

Secure Traceroute to Detect Faulty or Malicious Routing

Venkata N. Padmanabhan and Daniel R. Simon
Microsoft Research

Abstract—

Internet routing is vulnerable to disruptions caused by malfunctioning or malicious routers that draw traffic towards themselves but fail to correctly forward the traffic. The existing approach to securing routing is to validate routing updates by verifying their authenticity, accuracy, and/or consistency. In this paper, we propose a different, adaptive approach, the central idea of which is a *secure traceroute* protocol that enables end hosts or routers to identify an arbitrarily severely misbehaving router, so that appropriate action can be taken.

I. INTRODUCTION

Although a great deal of attention has been paid to securing network communication, the security of network routing has been neglected by comparison. For example, the predominant inter-domain routing protocol in the Internet, BGP, includes no mechanism for verifying either the authenticity (correct origin) or the accuracy of the routing information it distributes. As a result, traffic can be severely disrupted by routers refusing to serve their advertised routes, announcing nonexistent routes, or simply failing to withdraw failed routes, as a result of either malfunction or malice. A particularly problematic case is that of sophisticated malicious routers (e.g., routers that have been compromised) that attempt to hide or disguise their misbehavior.

Two approaches have been suggested to solving this problem. One, typified by Secure BGP (S-BGP) [7], requires routing information to be authenticated—say, by digital signature—so that routers advertising false routing information can be held accountable when detected. However, the overhead of digital signature is large and can be prohibitive—for example, when bringing a failed Internet router back on line, at which time all BGP routing advertisements for that router must be digitally signed at once. Moreover, authentication of routing information does little to help detect or diagnose faulty routing information emanating from a router (for example, a compromised one); it only ensures reliable identification of the information’s origin (for after-the-fact, out-of-band blame assignment) should the misbehavior somehow be detected.

The other approach is to maintain a centralized registry of “plausibility” information about routing advertisements (akin to the Routing Arbiter [15]), so that blatantly

invalid routing information can be discounted when received. This approach can prevent the most egregious routing problems arising from router misconfigurations, but it is still vulnerable to a wide range of both inadvertent and malicious false advertisements for routes that a particular router may be “entitled” to advertise, but cannot (or simply will not) in fact serve.

We propose a third approach, in which routers, assisted by end hosts, adaptively detect poorly performing routes that indicate routing problems, and use a *secure traceroute* protocol to attempt to identify an offending router. Once the offending router has been identified, it can be routed around, the detection can be publicized, or out-of-band action can be taken to attempt to resolve the problem.

The key idea behind secure traceroute is to securely trace the path of existing traffic, rather than that of special traceroute packets, to prevent adversaries from misleading the tracer by treating traceroute and normal traffic differently. Secure traceroute responses are also authenticated, to verify their origin and prevent spoofing or tampering.

Our approach is orthogonal to and can complement schemes such as S-BGP that secure the routing protocol itself. Secure traceroute can be incrementally deployed in the Internet; moreover, it is a general technique that can be used to diagnose routing problems in other settings such as single-ISP networks, ad hoc networks, and peer-to-peer networks.

II. ASSUMPTIONS AND THREAT MODEL

We assume a network in which nodes are individually globally addressable (which is often, though not always, true in the Internet), and in which each (non-faulty) node, on receiving a packet with an origin and destination address, chooses a single next node to which to forward it. Our goal is to deal with fairly limited, localized routing problems in this setting; if the routing disruption is too widespread, involving too many nodes, then it can interfere with any attempt to investigate it. A typical example of our target problem might be a single faulty router somewhere in the network (although our approach certainly does not assume that there is only one faulty router). Note that it is not necessary to distinguish between a merely faulty (e.g., misconfigured) router and a malicious one; in practice, after all, sufficiently erroneous behavior can be indistinguishable from malice. (An adversary may, for instance,

attempt to disguise its malicious behavior as mere error in order to avoid later retribution.) Hence we assume faulty routers to be capable of arbitrary Byzantine behavior.

III. SECURE TRACEROUTE

The normal traceroute protocol involves the sender simply sending packets with increasing TTL (time to live) values, and waiting for an ICMP time-exceeded response from the node that receives the packet when the TTL expires. Normally, this protocol easily generates a sequence of addresses of nodes on the path to the destination, or at least up to the point where packets are being lost on a faulty link. However, a malicious router could intercept and alter traceroute traffic to give an arbitrary misleading impression—say, by letting only traceroute packets through, or by constructing fake responses to them so as to give the impression that they are getting through and demonstrating a fully functioning path (or a path with a problem elsewhere). Secure traceroute is intended to prevent this type of disruption of the traceroute process by verifying the origin of responses, and preventing traceroute packets from being handled differently from ordinary traffic.

