Having your Cake and Eating it too: Routing Security with Privacy Protections

Alexander J. T. Gurney
University of Pennsylvania

Andreas Haeberlen
University of Pennsylvania

Wenchao Zhou
University of Pennsylvania

Micah Sherr
Georgetown University

Boon Thau Loo
University of Pennsylvania

ABSTRACT

Internet Service Providers typically do not reveal details of their interdomain routing policies due to security concerns, or for commercial or legal reasons. As a result, it is difficult to hold ISPs accountable for their contractual agreements. Existing solutions can check basic properties, e.g., whether route announcements correspond to valid routes, but they do not verify how these routes were chosen. In essence, today's Internet forces us to choose between per-AS privacy and verifiability.

In this paper, we argue that making this difficult tradeoff is unnecessary. We propose *private and verifiable routing* (PVR), a technique that enables ISPs to check whether their neighbors are fulfilling their contractual promises to them, and to obtain evidence of any violations, *without* disclosing information that the routing protocol does not already reveal. As initial evidence that PVR is feasible, we sketch a PVR system that can verify some simple BGP policies. We conclude by highlighting several research challenges as future work.

Categories and Subject Descriptors

C.2.2 [**Computer Systems Organization**]: Computer-Communication Networks—*Network Protocols*; C.2.6 [**Computer Systems Organization**]: Computer-Communication Networks—*Internetworking*

General Terms

Algorithms, Design, Reliability, Security

Keywords

Security, Privacy, Accountability, Interdomain routing, BGP

1. INTRODUCTION

In today's Internet, there is an inherent tension between verifiability and privacy. Network operators are routinely entering into agreements that concern various aspects of their interdomain routing policies [6]. For example, network A might promise network B that it will act as B's provider, or it might enter into a 'partial transit' relationship [24, 21] with network B and promise to deliver routes from, e.g., European peers in preference to other routes. On the one hand, it is legitimate for network B to want to verify that network A's policy logic is consistent with its promises. On the other hand, the actual policies are considered to be business secrets, and only the chosen routes are exported via BGP to the other networks. Thus, network A's promises are worth little: it can do whatever it likes.

The value of verifiability. There are many reasons why a network might not keep a promise it has made to another network. Intentional violations are certainly a possibility; it has been shown that networks may have economic incentives to lie about their routes [8], and there are notorious examples of malicious behavior [12, 19]. However, unintentional violations seem to be far more common, e.g., due to misconfigurations [16], compromised routers [18], or equipment failures [23]. Whatever the cause, it seems reasonable for networks to want to detect and diagnose such violations – to fix routing problems, or simply to ensure that they are getting their money's worth.

In this paper, we focus on verifying a network's control-plane actions; we do not consider promises about data-plane performance, e.g., with respect to bandwidth, latency, or packet loss rate. Techniques for verifying data-plane performance have been studied independently, e.g., in [1], although not with a particular focus on privacy. Secure variants of BGP, such as S-BGP [13], have been proposed as mechanisms for ISPs to check that a routing announcement does correspond to the claimed path and destination, but these mechanisms do not address the important question of whether the *route decision process* matches expectations.

The value of privacy. ISPs have traditionally been reluctant to disclose details of their routing policy. While some aspects of configuration may be revealed to neighbors, included in a route registry, or exposed indirectly via looking glass services, the *full* policy is not published anywhere. Operational security concerns, as well as commercial or legal considerations, weigh against disclosure. Interdomain routing policy encodes the nature of the business relationships between the participants, public knowledge of which could compromise negotiations with other parties ("I want the deal they're getting") or long-term commercial plans ("It looks like you're expanding in Belgium"). Knowledge of which routes are available to an AS can be useful to a competitor, even without access to the policy.

To some extent, the privacy barrier can be penetrated today; for example, it is possible to build a highly accurate map of the Internet [15, 22], or to classify AS business relationships on the basis of publicly available data [5, 7]. These inferences go beyond what was intended in publishing that data, but even they are inherently unable to verify certain types of promises, e.g., whether a better route *would have* been available, but was not exported. We could enable complete verification by revealing all routing tables, similar to [11], but then everything is revealed, and there is no way to limit the information that might be learned.

