A New Approach to Interdomain Routing Based on Secure Multi-Party Computation

Debayan Gupta*, Aaron Segal*, Aurojit Panda†, Gil Segev‡,
Michael Schapira§, Joan Feigenbaum*, Jenifer Rexford††, Scott Shenker†**
*Yale, †UC Berkeley, ‡MSR, §Hebrew University, ††Princeton, **ICSI

Interdomain routing involves coordination among mutually distrustful parties, leading to the requirements that BGP provide policy autonomy, flexibility, and privacy. BGP provides these properties via the distributed execution of policy-based decisions during the iterative route computation process. This approach has poor convergence properties, makes planning and failover difficult, and is extremely difficult to change. To rectify these and other problems, we propose a radically different approach to interdomain-route computation, based on secure multi-party computation (SMPC). *Our approach provides stronger privacy guarantees than BGP and enables the deployment of new policy paradigms. We report on an initial exploration of this idea and outline future directions for research.*

1 Introduction

The Internet architecture was born with the invention of the Internet Protocol (IP) whose purpose was to connect the disparate networking technologies that were then being developed by different groups. When this unifying architecture left the sheltered world of research and entered the commercial realm, it had to deal with a new and unforeseen task: interdomain routing. Moreover, this task did not have a clear and globally defined objective (such as finding the shortest path) but instead involved the individual (and often conflicting) commercial interests of each domain. Specifically, domains needed *flexibility* in the routes they use and in the routes they let other domains use, *autonomy* to determine these routing policies for themselves without consulting other domains, and the ability to keep their routing policies *private*.

At the time of the Internet's transition into a commercial infrastructure, no existing routing technologies offered the necessary policy flexibility, autonomy, and privacy. After a period of intensive research into the problem, the Border Gateway Protocol (BGP) was invented to address these requirements. BGP provides flexibility by allowing domains to choose freely among available routes (with the path-vector formulation preventing loops) and provides autonomy and (some) privacy by having both route-import and route-export choices exercised in a completely local manner; that is, routing policies are not announced globally but rather acted upon locally in each step of BGP's distributed routing protocol. Of course, the resulting routing decisions are visible to neighbors, which implicitly reveals some information, but the general policy is never announced.

Even though BGP has so far provided a durable foundation for the Internet, it is far from an ideal solution. First, BGP is notorious for taking a long time to converge after a topology or policy change, leading to significant disruptions for interactive applications when packets are lost, stuck in loops, or delivered out of order [8]. We are not listing policy oscillations as one of the problematic aspects of BGP because (i) it isn't clear how important this problem is in practice, and (ii) policy oscillations arise almost directly from the policy requirements, as opposed to being artifacts of their implementation. We believe that our SMPC-based approach to interdomain routing can enable alternative policy frameworks, and we plan to explore this aspect of the problem in future work.

The authors have been variously supported in part by the following grants: NSF CNS-1016875, NSF CNS 1117161, NSF CNS 1015459, and ONR N000141210478 Two of the authors were supported in part by the Intelligence Advanced Research Project Activity (IARPA) via Department of Interior National Business Center (DoI/NBC) contract number D11PC20197. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation hereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

*Yale University, New Haven, CT
{debayan.gupta,aaron.segal,joan.feigenbaum}@yale.edu
†UC Berkeley, Berkeley, CA
{shenker,apanda}@cs.berkeley.edu
‡Microsoft Research Silicon Valley, Mountain View, CA
gil.segev@microsoft.com
§Hebrew University, Jerusalem, Israel
schapiram@huji.ac.il
††Princeton University, Princeton, NJ
jrex@cs.princeton.edu
**ICSI, Berkeley, CA

Second, network operators cannot assess *a priori* the impact of a topology or policy change. This makes network planning difficult as even planned changes, such as might happen during scheduled maintenance, could result in long outages, protocol oscillation, or poor performance. Third, any significant enhancement to BGP (e.g., better security, multipath routing, or multicast communication) faces nearly insurmountable barriers, due to the need for global adoption. Moreover, the prevailing policy model is limiting; while domains control their own import and export policies, they have no control over the upstream portion of the path (as they do in [10]). All three of these problems are deeply rooted in BGP's local execution model.