As in normal traceroute, secure traceroute proceeds hop by hop, with each node on the path being asked to respond to traceroute traffic. However, there are several fundamental differences between them, including traceroute packet contents and the responses they elicit from routers. We outline the specifics of secure traceroute below:

1. In addition to their own address (which is included implicitly in the response packet), hosts responding to secure traceroute packets provide a next-hop address for the packet. Thus the node performing the traceroute always knows the (expected) next node on the path.
2. Prior to sending the traceroute packets, the tracing node establishes a shared key for encrypted, authenticated communication to and from the expected next node. (How this is done is described in Section IV.) Using this key, identifying information, which specifies the “signature” of the packets to be treated as traceroute packets, is securely passed from the tracing node to the expected next node. This information could include the source and destination addresses (or address ranges) of the selected packets, and the values (or constraints on the values) of certain fields on the packets. For example, it could be specified that all packets between certain source and destination addresses for which a certain field contains a particular value modulo a specific prime number should be examined. Alternatively, a random value (“nonce”) could be inserted into some field in the packet by the tracing node, and certain such values would be designated as signifying a traceroute

packet. Thus, traceroute packets will look indistinguishable from ordinary traffic to intermediate nodes, and will therefore constitute a representative sample of traffic between the specified source and destination ranges.

3. In replying to a packet identified as a secure traceroute packet, a node sends some agreed-upon identifying marker for the packet back to the tracing node, to “prove” that the packet has been received. Alternatively, some accumulated value based on multiple received packets may be returned. For example, if the tracing host can place data on the traceroute packets, then it can use a threshold secret-sharing scheme [12] to send a secret that can only be extracted when some minimum number of shares (and thus packets) has been received. (In a threshold secret-sharing scheme, a secret is “divided up” into a number of shares such that any subset of the shares of a certain “threshold” size suffices to reconstruct the secret, but any subset smaller than the threshold reveals no information about the secret. Threshold secret-sharing is an efficient operation, comparable in cost to symmetric-key cryptography.) By inserting one share per traceroute packet, and measuring how many shares must be sent before a reply is received, the tracing host can estimate what fraction of traceroute packets are getting through.

4. In addition to packet-identifying information, the secure traceroute reply contains a strongly secure Message Authentication Code (MAC) that ensures its authentic origin. The MAC is based on the original key shared between the tracing node and expected next node, and covers the entire response, including the address of the node to which the expected next node forwarded the traceroute packet, based on its destination. This new node becomes the “new” expected next node for the next step of the traceroute.

This iterative process produces, as does a normal traceroute, one of two possible results: either a complete route is determined, or a faulty link is found, such that one end of the link claims to be sending packets through the link, but the other end claims not to be receiving them (or simply doesn’t respond to the traceroute). However, the identification of this link is much more reliable than for ordinary traceroute, since return packets are (with greater or lesser certainty, as described below) established to be coming from the correct node, and the use of normal packets in place of identifiable traceroute packets ensures that the route (whether functioning or faulty) accurately represents the route taken by normal packets.

Because secure traceroute is more expensive than the normal variety, it may make sense to limit its use whenever possible. One way to do so is to initiate it starting at a point on the route just before where the preceding nor-

mal traceroute process indicates that the problem appears; thus, if that insecure traceroute turns out to be accurate, then the secure traceroute will demonstrate said accuracy at a relatively small cost. For example, in the (probably very common) case that the normal traceroute indicates that the path functions well until the destination host (and hence, that the problem is probably with the destination host rather than the route), then a secure traceroute can be initiated starting close to the destination host (if not at the destination host itself) on the presumed route. If the secure traceroute results in correct responses where it is initiated, then it can be assumed that the “skipped-over” portion of the route generated by the standard traceroute is accurate (or, in any event, that the “skipped-over” portion is not causing the problem).