Verifiability or privacy? Intuitively, it seems that verifiability and privacy are conflicting goals – by revealing more information, we can improve verifiability, but we reduce privacy. Despite this intuition, this paper provides initial evidence that it is possible to have one's cake and eat it too: we show that a network can allow its neighbors to *collectively* verify a simple but realistic example promise (shortest-path routing to a given IP prefix) *without revealing any additional information to each other*. We also briefly describe how our method could be generalized to handle more complex promises.

2. PROMISES AND POLICIES

Neighboring networks will surely have some understanding of the kind of routes they expect to be advertised between them, whether or not there is a formal contractual agreement to that effect. What sort of *promise* might an AS make to its neighbor? The absolute minimum is what we implicitly have today: "You get what you're given", which is no guarantee at all, since it cannot be violated. Some stricter potential promises include:

1. "I will give you the shortest route I receive."
2. "I will give you the shortest route out of those received from a specific subset of neighbors."
3. "I will give you a route no more than ϵ hops longer than my best route."
4. "The route you get is no longer than what I tell anybody else."

Promises further down the list allow more latitude to the sending AS, and would therefore be preferable for it, whereas the recipient might want a stronger guarantee about what routes it might obtain. The correctness of an AS's actions must be judged against the standard of whatever promise is agreed to exist, as a result of negotiation between the AS and its neighbors.

These promises can be understood as specifying, for each set of input routes the AS might receive, some set of permissible routes that its output must be drawn from. A *violation* occurs whenever an AS emits a route that was not in its permitted set, given the inputs it had received. Current BGP is unable to detect violations of any kind, since there is no way for the recipient of a route to know what inputs were available. S-BGP can detect violations of the promise that any output route (for an external destination) must have come from the neighbor that it claims. However, there is still no way to enforce any promise that depends on the internal decision rules used by an AS.

There will be many ways to implement each promise in terms of the language of router configurations. Equally, an AS has only a single concrete configuration, but it can still make different promises to different neighbors, since each promise is an overapproximation to the true behavior. This decoupling matches the privacy concerns: an AS certainly does not wish to disclose all configuration details to all neighbors, but only those details which are relevant to whatever promise has been made.

2.1 Policies as route-flow graphs

The actual route computation process is complicated, but for the purposes of verification, a network can present a simplified version of reality as a set of decision rules, whose structure corresponds to the promise that has been made. A recipient can check that the rules would, if followed, result in the promise being met; and at the same time, the sending network can prove that its actions are in accordance with the rules (as explained in Section 3).

For us, a rule is an operation that takes some set of input routes and emits a set of output routes (which may be a single route, or no route at all). The entire BGP decision process could be modeled by a single black-box rule, but it is preferable to decompose it into smaller interconnected pieces that can be hidden or revealed individually. We will refer to these pieces as *operators*, which operate on *variables* – typically routes and sets of routes, but also communities, AS paths, prefixes, etc. An example would be an operator for selecting, from a given set of routes, the routes with minimal AS path length (the second step in BGP). A pipeline of such operators, one for each attribute, makes up the usual route selection process. But there are some situations where other operator topologies are of use—say, when routes from different neighbors should be treated differently—so in general the connections between operators and variables will form a graph. In analogy to data flow graphs, we will refer to this graph as the *route-flow graph*. Two simple examples of such graphs will be presented in Section 3.

2.2 Access control policies

Visibility of operators and variables is governed by an access control policy. Let V be the set of vertices in the route-flow graph (operators and variables), and let N be the set of participating networks. A function $\alpha : N \times V \rightarrow \{\text{TRUE}, \text{FALSE}\}$ expresses which networks are allowed to see which parts of the graph. If v is a variable vertex, $\alpha(n, v) = \text{TRUE}$ means that network n is allowed to learn the current value of v; if v is an operator vertex, n is allowed to learn which function v computes.

