# Securing Data Through Avoidance Routing

Erik Kline
Laboratory for Advanced Systems Research
UCLA Computer Science Department
405 Hilgard Avenue
Los Angeles, California, 90095
icebeast@cs.ucla.edu

Peter Reiher
Laboratory for Advanced Systems Research
UCLA Computer Science Department
405 Hilgard Avenue
Los Angeles, California, 90095
reiher@cs.ucla.edu

## ABSTRACT

As threats on the Internet become increasingly sophisticated, we now recognize the value in controlling the routing of data in a manner that ensures security. However, few technical means for achieving this goal exist. In this paper we propose and design a system that allows users to specify regions of the Internet they wish their data to avoid. Using our system, data will either arrive at the destination along a path that avoids the specified regions, or no avoiding path exists. Beyond the design, we discuss the deployment, performance and security issues of this system, along with alternative approaches that could be used.

## Categories and Subject Descriptors

C.2.0 [**General**]: Security and protection; C.2.2 [**Network Protocols**]: Routing Protocols

## General Terms

Security

## Keywords

Avoidance Routing, BGP, DFS, Security Properties

## 1. INTRODUCTION

Throughout history, nations, enterprises and individuals have gone to great lengths to protect valuable information. While cryptography is the most commonly used technique, history is rife with cases where one party was able to break the cryptography of another [10][14][31]. It is clear that cryptography is not a perfect solution. A complementary technique is to ensure that sensitive data is not available to an opponent. Reducing your adversary's access to your encrypted data can significantly increase its safety. Conversely, adversaries have historically made effective use of intercepted encrypted data. British intelligence's ability to intercept and decrypt German communications in both World War I and World War II was a great boon to the allied war efforts. The intercepted and decrypted Zimmerman Telegram, for example, allowed the British to influence the United States to enter World War I [34].

The same situation applies to the modern information age. While we can use cryptography in an attempt to protect our information, we have little control over which parties can observe our encrypted packets. Even without decryption, merely observing traffic patterns and volume can provide useful intelligence [29]. Finally, opponents can selectively drop some or all of our traffic, denying service when it may be critical. This clearly shows the need for finer control over the routing of information.

The security value of controlling our Internet routing paths has been recognized [28], but there are currently no feasible methods to facilitate this control. At best, we can determine the route that will be taken, although this is not guaranteed. While source routing [26] would allow one to force a specific route, it is not clear how one would go about determining this route. Further, relatively few Internet routers support source routing [9]. The only current approach is to have service providers and autonomous systems manually set up specific routes via Border Gateway Protocol (BGP) policies. This is generally done for traffic engineering and quality-of-service purposes, but not for security reasons. A practical method for avoiding specific regions of the Internet is needed for those who require better security for their sensitive data.

One important aspect of this problem is geopolitical. Incidents of cyber warfare and cyber espionage continue to increase. These include the Estonian-Russian incident [33], Georgian-Russian incident [21], and possible Chinese espionage efforts [27]. Clearly, various nations would prefer not to route their packets through the territories of their enemies, rightly assuming that those enemies will attempt to use that data for espionage or to selectively deny service. One may also wish to avoid countries for legal reasons. For example, some countries restrict the use of cryptography and may try to prosecute those using it. Other countries, such as Sweden [41], have passed laws allowing them to monitor all cross-border internet traffic. Further, given the prevalence of economic espionage [18][23], some commercial entities may have similar concerns.

In this paper we propose a method which allows senders to specify security properties that they want their data to avoid, with reasonable assurance that their wishes will be satisfied. These properties may include geopolitical location, corporate ownership, router type or manufacturer, and specific ASes. Another possible security property is path attribution, where a path is chosen only if it can or cannot provide a certain level of attribution about the sender and/or receiver. We will show how our method is relatively compatible with existing Internet routing protocols. We will also show that this method will incur low additional routing costs and achieve reasonable scaling. This will be accomplished while still allowing many different parties to use avoidance routing, specifying different sets of properties that they wish to avoid when sending data.