BGP's many problems are well known, and there have been many proposals to fix various isolated aspects of BGP. These proposals improve the convergence and/or consistency properties of BGP [7], alter the degree of policy privacy and autonomy in return for more stable dynamics [14], give some route choice to the ends rather than having routes be chosen solely by the domains themselves [5, 15], and introduce enforcement mechanisms to ensure that policy choices are indeed respected [10, 18]. However, none of these interdomain-routing proposals alters the basic fact that currently the only way to match BGP's level of policy flexibility, autonomy, and privacy is to leave BGP's distributed execution model largely intact, and that the problems cited above are largely inherent in this approach.

In this paper, we describe a radically different approach that leverages two recent trends—the vast computational resources in modern data centers and advances in Secure Multi-Party Computation (SMPC)—to address the three problematic aspects of BGP while retaining flexibility and autonomy and improving privacy. We propose that, rather than using local execution to compute interdomain routes, a small number k of computational clusters compute interdomain routes in a manner that both allows domains (at least) the same degrees of flexibility and autonomy as with BGP and is provably privacy-preserving. Privacy is achieved through cryptographically based SMPC protocols rather than through the local execution of routing policy.

In the next section, we give a brief introduction to SMPC, and explain how we apply it to interdomain routing. Here, we describe some of the benefits of this approach and how it "fixes" the BGP shortcomings described above.

- **Faster convergence.** Working together, the clusters compute the routing *outcome* for all domains, rather than have each domain perform a long path-exploration process. This leads to faster convergence, especially if the SMPC algorithm completes quickly or can run in advance. In addition, domains no longer set local parameters (e.g., MRAI timers, route-flap damping thresholds, *etc.*) that have unpredictable effects on global convergence. Convergence time becomes a question of the network latencies to/from the clusters and the computation time in the clusters, the latter of which can be improved over time through better algorithms and machines.

- **'What-if' analysis.** The clusters can perform 'what-if' analysis to precompute paths after a failure and identify network events that would partition parts of the Internet. For example, undersea cable cuts often cause serious disruptions in Internet connectivity. The clusters could notify domains about potential risks to support advanced planning to install additional capacity or negotiate backup relationships.

- **More efficient routers.** With most computation moved to the k clusters, IP routers do not need to store a large number of BGP-learned routes and run the BGP decision process. Routers would (at most) need to select among multiple "best" egress points to reach a destination, based on intradomain-routing information. In addition, future enhancements to interdomain routing would only impact the clusters and would not impose a computation and storage burden on legacy routers.

- **Better security.** In addition to offering provable privacy, our approach enables better security generally. The clusters could upgrade to new security solutions, such as comparing routing information against historical data, Internet routing registries, or Route Origin Authorizations (ROAs). Upgrading a small number of clusters is much easier than convincing every domain to upgrade its router software (and perhaps even hardware) and move to new operational practices.

- **Freedom to innovate.** Similarly, the clusters can easily move to new computational models, such as computing multiple paths for a destination prefix or computing multicast routes. Since the policy model is no longer tied to the distribution model, these clusters could support routing policies that go beyond today's "route-maps" that manipulate and rank locally-learned routes. For example, business relationships between domains could move beyond bilateral arrangements to allow a group of networks to cooperate in offering a value-added service.

The preceding words sketch out a vision. This paper reports on an initial step toward this vision – a full but slow implementation of this approach for a special case of routing policies. Our hope is that these initial results are sufficient to illustrate the benefits and the challenges inherent in our vision.

2 SMPC-Based Interdomain Routing

In this section, we give some basic background on SMPC and then explain how we use it to compute routes.