A network may be able to tell, given the rules to which it has access, whether particular promises made to it will be kept. This is based purely on static inspection of the route-flow graph, tracing connections from input variables (which correspond to incoming route announcements) to output variables. The problem, of course, is that a malicious network might not follow the declared rules: we need incorrect computations (and, hence, violations of promises) to be detectable.

2.3 Goal: PVR

We propose *PVR (private and verifiable routing)*, which has the following properties:

Detection If an AS A incorrectly evaluated its route-flow graph, an incorrect result is visible to at least one neighbor B, and all of A's neighbors are correct, then at least one neighbor can detect this.

Evidence If an incorrect evaluation is detected in an AS A, then at least one AS B can obtain evidence against A that will convince a third party.

Accuracy If an AS A has evaluated its route-flow graph correctly, no correct AS can detect a violation in A, and A can disprove any evidence that is presented against it.

Confidentiality No AS will learn information from running PVR that it could not learn in the unsecured system, unless this was explicitly authorized by α.

The last point requires some elucidation. If the unsecured system already reveals a fact to some network, whether directly or indirectly, we should not demand that the protected system conceal this fact. For example, if X promises Y that it will deliver the shortest route, and Y receives a route going through X's neighbor Z, then Y can infer that (if X was telling the truth) X had no route that was shorter than Z's. Thus, Y learns the values of some of X's input variables, even though, according to α, it may not have access. However, note that α only controls what Y may *additionally* learn through PVR. Since the information is already revealed by standard BGP in this case, we do not consider this to be a violation of confidentiality.

3. A SIMPLE PVR SYSTEM

Presenting a general PVR system is beyond the scope of this paper; rather, our goal here is to present evidence that PVR is feasible, and to illustrate how a PVR system could work. To this end, we present a simple PVR system that enables a network to verify the second example promise from Section 2, i.e., that the route advertised by a neighbor network was the shortest route it received from a known subset of its peers.

We begin with the simple scenario shown in Figure 1. Network A is connected to neighbors $N_1, \ldots, N_k$ and B, and we assume that this is known to each of the networks. N_1 through N_k each advertise to network A a route r_i to some prefix, and A has promised to network B that it would export the shortest of these routes. To make things challenging,[1] we assume $\alpha(N_i, r_i) = \alpha(B, r_0) = \text{TRUE}$, $\alpha(n, \min) = \text{TRUE}$ for all networks n, and $\alpha(n, v) = \text{FALSE}$ otherwise. Our goal is to enable $N_1, \ldots, N_k$ and B to collectively verify that A is keeping its promise, without forcing them to reveal additional information to each other. In particular, none of the N_i should learn whether some other N_j has advertised a route to A, or which route was chosen by A. B obviously learns the chosen route, but should not learn anything about any other routes, except that they would have been longer.

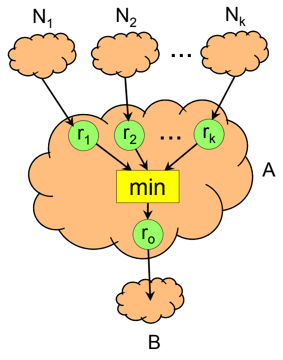

Figure 1: PVR example.

We adopt a conservative threat model and assume that an unknown subset of the networks is Byzantine and can behave arbitrarily. Moreover, incorrect networks may collude and share knowledge via zero-latency private communication channels.

3.1 Strawman designs

We can imagine a strawman solution in which the networks use secure multiparty computation (SMC) [9], possibly with output privacy [3], to compute the routes (there are even SMC protocols that provide both security and privacy, e.g., [14]). However, such a system would seem prohibitively expensive: even with only five players, state-of-the-art SMC systems take about 15 seconds of computation time for a simple task like voting [2], and such a task would have to be performed for every single BGP update. Apart from being computationally infeasible, this solution is also too weak: it gives us excellent privacy but no evidence, so we cannot enforce contracts in this way. Another strawman could be built using general zero-knowledge proofs (ZKPs) [10], which are also very general, but at the same time, there are scaling concerns as the complexity of policy increases.