## 2. AVOIDANCE ROUTING

In order to conduct routing that avoids nodes with specific security properties, a system must be able to gather and use that information in routing decisions. We assume that participating routers are already running BGP, and we make use of the information they have already gathered about available routes. However, several additional steps are required. First, the security properties of routers and autonomous systems must be determined. Second, the properties of advertised routes must be disseminated between routers. Finally, using this information, particular packets must be routed to avoid entities with particular security properties. While the system discussed here is designed at the inter-AS layer, a similar intra-AS system is required to correctly route packets.

### 2.1 Determining Security Properties

In order to perform avoidance routing, every AS must know the security properties of its routes. While it may be possible for every router to determine the security properties of every other router, we believe this is infeasible. If this information was easily available, it would make more sense for end hosts to obtain it and use source routing, rather than gather it at each individual router. Further, since ASes use BGP to do their inter-AS routing, they only have information about routes advertised to them, and nothing about alternative routes. Having global knowledge of security properties would only help an AS determine if a route intersects specific properties, but not if an alternative avoiding route exists. Alternative approaches that could make use of this information are discussed in Section 5.

The security properties of routes must be determined in another manner. Obviously, every AS is aware of the security properties of its own routers. While an important problem, we consider the method used by an AS to inform its routers of their security properties to be orthogonal to goal of this paper. The scheme presented in this paper is independent of whether ASes manually configure their routers with their security properties or configure them in a automated manner.

However, we cannot necessarily rely on ASes to be either cooperative or truthful about this information. On the contrary, it may be in the best self-interest of an adversary to lie about this information. Some security properties can be validated by a neighbor or remotely. ASes have a high degree of confidence in the geolocation of their neighbors. Security properties such as corporate ownership can be validated by services such as Whois. The AS path information can be validated using S-BGP [15]. However, other information, such as router type, manufacturer or firmware version may not be easily verifiable. How to validate and trust the information an AS tells its neighbors is discussed in more detail in Section 4.

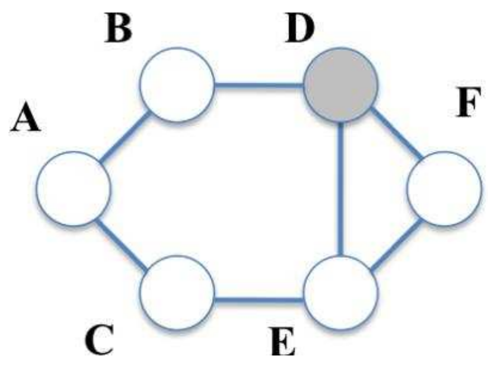

**Figure 1: A simple network topology**

### 2.2 Disseminating Security Information

Once the security properties of each AS have been determined, the ASes must determine the security properties of specific routes. As stated in Section 2.1, it is infeasible for each router to determine the security properties of other routers. Instead, each router shares security information with its neighbors. Routes containing security properties can be built up and shared in the standard distance vector or path vector manner. Using this approach, we can leverage BGP to help pass information about the security properties of each AS in an advertised path, using BGP optional path attributes, for example.

The technique we use is based on the augmented addressing scheme of Oliveira et al. [24]. In their scheme, they augment BGP advertisements such that geolocation is encoded along with AS number. They find that using geolocation information can determine better routes, rather than simply using smallest hop count. They also find that this allows for more aggregation, leading to smaller routing tables.

While their scheme is designed to solve a different problem, we believe the augmented addressing technique can be applied in our system. In our system, addresses are augmented with all the security properties of an AS. We determined the best place to place this data is in BGP optional path attributes, allowing legacy BGP speakers to operate without understanding avoidance routing.