2.1 SMPC Background

The study of SMPC was initiated in the seminal papers of Yao [16, 17] and has been pursued intensively for more than 30 years by the cryptographic-theory community. SMPC addresses scenarios in which n parties $P_1, \ldots, P_n$ hold private inputs $x_1, \ldots, x_n$ and wish to compute $y = f(x_1, \ldots, x_n)$ in such a way that all parties learn y but no P_i learns anything about x_j, $i \neq j$, except what is logically implied by the result y and the particular input x_i that he already knew. Moreover, they wish to do so by executing a protocol in which they all play equivalent roles; that is, they don't want simply to send the x_i's to one trusted party that can compute y and send it to all of them. Natural applications include voting, survey computation, and set operations. One of the central results of cryptographic theory is that such privacy-preserving protocols can be obtained for any function f, provided one is willing to assume that certain cryptographic primitives are secure or that some fraction of the P_i's do not cheat (i.e., that they follow the protocol scrupulously). An in-depth survey of SMPC and its application to data mining can be found in [9].

The following structure is common to many SMPC protocols. In the first round, each P_i splits its input x_i into *shares*, using a *secret-sharing scheme* in the sense of [13], and sends one share to each P_j; the privacy-preserving properties of secret sharing guarantee that the shares do not reveal x_i to the other parties (or even to coalitions of other parties, provided that the coalitions are not too large). The parties then execute an interactive protocol to compute *shares* of y; the multi-round protocol is designed to ensure that the shares of intermediate results computed in each round also do not reveal x_i. In the last round, the parties broadcast their shares of y so that all of them can reconstruct the result. Alternatively, they may send the shares of y to an outside entity or to a subset of the P_j's if none (or only a subset) of the P_j's is supposed to learn the result. The choice of secret-sharing scheme is determined by the size of the largest coalition of cheating parties that the protocol must be able to thwart and by the "adversarial model," i.e., the capabilities and resources available to the cheaters.

Because secret-sharing-based SMPC is very common (and for ease of exposition), we will refer at some points to parties' "sharing" or "splitting" their inputs. Note, however, that there are SMPC protocols that use other techniques (such as threshold public-key cryptography) to encode inputs and execute multi-round computations in a privacy-preserving manner.

One crucial limitation of the powerful and general SMPC protocols in the literature is their highly interactive nature; for example, they require multiple rounds of communication and computation and may require every pair of parties to exchange messages in every round – clearly impractical for large n. This leads naturally to an architecture, which we adopt here, in which the original parties $P_1, \ldots, P_n$ (in our setting, the domains) play the role of *input providers* and a disjoint set of parties $S_1, \ldots, S_k$ (in our setting, the computational clusters) play the role of *secure-computation servers*. The domains share their inputs with the computational clusters; once the clusters have their shares of the route that a particular domain is assigned, they send the shares *to that domain only*. This hybrid form of SMPC is often referred to as "secure outsourcing." It has been applied, for example, to the problem of privacy-preserving survey computation [3]. To the best of our knowledge, ours is the first attempt to apply it to interdomain routing.

One can implement SMPC using a general-purpose platform, such as FairplayMP, SEPIA, VIFF, or Tasty, or develop special-purpose implementations of SMPC that can be more efficient, as in [6]. In this paper we focus on the former, using the SEPIA platform to implement our approach to a special case of interdomain policies (next-hop policies [12]). However, this implementation has not been optimized, nor has it been run in a fully parallel mode; so the timing results we present later are merely an initial indication of the scheme's viability.

2.2 Application to Interdomain Routing

We propose a secure-outsourcing architecture in which the input providers are the ASes, and each of the k secure-computation servers is operated by a different organization with many machines at its disposal (e.g., companies such as Google, Amazon, and Microsoft that run very large data centers). We believe that a plausible value for the number of servers is $k = 3$. Section 3 describes a SEPIA implementation that works for any $k \geq 3$. It is secure against coalitions of fewer than $k/2$ "honest but curious" cheating servers. In the honest-but-curious adversarial model, all parties follow the protocol specification but may later collude by pooling information they obtained during an execution.

In our design, as in today's BGP, routes to every destination IP prefix are computed independently; thus, we describe the computation process for a fixed destination prefix. In SMPC terms, the private input x_i of each AS i is its routing policy, which consists of the following two components:

1. **Ranking of routes:** an order over all of i's possible (loop-free) AS-level routes to the destination.
2. **Export policy:** for every neighboring AS j, a specification of which routes to the destination AS i is willing to make available ("export") to j.