[1]If a system can enforce some access control policy α, it can trivially enforce any policy that is strictly weaker.

In any case, the full generality of SMC and ZKP seems unnecessary for interdomain routing. Although BGP policy is complex, and could be arbitrary, there are common patterns which tend to reoccur in practice. Hence, it seems more promising to aim for a domain-specific solution, similar to the one we propose here.

3.2 Example #1: The existential operator

To explain our PVR system, we first consider a simplified variant of the scenario in Figure 1 in which the minimum operator is replaced by an existential operator – that is, A promises B that it will export a route whenever at least one of the N_i provides one. To enable PVR verification of this promise, we can break it down into two conditions that can be verified independently by A's neighbors:

1. B verifies that, if a route was exported by A, then that route was provided to A by some N_i; and
2. Each N_i verifies that, if it provided a route r_i to A, then A exported *some* route to B.

To support condition 1, we can sign all the routing announcements. Supporting condition 2 is more difficult, since we would like to prevent the N_i from learning which route was chosen. To achieve this, we can ask A to commit to a bit b, which is set to 1 whenever A has received at least one route. A can do this by publishing a commitment $c := H(b\,||\,p)$, where H is a cryptographic hash function and p is a random bitstring. A's neighbors can gossip about c to ensure that they all have the same view of b, although they do not yet know b's value.[2]

Now A can reveal b and p to each N_i that has provided a route, and the signed route (if any) to B. All neighbors that have received b and p immediately verify whether $c = H(b\,||\,p)$. B also checks condition 1 by verifying that either $b = 0$ or it has received a properly signed route, and the N_i each check condition 2 by verifying that, if N_i has provided a route to A, then A has revealed b and p to N_i, and $b = 1$. Since both conditions are verified, we achieve detection, but since no neighbor has learned anything it did not already know, we *also* achieve confidentiality. If A performs the computation correctly, all the checks will pass (ensuring accuracy), and any neighbor that detects a violation has also obtained evidence to prove it.

We note that, since BGP updates contain the AS path, it is obvious which N_i has to be the signer. But this is not a requirement. Suppose we apply PVR to a link-state protocol that only exports whether a path exists. Then the N_i can use a ring signature scheme, such as [20], to sign the statement "A route exists". Thus, B could tell that *some* N_i had provided a route, but it could not tell which one.

3.3 Example #2: The minimum operator

We now return to the original scenario in Figure 1, i.e., A promises B to export the *shortest* route among the ones provided by $N_1, \ldots, N_k$. We can extend our existential operator from Section 3.2 to verify this promise by adding a third condition:

3. Each N_i that has provided a route r_i to A verifies that the route A has exported to B is *not longer* than r_i.

To verify this condition, we need A to commit to more values. Suppose the maximum AS-path length at A is k. Then we can ask A to compute k bits $b_1, \ldots, b_k$, such that $b_i = 1$ iff at least one of the input routes has a path length of i or less. A commits to each b_i, and A's neighbors can again gossip about the commitments to detect equivocation.

To each N_i that has provided a route r_i to A, A now reveals the bit $b_{|r_i|}$. Each such N_i then checks the commitment to verify that this bit is 1 (clearly, the chosen route cannot be longer than N_i's route). A also reveals *all* the bits b_i to B. B verifies that a) if at least one bit is set to 1, then it must have received a properly signed route, and b) if some b_i is set to 1, then all the b_i, $j > i$, must also be set to 1. Again, any misbehavior by A can be detected by at least one neighbor, but neither B nor any N_i learn anything in the process that they did not already know.