IV. AUTHENTICATING SECURE TRACEROUTE

The above protocol assumes that a secure (that is, encrypted, authenticated) key exchange can be performed between the tracing host and the expected next host. Ideally, a public-key infrastructure (PKI) would be available to allow such key exchange to occur using standard protocols (e.g., IPSEC [6]). Such a PKI for infrastructure operators (as opposed to end users) is not unthinkable, as the widely deployed “SSL” PKI for Web servers demonstrates; however, it cannot necessarily be assumed. In its absence, various ad hoc mechanisms can be used. For example, PGP-style “Web of trust” techniques [16], [2] can be used to propagate routers’ public keys from node to (trusting) node; using Web- or email-based protocols, nodes can distribute public keys of nodes they have established confidence in the sources of, and receive such keys in turn from others whose judgments they trust. Similarly, certain widely trusted hosts could voluntarily act as “key servers”, collecting and verifying public keys of nodes and offering them for authenticated download to nodes that trust the key server. Finally, the redundancy of the network can be used to attempt to determine the valid public keys of other hosts at authentication time. By requesting the public key of a host multiple times, via multiple paths—possibly with the help of a set of widely distributed, trusted routing intermediaries (an overlay network of sorts)—a tracing host can increase the likelihood that the public key being returned has not been tampered with en route by a malicious host. Once the expected next host’s public key has been obtained, a (one-way) authenticated key exchange is easy, and the secure traceroute can proceed.

V. USING SECURE TRACEROUTE

We propose a general five-stage process for using secure traceroute to detect, identify and mitigate routing prob-

lems, consisting of the following steps:

1. **“Complaint”:** If an end host detects an end-to-end performance problem to a particular destination, it sets a “complaint bit” in its subsequent traffic to that destination, indicating that a problem is occurring. Complaints emanating from sources found to be generating false complaints are discounted or ignored. (Obviously, this mechanism is vulnerable to source address spoofing; we assume some other solution to the spoofing problem—ingress filtering, traceback, authentication—has been implemented.)
2. **Complaint Evaluation:** If a router’s “complaint level” get high enough—say, if most or all traffic destined for a particular set of addresses complains—then the router may choose to investigate it. Note that a sufficiently severe and sustained congestion problem is indistinguishable from a routing problem, and can be treated as such. In principle, of course, an end host could investigate its own complaints. However, it would be best for a router that is as far downstream as possible (i.e., closest to the problem) to investigate, since its resolution of the problem (say, by rerouting traffic) is likely to benefit the maximum number of sources. We propose an adaptive approach where each router waits for a random interval based how far downstream it thinks it is (say, based on the TTL value of packets it receives) before initiating the investigation.
3. **Normal Traceroute:** The first step in the investigation is to perform a traceroute to attempt to identify the offending node or link in the downstream path. The (possibly partial) path returned by the traceroute may be completely misleading; a malicious router along the route could easily intercept and respond to traceroute packets so as to suggest an arbitrary failed or successful subsequent path. However, the path returned by the traceroute may be useful in locating a faulty node in the presumably more common cases of router misconfiguration or failure, or end-host failure. So this information is used as the starting point for the secure traceroute step described next.
4. **Secure Traceroute:** To verify the results of the traceroute, the investigating router then performs a secure traceroute, described in Section III. Since secure traceroute is a more “heavyweight” protocol, it is initially only performed to confirm the results of the standard traceroute. That is, if the normal traceroute was successful, then secure traceroute is used to check that packets are in fact getting to the destination node; if the normal traceroute terminated prematurely, then a secure traceroute is initiated starting with the successful node closest to the point of failure. Thus, secure traceroute is a relatively cheap procedure if the normal traceroute is in fact valid. However, if this procedure reveals that the normal traceroute was misleading,

then secure traceroute is performed starting with the very next downstream hop, all the way to the destination.

If secure traceroute is supported only by a subset of the routers in the network, we proceed as follows: for each router on the path reported by the normal traceroute, a secure traceroute step is attempted. If it succeeds for a particular router, we deduce that the path up to that router is good; otherwise, we simply ignore that router. The subset of secure traceroute-capable routers thus divides the end-to-end path into a sequence of subsegments. Eventually, the secure traceroute will have identified a subsegment (possibly the last one) that contains a faulty link. Hence, even when only incrementally deployed, secure traceroute still provides partial information about the location of faulty links.

Note that the path information used here is obtained from a normal traceroute and hence is not authenticated; it need not be, since is verified by the secure traceroute. A bad link found by the secure traceroute is in this case either a faulty link on the true end-to-end path, or a point at which the normal traceroute was led astray.

5. Problem Correction: A Secure traceroute can at best identify a faulty link on a route—a link that one node claims to be sending packets through, while the node on the other end claims not to be receiving them. The router that determines such a faulty link can try to route around it; notify downstream routers of the problem, expecting them to make the appropriate routing adjustments; or even pursue the matter through human intervention (such as contacting the administrator of the suspected offending router). We will not examine possible correction strategies in detail here.