We treat security properties like any other routing dynamic. If a security property changes, new BGP updates must be distributed. We believe security properties will change no faster than routing changes, therefore requiring BGP updates for changes seems reasonable. Changing security properties generally requires manual intervention, such as deploying a new router, and thus occurs infrequently. Even properties that can be changed automatically, e.g., upgrading firmware, occur on the order of months, not minutes. Further, many changes in security properties will result in routing changes. For example, if an AS deploys a new router in a different country, this will undoubtedly generate routing changes along with the change in security properties.

Figure 1 shows a simple topology, where the only security property is geopolitical location. In this topology, AS A originates a data flow to an address in AS F. The shaded AS D is located in countries that the sender wishes to avoid. All other ASes are in trusted countries. Clearly, the data flow should be routed through C. Assuming that B and C have both advertised a path to F, and both have included information about the country security property of the next

ASes in the paths, A will know that C is the optimal choice. If we extend our topology to include more ASes between B-D and C-E, as long as the security information is included in each path vector, A can see the entire path and still choose C.

Obviously, this simple topology makes the decision easy. In a more complex topology, there might not be an advertised path that avoids the undesired properties, while an unadvertised path may exist that does. Network conditions may also change, causing an AS to route along a different path than advertised. Thus, the default advertised routes can help select an avoidance route, but cannot guarantee that this route will succeed or another route will fail. These issues must be overcome by the method used to actually route packets.

The list of path security properties is called the *security vector*. The security vector is built up step by step as the path is created, just like the normal BGP path vector. For example, consider the route from A to C to E to F in Figure 1. This route originated when E advertised to C that E has a direct route to F. Since each AS only has one security property in this example, we shall denote the security property of F as *f*, E as *e*, etc. E's advertisement to C then contains the security vector *f*. C then advertises an E-F route to A, including the security vector $e - f$. A would finally store the C-E-F path vector with the security vector $c - e - f$.

It is important to realize that the security vector has a one-to-one correspondence to the path vector. That is, C has the security property *c*, E has the security property *e*, etc. If F had two different security properties (e.g., located in two countries), it would specify both. In this case, the path vector would be C-E-F with security vector $c - e - f_1, f_2$.

Some ASes may wish to participate in more limited forms of avoidance routing. These ASes may be willing to distribute some of their security properties, but not others. We allow this capability by making the security vector a variable length. In this way, an AS can add to the vector only the security properties it wishes to share. If an AS does not provide a specific security property that a user cares about, this AS is treated as untrusted in regard to that property.

## 2.3 Using Security Properties for Routing

To effectively route packets along avoidance routes, the senders must be able to specify which properties they wish to avoid. Further, using this information along with the information disseminated in Section 2.2, routers must be able to determine the appropriate route. Attempting to store all possible avoidance routes to all possible destinations would result in a combinatorial explosion. Knowing that most routes will never be needed, these routes should be generated on demand.

When a host wants to use avoidance routing, it must make a request for the service. This request specifies the destination and the security properties the user wishes to avoid. This request is call an *avoidance request*. Avoidance requests will be discussed in more detail later.

When a router receives an avoidance request, it has several steps to perform. It first searches its known routes for any path to the destination that satisfies the specified security properties. If such a route exists, the router can immediately forward the avoidance request on to the next router. However, if the router does not know of such a route, it must search for one.

**Algorithm 1** The Search Algorithm.

```
 1: Receive avoidance request
 2: Create search table entry
 3: for all Advertisements $a_x$ to destination do
 4:    if Security vector in $a_x \cap$ avoidance criteria then
 5:       Forward along interface for $a_x$
 6:       Delete search table entry
 7:       return success
 8:    end if
 9: end for
10: Heuristically sort advertisements to destination
11: for all Sorted advertisement $sa_x$ do
12:    Forward along interface for $sa_x$
13:    Delete search table entry
14:    if Success returned then
15:       return success
16:    end if
17: end for
18: Delete search table entry
19: return failure
```

The search used is a standard heuristic-based depth-first search (DFS). The router sorts its possible routes to the destination based on interface using a heuristic. It then forwards the avoidance request along the interfaces in order, until one succeeds or all fail. The search algorithm is shown in Algorithm 1.