Each i splits its policy x_i into k shares and sends one to each of the k computational clusters. The clusters now engage in an SMPC to compute the BGP-routing outcome determined by these inputs.[1] At the end of this computation, each of the k clusters holds, for every AS i, a share of i's route in the computed outcome. Each cluster then sends to

[1]We make the assumption that, for every combination of ASes' routing policies, the routing outcome (stable state) that BGP computes is uniquely defined. The Gao-Rexford conditions [4], which capture many of today's routing policies, indeed induce a unique routing outcome.

each i that cluster's share of i's computed route. AS i can then combine these k shares to learn its own route to the destination.

Such an SMPC-based interdomain-routing protocol provides two strong guarantees: (a) each domain learns its own routes but learns nothing about the others' routes except for the information logically implied by its own route; (b) the clusters do not learn anything about the domains' routing policies or about the resulting routes themselves unless at least half of them collude and pool their shares.

3 Proof-of-Concept

We now describe our initial attempt to test out these ideas.

3.1 Simplifications and Structure

Rather than attack BGP in its full policy generality, we considered a special case that simplifies the problem along several dimensions. In terms of policy, we assume that domains use only next-hop policies [12]; that is, their import and export policies depend only on the first-hop domain of each route. Moreover, we assume that the Gao-Rexford [4] constraints hold, so that no policy oscillations result from such policies. With respect to privacy, we distinguish between information about the network (e.g., which nodes are stubs, which are transit nodes, which pairs of nodes are neighbors, and which prefixes are hosted where) and information about the individual ASes' policies. We do *not* attempt to protect the network information, and we protect the policy information with SMPC.

More precisely, we assume that the interdomain topology is explicitly revealed by domains (i.e, they inform the computational clusters who their neighbors are), which makes it easier to execute 'what-if' scenarios. This peering topology is not officially revealed by BGP, but large components of it are easily inferred by looking at BGP routes; thus, we do not consider this a major violation of domain privacy. In a similar vein, we assume that stub domains (domains not willing to provide transit to anyone) are willing to reveal this fact. We note that currently the Internet has over 50,000 domains, the vast majority of which are estimated to be stub domains. Thus, the interdomain route computation can consist of two components: computing the routing choices for each of the transit domains, and then having the stub domains choose among the transit choices made available to them.

A more significant simplification, which is not necessary for the correctness of our approach but is important for its feasibility, is that we compute routes on a per-domain rather than per-prefix basis. To this end, we require domains to explicitly reveal which prefixes they host and assign to each prefix the route to the domain to which it belongs. This limits certain policy choices (routing two prefixes destined for the same domain along different paths), but greatly increases our approach's scalability.

In principal, SMPC could be used to hide network information as well as policy information, and we may explore that possibility in future work.

Lastly, even though our approach gives us great flexibility in how we might structure the computation, in our initial exploration we use an approach that essentially mimics BGP's per-domain iterative computation: Our computation proceeds in rounds, and in each round the clusters compute the same policy choices that the domain would have. Of course, the advantage here is that this iterative computation is now occurring among k computational clusters, not run over the global Internet between all the domains. With next-hop policies and Gao-Rexford constraints, it is known that such a round-based computational process will converge after $2D+1$ rounds [11], where D is the height of the Gao-Rexford hierarchy [4].

Note that there is an exceptional level of parallelism in this computation. First, the paths for each destination prefix can be computed independently. Second, the choices for each domain during a given round can be computed in parallel.[2] Third, some of the basic crypto operations in the SMPC algorithm can be done in parallel. This is why the available number of machines should be large.

Our proof-of-concept shows that this approach can indeed be made to work but points to computation time as a concern (though, as noted above, there is substantial room for improving the SEPIA results).

