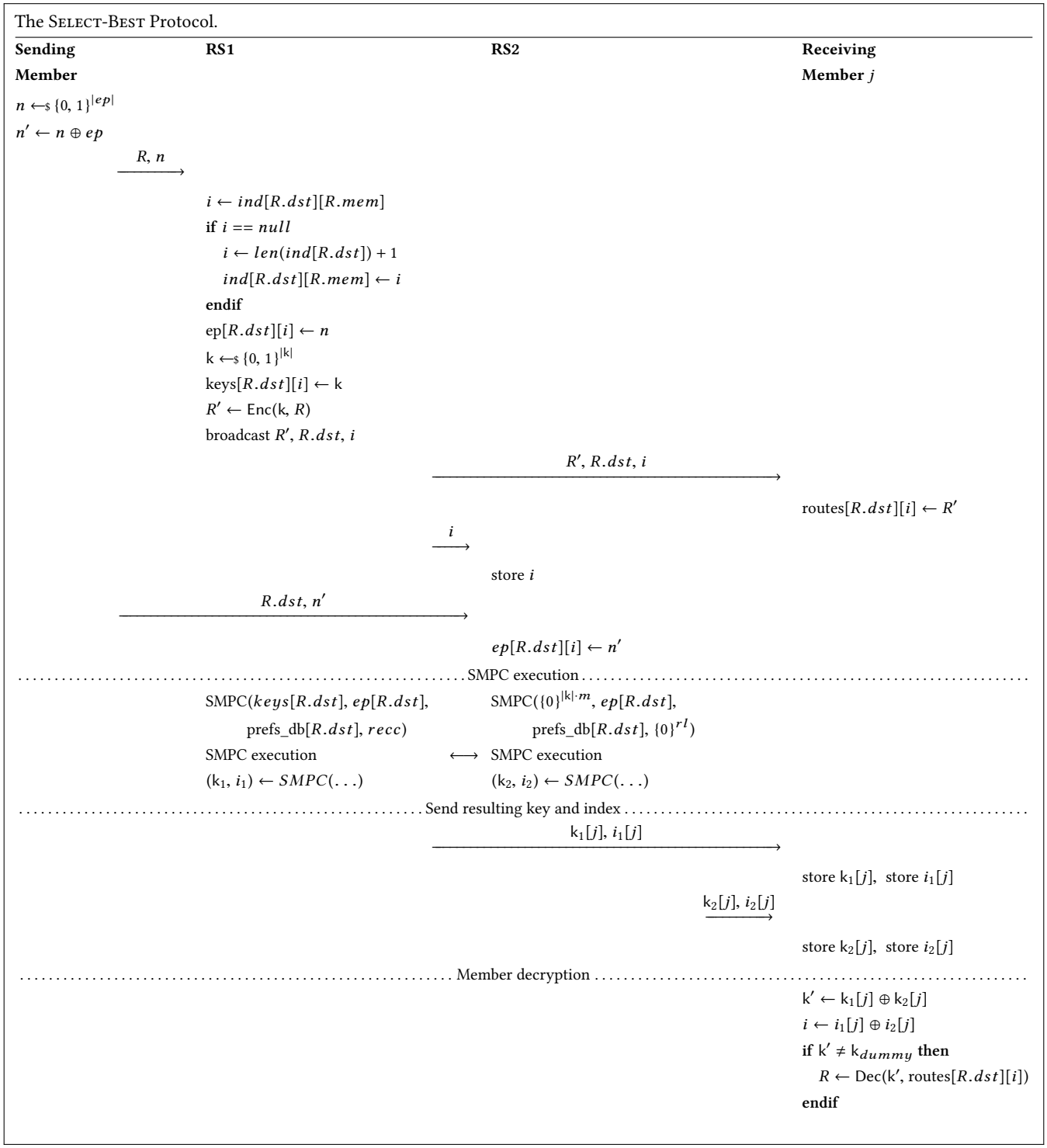


Figure 16: The SELECT-BEST protocol. Inputs: R: a BGP route stripped off its export policy ep; $|ep|$: the length in bits of ep; |k|: the bit length of key k; $recc(R)$: route recommendation at the IXP; m is the number of members.