VI. ROUTING ASYMMETRY

Internet routing is, in general, asymmetric. The path from node A to node B may be different from that from B to A. This asymmetry can create two problems. First, an end host cannot be sure whether its inability to communicate with a peer host is the result of a network problem in one direction, the opposite direction, or both. Second, the same ambiguity can also affect a node that is performing a secure traceroute. We discuss both these issues in turn.

A. Impact on the End-host Complaint Process

Consider an end host A that is trying to communicate with end-host B. If A does not receive any response (e.g., TCP ACKs) to the packets it has sent to B, A cannot be sure whether there is a network problem in the A→B direction, in the B→A direction, or in both directions. The question then becomes when end host A should start “complaining” by setting the appropriate bit in its packets.

If A receives a steady stream of packets from B but none that indicates that A’s packets have been received by B, then A can be quite sure that the problem is in the A→B direction. Certain applications, such as conferencing, may generate a steady, bidirectional flow of traffic that enables such disambiguation. Certain protocols as well, such as TCP, generate sustained traffic in at least one direction (e.g., TCP retransmissions). In cases where no such traffic is normally generated, we propose that peer hosts that are in the midst of communicating with each other transmit “keep-alive” packets (at a low frequency) during times when they do not have regular packets to send. Receipt of traffic (with little or no loss) from host B would indicate to A that the problem is in the A→B direction. A can then start setting the “complaint bit” in its packet to B. On the other hand, if the problem is only in the B→A direction, then B would hear A’s traffic, deduce that the problem is in the B→A direction, and initiate the complaint process.

There are, however, two problem cases: (a) there may be a failure in both the A→B and B→A directions, or (b) the network failure may precede any communication between A and B, preventing the hosts from exchanging any traffic at all. In both cases, since neither A nor B may receive traffic from its peer, neither would be in a position to determine definitively the direction of failure. In such cases, the deadlock can be broken by having the host(s) initiate the complaint process anyway, after having waited for a random extra duration to give its peer the opportunity to initiate the complaint process and possibly resolve the problem.

B. Impact on Secure Traceroute

Asymmetry in network routing can make it difficult for an investigating router, R, to check whether a downstream router, D, is in fact receiving packets. First, even if D is receiving all packets forwarded by R (and hence there is no network problem in the R→D direction), the two routers may not be able to establish a secure communication channel simply because of a network problem in the D→R direction. Second, even if a secure channel has been established, R may not receive the appropriate response from D due to a (new) network problem in the D→R direction. Thus if R’s secure traceroute attempt does not elicit the appropriate response from D, R cannot be sure whether there is a problem in the R→D direction or in the D→R direction.

The solution we suggest is as follows: The investigating router, R, first initiates the secure traceroute process as described earlier. If it does not receive the appropriate response from a certain downstream router, D, router R repeats the secure traceroute process with one difference: it

includes the reverse route that D could use to communicate back to R using a mechanism such as IP source routing. The reverse route consists of the sequence of intermediate routers along the path from R to D in reverse order. (Note that due to routing asymmetry, the reverse route may not be the same as the route from D to R.) The underlying assumption is that if in fact there is no network problem in the R→D direction, it is quite likely (modulo the presence of unidirectional links) that D will be able to communicate back to R via the reverse route.

In the worst case, the inability of D to communicate back to R would just mean that R would incorrectly deduce that the problem is at or around D and would proceed with unnecessary rerouting or other corrective action, an undesirable but hardly disastrous outcome.

VII. ATTACKS

There are a number of potential attacks against the approach presented here; we outline a few below, along with some potential countermeasures.

1. Because the most frequent cause of failed connections will be unresponsive end hosts—a problem which cannot be fixed by routing adjustments—a malicious router can avoid detection via secure traceroute by simulating a “dead” end host, simply by advertising a (non-responsive) direct link to the targeted end host. If the claimed last-hop router is very far away from the targeted end host, then routing metrics (such as hop counts) can suggest a problem; also, an attempt to find an alternate route should yield one that avoids the offending router. On the other hand, if the malicious router is very close to the targeted end host, then these measures are likely to be less successful; of course, in the worst case, where the misbehaving router is really the (sole) last-hop router, then it will obviously be impossible to distinguish its “blackholing” activity from a truly dead end host.