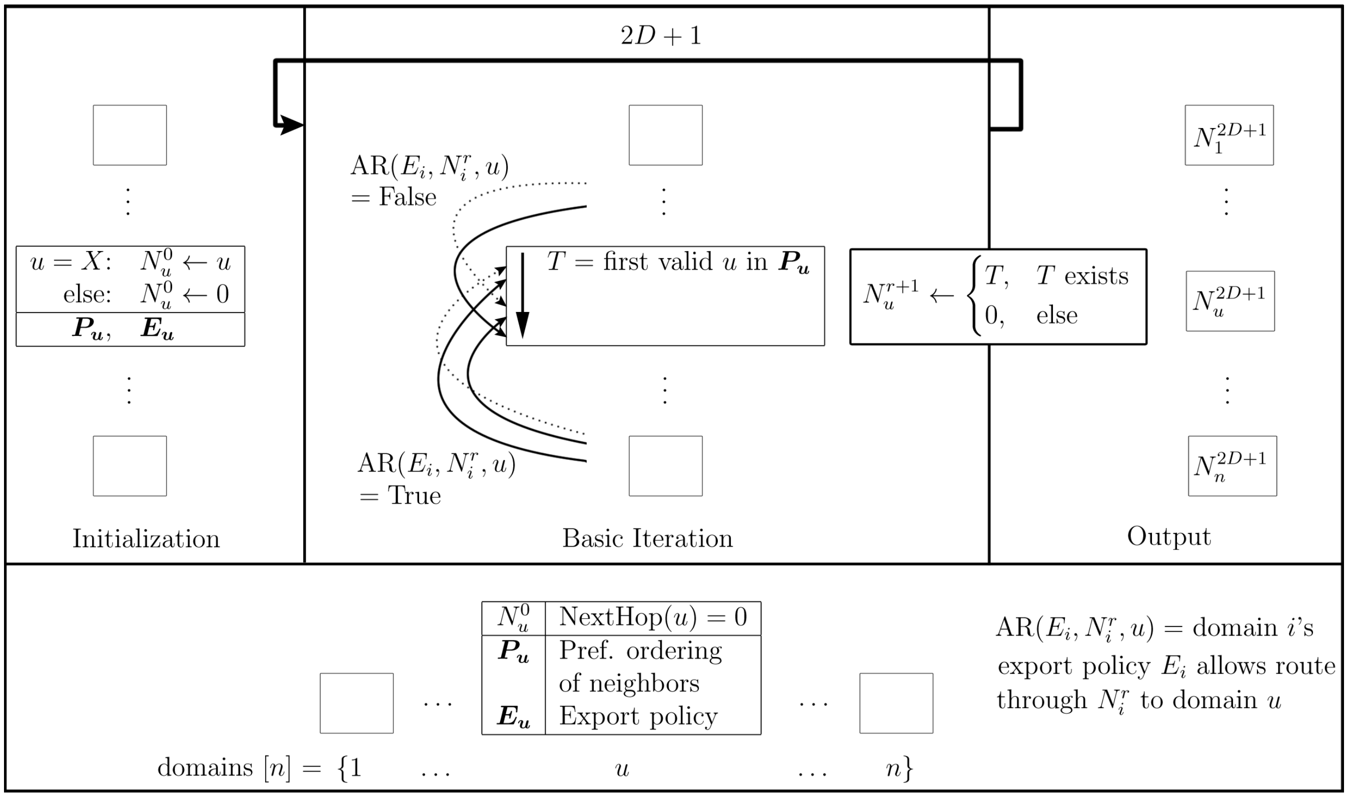
3.2 Overall Structure

Ignoring the startup phase and looking only at what happens incrementally, our approach works as follows. When a domain has a link event (failure or recovery) or a change in policy, it informs the cluster. For a link event, this is sent "in the clear" to the clusters, who then recompute routes with this modified topology and distribute them to all the domains. Inside a domain, these updates are handled in a manner similar to RCP [2], or one could have a small set of top-level route reflectors distribute the information. If two domains have multiple peering links, they are free to exchange information about which of these links to use; we leave this outside of our simplified BGP computation, but domains can find some other method of coordination.

Because the topology is known, the clusters can precompute routes for a variety of failure scenarios. When informed of a topology change for which the routes have been precomputed, they can immediately send out the result of their computation. Thus, the response time of this algorithm could be reduced, in most cases, to the latencies of communication to/from the clusters.