3.4 Building blocks

From these examples, we can identify three mechanisms that are needed to evaluate PVR queries, and that are likely to be part of any generalized PVR system capable of checking more complex promises. The three building blocks are:

1. A **commitment mechanism** to ensure that a network cannot change its mind about its decisions after the fact, and that all the neighbors are seeing the same decisions;
2. A **selective disclosure mechanism** to selectively reveal information to authorized networks, without leaking any other information; and
3. A **verification mechanism** to establish whether a promise has been kept or violated.

In general, the verification mechanism requires some additional data (such as the bits b and b_i above). We refer to this data as *evidence* that a promise has been kept.

3.5 Generalizing the mechanism

So far, we have only considered route-flow graphs with a single operator. However, recall from Section 2.1 that we would also like to verify more general route-flow graphs with multiple operators and variables, which correspond to more complex routing policies. In such a graph, an edge (o, v) from an operator o to a variable v indicates that v is computed by o; an edge (v, o) indicates that v is an input to o. For example, the simple graph in Figure 2 enforces a policy that corresponds to the promise "I will export some route via $N_2, \ldots, N_k$ unless N_1 provides a shorter route".

[2]If p were not included in the hash, any neighbor could simply check whether $c = H(0)$ or $c = H(1)$.

To achieve this, we must enable a network's route-flow graph to be navigated by that network's neighbors without learning about the existence of rules or variables they are not authorized to see. This means that we need a more general mechanism for commitment and selective disclosure, in which we can store some information $I(x)$ for each vertex x – i.e., the operator type for operator vertices and the current value for variable vertices – as well as information about incoming and outgoing edges. We discuss how to do this next.

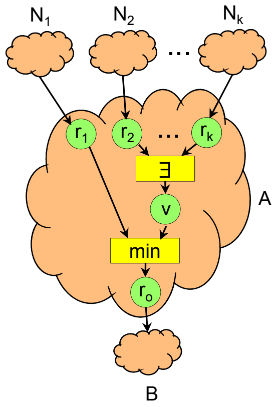

Figure 2: Example with multiple operators.

3.6 Commitment and selective disclosure

We assume that a) each network can assign a unique bitstring to each of its rules, as well as to any output produced by these rules, and that b) the resulting bitstrings are prefix-free, i.e., no valid bitstring is a prefix of another valid bitstring. A simple way to ensure both is to encode the string `rule(`x`)` for each rule x and `var(`v`)` for each variable v, although there are more efficient representations.

We can now construct a Merkle Hash Tree (MHT) [17] that (conceptually) contains one leaf for each bitstring. We imagine the two edges out of each inner node of the MHT to be labeled 0 and 1; thus, we can associate each leaf with the bitstring that is constructed by concatenating all the labels along the path from the root to the leaf. Since the bitstrings are prefix-free, no inner node can construct to a valid bitstring. Of course, an actual network will only have a small finite number of rules and variables instantiated at any given time. The network constructs its MHT as the union of a) the instantiated leaves, b) all the inner nodes along a path from an instantiated leaf to the root, and c) all the immediate children of these inner nodes.

The MHT gives us an efficient way to perform commitments: Each network simply computes the hash value of its MHT's root node, signs that hash value, and publishes it to its neighbors. The neighbors can then gossip about the hash value to ensure that they all have the same view of the MHT. The MHT also gives us selective disclosure: To disclose the information $I(x)$ in a vertex x to a neighbor, the network can send the neighbor $I(x)$ and all the hash values for interior nodes along the path from x to the MHT's root. The neighbor can use these values to recompute the top-level hash value, and to compare it against the previously published value. If successful, this validates $I(x)$. Since the neighbor does not know whether the hash values are random bitstrings or hashes of 'real' interior nodes, this does not reveal the presence or absence of any vertices other than x.