2. A malicious router may adjust its disruptive behavior so as to avoid detection. For example, it may confine its attacks to periods of time where it does not detect any secure traceroute initiation attempts (i.e., key exchange packets from upstream routers). A simple solution is to give occasional indications of traceroute activity (such as sending key exchange packets—whether real or bogus) whenever there is any hint of a problem. Since the malicious router cannot distinguish real secure traceroute attempts from fictitious ones (beyond detecting the presence or absence of key exchanges), the presence of such simulations should ensure that misbehavior occurs either at such times when it can be detected by secure traceroute, or else not at all.

Alternatively, the malicious router may attempt to interfere with the secure traceroute by selectively blackholing

the packets used in the key exchange phase, so as to give the impression that a router further downstream is not accepting key exchanges (and hence either malfunctioning or malicious). This attack cannot be used by a single misbehaving router to frame a router further downstream: if the misbehavior affects normal traffic, then the secure traceroute will correctly detect a misbehaving link when the (honest) router immediately downstream of the adversary on the path reports the anomalous traffic pattern. However, two misbehaving routers could collude to frame a router between them on a path; the downstream confederate disrupts traffic, while the upstream one disrupts key exchanges to the victim router so as to implicate it. A simple countermeasure to this attack (if multiple colluding routers are deemed a threat, and if redundant routes are not being used to effect the key exchange) is to use “onion routing”-style encryption of key exchange messages [14]. Since each step of the traceroute involves a key exchange, the key exchange traffic can be encrypted hop by hop, so that each router along the route does not know the final destination of the message (and therefore cannot consistently frame a single router).

3. If an attacker can create numerous fictitious nodes in the network, then both identifying a bad link and attempting to route around it can become much more difficult and time-consuming. A single attacker on a path, for instance, could divert a secure traceroute by returning a bogus next-hop router. This may still remain undiscovered if the bogus router is either a confederate of the attacker or the attacker itself under the garb of a different address (i.e., a “dummy” node). In fact, the attacker could lead the secure traceroute into a thicket of bogus routers before it peters out. However, the secure traceroute will eventually identify a bad link, at least one end of which is the attacker or its confederate. Thus, this link deserves to be avoided. There may still be other misbehaving nodes in the path, but persistent application of a succession of secure traceroutes can eliminate them, one by one, from the path until they have all been purged. Hence, if adding confederates or dummy nodes to the network is sufficiently costly, or if owners or administrators of misbehaving nodes are investigated and penalized sufficiently harshly if found to be guilty, then the application of secure traceroute would still help identify misbehaving nodes, so that they can be eliminated and/or punished.

4. Finally, the need for computationally expensive key exchanges creates an opportunity for powerful denial-of-service attacks using bogus key exchange messages that require significant computational resources to detect and discard. A simple solution is just to throttle the number of secure traceroute key exchanges honored to keep it below

an acceptable threshold. This raises the possibility of combining malicious routing with denial-of-service attacks to foil secure traceroute attempts. One possible solution to this problem is to respond to a key exchange message with a “client puzzle” (as in [1], [5]). Such puzzles are easy for the responding router to generate and check, without maintaining state; the requesting router (or the attacker), in order to have its traceroute request attended to, would have to solve the puzzle—which would require at least the same order of magnitude of computation as the responding router has to perform in order to handle the secure traceroute—and return the solution along with a resend of its key exchange message.

Of course, the attacker could always simply muster the computational resources to mount the attack (say, by harnessing thousands of hacked “zombie” computers to the task). But anyone with that level of resources is likely able to mount a more conventional denial-of-service attack against his intended targets in any event—probably without ever needing to subvert the routing system.

VIII. RELATED WORK

There have been several pieces of work on making Internet routing protocols robust against attacks by malicious entities. Perlman [9], [10] presents an approach for “sabotage-proof” routing. The key idea is to use “robust flooding” to distribute link-state packets (LSPs) and the public keys of all nodes throughout the network. Robust data routing is then accomplished by having end hosts construct digitally signed source routes using the link-state information they have gathered. There are a couple of issues that limit the practicality of this approach. First, flooding LSP information on a global scale is likely to be infeasible; indeed this is the rationale for using BGP, a path-vector protocol, for inter-domain routing. Second, allowing each end host to do source routing severely weakens the ability of ISPs to engineer traffic in their networks.

Other proposals [7], [13] have considered less disruptive approaches to securing the routing protocol. In particular, Secure BGP (S-BGP) [7] proposes using public key infrastructures (PKIs) and IPsec to enable a BGP speaker to validate the authenticity and data integrity of BGP UPDATE messages that it receives and to verify the identity and authorization of the senders. As noted in Section I, S-BGP could impose an unacceptable overhead, and more importantly does not offer protection against a misconfigured or failed router that is authorized to advertise routes for an address prefix but fails to deliver packets anyway.