To avoid having a "single-point-of-failure" (if one of the cluster nodes goes down, that set of clusters cannot compute routes), we envision having several sets of clusters, all computing routes in parallel. These routes should agree,

[2]The convergence result in [11] applies regardless of whether the updating in a round is synchronous or asynchronous.



$2D+1$
$\mathrm{AR}(E_i, N_i^r, u) = \text{False}$
$\mathrm{AR}(E_i, N_i^r, u) = \text{True}$
$u = X$: $N_u^0 \leftarrow u$
else: $N_u^0 \leftarrow 0$
$\boldsymbol{P_u}$, $\boldsymbol{E_u}$
T = first valid u in $\boldsymbol{P_u}$
$N_u^{r+1} \leftarrow \begin{cases} T, & T \text{ exists} \\ 0, & \text{else} \end{cases}$
N_1^{2D+1}
N_u^{2D+1}
N_n^{2D+1}
Initialization
Basic Iteration
Output
N_u^0 NextHop$(u) = 0$
$\boldsymbol{P_u}$ Pref. ordering of neighbors
$\boldsymbol{E_u}$ Export policy
$\mathrm{AR}(E_i, N_i^r, u)$ = domain i's export policy E_i allows route through N_i^r to domain u
domains $[n] = \{1 \quad \dots \quad u \quad \dots \quad n\}$

Figure 1: A visual representation of the computation. We show domains and the associated preference list (P_i), and export table (E_i). During a round, r, the algorithm computes the next hop table N_i^r for each domain. This next hop table is used to compute information exchanged between domains, and is ultimately the output from the algorithm.

but if they do not then there should be a rank ordering for deciding which set of clusters to follow.

There is an obvious bootstrap problem: how will domains reach the clusters, and vice-versa? We will run a traditional form of BGP on a small set of addresses: one for each cluster and domain. This "normal" BGP will provide routes for traffic to and from the clusters. The addresses will come from a currently unused portion of the address space; so there is no worry about clashes between the two forms of BGP. This approach may be overkill; the update traffic will likely be very light, and one could imagine using OSPF-style broadcasts, which would require the domains to forgo policy control over this routing traffic.

3.3 Structure of Computation

We implemented this scheme using the SEPIA Java library for SMPC [1]. We expected that using a general-purpose platform like SEPIA would be computationally inefficient, but this implementation provides a useful proof-of-concept with existing tools.

Figure 1 presents a graphical representation of our algorithm. We describe our implenetation using SEPIA below.

We assume that the topology of the network is publicly known to all parties and denote by $[n] = \{1, \dots, n\}$ the set of domains. Each domain u with neighboring domains $v_1, \dots, v_{d_u}$ has two private inputs: (1) a ranking of its neighbors; and (2) an export policy that, for every two neighboring domains i and j, specifies whether u is willing to make routes that have i as a next-hop available to j. As the computation is done independently for each destination domain, fix a specific destination X.

Initialization. For each domain $u \in [n]$, we initially define $NextHop(u) = u$ if $u = X$ (that is, u is the destination), and $NextHop(u) = 0$ otherwise. Throughout the execution of the protocol, the value $NextHop(u)$ specifies u's current next hop en route to the destination X, and $NextHop(u) = 0$ captures the scenario that u currently has no next hop en route to X.

The basic iteration. As discussed in Section 3, the protocol consists of $2D + 1$ iterations, where D is the depth of the Gao-Rexford hierarchy. In each iteration, the following is executed in parallel for each domain $u \in [n]$:

- Set $NextHop(u)$ to be u's most preferred neighboring domain i, according to u's ranking of neighbors, such that (1) $NextHop(i) \neq 0$ and (2) according to i's export policy i is willing to announce routes through $NextHop(i)$ to u.

- Set $NextHop(u) = 0$ if no neighboring domain i as defined above exists.

Output. Output for each domain u the next hop domain stored in $NextHop(u)$.

3.4 Computation Time Estimates

Here we estimate the computation time of this algorithm, under the following assumptions. We assume *perfect* parallelization between the domains (each domain is computed in a separate process) and for each destination domain; so one in general has the square of the number of domains running in parallel. Thus, the unit of parallelism is computing the routes used by a single domain for a single destination domain. This requires a vast number of processes, but we expect many of them will be quite fast because the domains have so few neighbors.