3.7 Graph navigation

We now consider the question of how $I(x)$ should be chosen. Clearly, we must have a way to reveal the structural information (incoming and outgoing edges) independent of the data (route, or type of operator), so that a neighbor may navigate parts of the graph it is not allowed to see. For instance, in our example from Section 3.3, B would want to verify that the minimum was computed over routes provided specifically by $N_1, \ldots, N_k$, even if it is not authorized to see what the routes were. We can enable this by choosing $I(x)$ to be $(c(x_1^p, \ldots, x_a^p),\ c(x_1^s, \ldots, x_b^s),\ c(\bar{x}))$, where the $c(\cdot)$ are commitments and the x^p and x^s are bitstrings identifying predecessor and successor vertices, respectively. $\bar{x}$ is the route itself (in the case of a variable) or the operator type and the evidence (in the case of an operator). Thus, the three types of information can be revealed independently, depending on the authorization of the querying neighbor.

3.8 Overhead

The simple PVR mechanism we have sketched here is certainly far less general than SMC or ZKP. However, in return, it also seems far less expensive. The most expensive operations we have used are a cryptographic hash-function (such as SHA-256), which are relatively cheap, and a public-key signature scheme (such as RSA). A RSA-1024 signature takes about two milliseconds on current hardware. This overhead can be burdensome during BGP message bursts, but it seems feasible to sign messages in batches, perhaps using a small MHT to reveal batched routes individually. Of course, we have demonstrated only two very simple operators, and it is possible that more advanced operators may require more expensive cryptography. Nevertheless, it is encouraging that our PVR mechanism already seems sufficient to verify nontrivial promises, like the one shown in Section 3.5.

3.9 Summary

Our simple PVR system demonstrates the feasibility of PVR. We have discussed only two simple operators, and there are clearly many practical challenges that we have not yet addressed. However, recall that our main goal was to show that verifiability and privacy are not mutually exclusive, and that it seems possible to build efficient systems that can provide both. Building a deployable PVR routing system will require solving additional research challenges, some of which we describe in the next section.

4. CHALLENGES

More operators: Clearly, the two operators we have sketched in Sections 3.2 and 3.3 are not sufficient to express all routing policies that a network would want to use. For example, we do not yet have operators that evaluate communities or check for the presence of particular ASes on the path. To build a practical PVR system, it will be necessary to extend the set of operators considerably. Also, such a system should have language support for compiling a high-level policy description (or router configuration file) into a compact route-flow graph.

Minimum access: Not all promises are verifiable under all access control policies. A trivial example entails a network that exports a route but does not allow any of its neighbors to see the operator that was used to derive it; thus, promises about that route are not verifiable. A practical PVR system must have a way for a network's neighbors to tell whether a) the visible route-flow graph implements a given promise and b) the access privileges granted by the network are sufficient to verify that promise.

Structural privacy: When a network allows a neighbor to see only part of the logic that was used to derive a certain route, it may have to reveal some operators without specifying what they do. This might still leak some information to the neighbor, since they may be able to guess the purpose of a computation from its structure. It should be possible to avoid this by introducing some form of hierarchy into the route-flow graph, i.e., by adding a composite operator whose internal structure is only revealed to authorized neighbors, analogous to the approach proposed by Davidson *et al.* [4].

Other applications: PVR may be useful not only for network-level protocols, but also more generally to distributed systems that span more than one administrative domain. Both privacy and security concerns are common in such systems, and an approach that combines both seems attractive. With our domain-specific approach, we cannot expect to match the generality of SMC or ZKP, but we may be able to support some of the most common operations, and thus reduce the reliance on SMC or ZKP.

5. CONCLUSION

We have argued that verifiability and privacy are both valuable goals for an interdomain routing protocol and that, contrary to popular belief, there is no need to choose between them. To demonstrate that it is feasible to provide both properties simultaneously, we have sketched the design of a simple system that achieves this for a small but nontrivial set of routing policies, without relying on general but expensive primitives such as multiparty computation or zero-knowledge proofs. We have also described several research challenges that need to be addressed before practical routing systems of this type can be built and deployed.

Acknowledgments

We thank the anonymous reviewers for their helpful comments and suggestions. This work was supported by NSF grants IIS-0812270, CCF-0820208, CNS-1040672, CNS-1054229, CNS-1065130, and AFOSR grant FA9550-08-1-0352.