There has been considerable interest recently in securing routing on mobile ad hoc networks. In [8], a “watchdog” technique is proposed to enable a node to check

that a neighboring node did in fact forward a packet onward without tampering with it. This technique makes the strong assumption that nodes can hear the onward transmissions of their neighbors, something that may not be true even in wireless ad hoc networks (for instance, due to directional antennae). SEAD [3] focusses on a lightweight scheme to enable nodes to authenticate routing updates from other nodes for the specific case of a distance-vector routing protocol. As with S-BGP, authentication does not solve the problem of a faulty node that fails to forward packets. [4] proposes a self-organized PKI suitable for mobile ad hoc networks. In the absence of a centralized PKI, we could use a similar approach to support secure traceroute.

Finally, recent work on IP traceback (e.g., [11]) tries to solve a different problem related to the one addressed by secure traceroute. The goal of IP traceback is to determine which routers a specified subset of traffic (typically the “attack traffic” during a DDoS attack) traverses. In IP traceback, the information provided by routers is trusted. Secure traceroute, on the other hand, is used to determine whether traffic did in fact traverse a router.

IX. CONCLUSIONS

In this paper, we have proposed an approach to robust routing in which routers, assisted by end hosts, adaptively detect poorly performing routes that appear suspicious, and use a *secure traceroute* protocol to attempt to detect an offending router. This approach complements efforts that focus on securing the routing protocol itself. We view secure traceroute as a general technique with wide applicability, and are presently investigating it in the context of peer-to-peer networks.

REFERENCES

- [1] C. Dwork and M. Naor. “Pricing Via Processing or Combatting Junk Mail”, In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO ’92*, volume 740 of *Lecture Notes in Computer Science*, pages 139–147, 16–20 August 1992. Springer-Verlag, 1993.
- [2] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas and T. Ylonen. “SPKI Certificate Theory”, RFC 2693, September 1999.
- [3] Y. C. Hu, D. B. Johnson, and A. Perrig. “SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks”, *IEEE WMCSA 2002*, June 2002.
- [4] J. P. Hubaux, L. Buttyan, and S. Capkun. “The Quest for Security in Mobile Ad Hoc Networks”, *ACM MobiHoc 2001*, October 2001.
- [5] A. Juels and J. Brainard. “Client Puzzles: A Cryptographic Defense Against Connection Depletion Attacks”, *NDSS ’99 (Networks and Distributed Security Systems)*, February 1999.
- [6] S. Kent and R. Atkinson. “Security Architecture for the Internet Protocol”, RFC 2401, November 1998.
- [7] S. Kent, C. Lynn, and K. Seo. “Secure Border Gateway Protocol (Secure-BGP)”, *IEEE Journal on Selected Areas in Communications*, Vol. 18, No. 4, pp. 582–592, April 2000.
- [8] S. Marti, T. J. Giuli, K. Lai, and M. Baker. “Mitigating Routing Misbehavior in Mobile Ad Hoc Networks”, *ACM Mobicom 2000*, August 2000.
- [9] R. Perlman. “Network Layer Protocols with Byzantine Robustness”, Ph.D. Thesis, Department of Electrical Engineering and Computer Science, MIT, August 1988.
- [10] R. Perlman, “Interconnections: Bridges, Routers, Switches, and Internetworking Protocols”, *Addison-Wesley Professional Computing Series*, Second edition, 1999.
- [11] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. “Practical Network Support for IP Traceback”, *ACM SIGCOMM 2000*, August 2000.
- [12] A. Shamir. “How to share a secret.” *Communications of the ACM*, 22:612–613, 1979.
- [13] B. R. Smith and J. J. Garcia-Luna-Aceves. “Securing the Border Gateway Routing Protocol”, *Global Internet 1996*, November 1996.
- [14] P.F. Syverson, G. Tsudik, M. G. Reed, and C. E. Landwehr. “Towards an Analysis of Onion Routing Security”. In H. Federrath, editor, *Designing Privacy Enhancing Technologies*, vol. 2009 of LNCS, pp. 96–114. Springer-Verlag, 2001.
- [15] The Routing Arbiter Project, <http://www.isi.edu/div/ra/>
- [16] P. Zimmermann, “The Official PGP User’s Guide”, MIT Press, 1995.