On the one hand, these are extremely optimistic assumptions. On the other hand, this is extremely unoptimized code, so we hope that the gains made through improving the implementation will outweigh the overoptimism about parallelism.

With these caveats in hand, we found that the operation for a single domain with 19 neighbors to compute 11 rounds was .13 seconds. This was on a 2.7 GHz quad-core machine with 8 GB of memory. The algorithm scales poorly as you increase the number of neighbors (remember we don't count stub neighbors), but we expect this can be improved by increasing the parallelization across these neighbors.

4 Discussion

Our experience with our SEPIA-based implementation suggests that this approach may be feasible. We hope to confirm this with additional work on the following topics:

- Improving the performance in the special case considered here with a from-scratch implementation of the protocol.
- Investigating the performance when fully parallelized.
- Implementing route precomputation and investigating its impact on under real network traces.

However, we think the most important implication of this approach may be that, be breaking away from BGP's distributed computational framework, we can explore new policy models that provide better stability and/or more flexibility.

5 References

[1] M. Burkhart, M. Strasser, D. Many, and X. Dimitropoulos. SEPIA: Privacy-preserving aggregation of multi-domain network events and statistics. In *Proc. of Usenix Security*, August 2010.

[2] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and K. van der Merwe. Design and Implementation of a Routing Control Platform. In *Proc. of NSDI*, 2005.

[3] J. Feigenbaum, B. Pinkas, R. Ryger, and F. Saint-Jean. Secure Computation of Surveys. In *EU Workshop on Secure Multiparty Protocols*, 2004.

[4] L. Gao and J. Rexford. Stable internet routing without global coordination. *IEEE/ACM Trans. Netw.*, 9(6):681–692, 2001.

[5] P. B. Godfrey, I. Ganichev, S. Shenker, and I. Stoica. Pathlet routing. In *ACM SIGCOMM*, August 2009.

[6] P. Golle. A private stable matching algorithm. In *Financial Cryptography*, pages 65–80, 2006.

[7] J. P. John, E. Katz-Bassett, A. Krishnamurthy, T. Anderson, and A. Venkatarami. Consensus routing: The Internet as a distributed system. In *NSDI*, June 2008.

[8] N. Kushman, S. Kandula, and D. Katabi. Can you hear me now?! it must be BGP. *SIGCOMM CCR*, April 2007.

[9] Y. Lindell and B. Pinkas. Secure Multiparty Computation for Privacy-Preserving Data Mining. *Journal of Privacy and Confidentiality*, 1:59–98, 2009.

[10] J. Naous, M. Walfish, A. Nicolosi, D. Mazieres, M. Miller, and A. Sehra. Verifying and enforcing network paths with ICING. In *CoNext*, December 2011.

[11] R. Sami, M. Schapira, and A. Zohar. Searching for stability in interdomain routing. In *INFOCOM*, pages 549–557, 2009.

[12] M. Schapira, Y. Zhu, and J. Rexford. Putting BGP on the right path: A case for next-hop routing. In *HotNets*, October 2010.

[13] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[14] L. Subramanian, M. Caesar, C. T. Ee, M. Handley, M. Mao, S. Shenker, and I. Stoica. HLP: A next-generation interdomain routing protocol. In *ACM SIGCOMM*, August 2005.

[15] X. Yang, D. Clark, and A. Berger. NIRA: A New Inter-Domain Routing Architecture. *IEEE/ACM Trans. Networking*, 15(4):775–788, August 2007.

[16] A. C. Yao. Protocols for secure computation. In *IEEE Symposium on Foundations of Computer Science*, pages 160–164, 1982.

[17] A. C. Yao. How to generate and exchange secrets. In *IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.

[18] M. Zhao, W. Zhou, A. J. T. Gurney, A. Haeberlen, M. Sherr, and B. T. Loo. Private and verifiable interdomain routing decisions. In *ACM SIGCOMM*, August 2012.