6. REFERENCES

[1] K. J. Argyraki, P. Maniatis, and A. Singla. Verifiable network-performance measurements. In *CoNEXT*, 2010.

[2] A. Ben-David, N. Nisan, and B. Pinkas. FairplayMP: A system for secure multi-party computation. In *Proc. ACM CCS*, 2008.

[3] E. Bresson, D. Catalano, N. Fazio, A. Nicolosi, and M. Yung. Output privacy in secure multiparty computation. In *Proc. YACC*, 2006.

[4] S. B. Davidson, S. Khanna, T. Milo, D. Panigrahi, and S. Roy. Provenance views for module privacy. In *Proc. PODS*, 2011.

[5] X. Dimitropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, kc claffy, and G. Riley. AS relationships: inference and validation. *SIGCOMM CCR*, 37:29–40, January 2007.

[6] P. Faratin, D. Clark, P. Gilmore, S. Bauer, A. Berger, and W. Lehr. Complexity of internet interconnections: Technology, incentives and implications for policy. In *Proc. TPRC*, 2007.

[7] L. Gao. On inferring autonomous system relationships in the internet. *IEEE/ACM Trans. Netw.*, 9:733–745, 2001.

[8] S. Goldberg, S. Halevi, A. Jaggard, V. Ramachandran, and R. Wright. Rationality and traffic attraction: Incentives for honestly announcing paths in BGP. In *Proc. ACM SIGCOMM*, Aug. 2008.

[9] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proc. ACM STOC*, 1987.

[10] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM*, 38:690–728, 1991.

[11] A. Haeberlen, I. Avramopoulos, J. Rexford, and P. Druschel. NetReview: Detecting when interdomain routing goes wrong. In *Proc. NSDI*, Apr 2009.

[12] A. J. Kalafut, C. A. Shue, and M. Gupta. Malicious hubs: detecting abnormally malicious autonomous systems. In *Proc. INFOCOM*, 2010.

[13] S. Kent, C. Lynn, and K. Seo. Secure border gateway protocol (S-BGP). *IEEE JSAC*, 18(4):582–592, 2000.

[14] Y. Lindell and B. Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *Proc. EUROCRYPT*, 2007.

[15] H. V. Madhyastha, E. Katz-Bassett, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane Nano: path prediction for peer-to-peer applications. In *Proc. NSDI*, 2009.

[16] R. Mahajan, D. Wetherall, and T. Anderson. Understanding BGP misconfiguration. In *Proc. ACM SIGCOMM*, Sep 2002.

[17] R. Merkle. Protocols for public key cryptosystems. In *Proc. Symposium on Security and Privacy*, Apr. 1980.

[18] O. Nordstroem and C. Dovrolis. Beware of BGP attacks. *ACM CCR*, Apr. 2004.

[19] N. Patrick, T. Scholl, A. Shaikh, and R. Steenbergen. Peering Dragnet: anti-social behavior amongst peers, and what you can do about it, 2006. NANOG 38: `http://nanog.org/meetings/nanog38/presentations/scholl-peering-dragnet.pdf`.

[20] R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *Proc. ASIACRYPT*, 2001.

[21] M. Roughan, W. Willinger, O. Maennel, D. Perouli, and R. Bush. 10 lessons from 10 years of measuring and modelling the Internet's Autonomous Systems. *IEEE Journal on Selected Areas in Commun.*, 29(9):1810–1821, 2011.

[22] R. Sherwood, A. Bender, and N. Spring. Discarte: a disjunctive internet cartographer. In *SIGCOMM*, 2008.

[23] J. Wu, Z. M. Mao, J. Rexford, and J. Wang. Finding a needle in a haystack: Pinpointing significant BGP routing changes in an IP network. In *Proc. NSDI*, May 2005.

[24] M. Yoshinobu. What makes our policy messy. BGP Workshop April 2009: `http://www.attn.jp/maz/p/c/bgpworkshop200904/`.