Success and failure are explicitly indicated by the next router. When a router receives a success message, this indicates that the next router found a path to the destination. The router can now terminate its search and indicate success to the previous router. When success propagates all the way back to the sender, the route has been established. If a router receives a failure message, this indicates that the next router could not find a path to the destination. In this case, the router tries the next interface. If all interfaces are exhausted, the router indicates failure to the previous router.

Some information must be kept during a search. This information must include the source doing the search, the destination, the avoidance criteria, and an ordered list of interfaces yet to try. As the search is conducted, each tested interface is removed from the list. If the list becomes empty, then no interface succeeded and failure is indicated. Finally, the interface on which the search arrived is also stored. This is done to prevent search loops which are possible using DFS. A search entry is shown in Figure 2.

One important caveat is that our system does not consider path-dependent properties. A path-dependent property is any property where the routing decision is dependent on the path used to reach the current router. For example, delay is a path-dependent property. Path-dependent properties greatly increase the complexity of a search by reducing the pruning of already visited nodes. If we arrive at node A from node B and the search fails, this does not mean that if we arrive at node A from node C, the search will fail.

We could possibly handle path-dependent properties by using partial searches. Each time a node receives an avoidance request, it conducts a search regardless of whether it can currently satisfy the path-dependent properties. In this way, the search data is available and usable for pruning if the

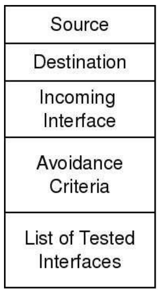

**Figure 2: Search table entry**

search reaches this node from another path. Another possibility is to use a limited crankback approach as discussed by Shin et al. [30], which limits the number of redundant searches based on several factors, such as the number of searches already conducted. However, because of the complexity of the problem and our desire to have a high-speed system, we do not consider path-dependent properties.

What exactly is an avoidance request? An avoidance request can come in three forms. The first form, called *header form*, is an IP option. In header form, the avoidance criteria (the properties to avoid) are specified explicitly as the option. Header form is most useful if the number of security properties is small and the number of packets to send is small.

The second form, called *packet form*, uses an entire IP packet to specify the avoidance criteria. Since IP headers have a maximum length, if you wish to specify a large number of criteria, these must be placed elsewhere. Thus, the packet form was created. The packet form is also essentially a ping, where the receiver will echo the request back to the sender. We use a ping in order to not adversely affect TCP. Assuming the common case of the first real packet being a TCP SYN, that packet's time to traverse the network will set up its TCP round-trip time estimate. Finding the avoidance path may require a heavyweight search operation, which will distort the true round-trip time. The ping sets up the path prior to the SYN being sent, ensuring that the round-trip time is accurate.

The final form is called *transport form*. In this form, a full two-way communication session is established between each hop. The previous hop then sends to the next hop the full list of avoidance criteria. This is the most heavyweight form and should only be used if the number of avoidance criteria is exceedingly large.

## 2.4 Caching

It is clear that conducting a search for every packet is impractical, and with larger avoidance requests it is impossible. Further, every packet in a particular data flow will want to take the same avoidance path. Searching for a path for each packet is inefficient, since the first packet will have already discovered a viable avoidance path. Finally, since avoidance requests may require an entire packet or more, routers must be able to recognize packets that wish to use avoidance routing without needing to know their avoidance criteria every time. To solve this issue, we employ caching. The goal is to ensure that once an avoidance path has been created, all subsequent packets can be forwarded via the cache without specifying avoidance criteria each time. In this way, the search is only conducted once.

Avoidance routing actually employs two different caches. The first cache is called the *forwarding cache*. This cache is organized by an index number and contains only three pieces of information: a destination, a forwarding interface, and the next index number. When a route is successfully discovered, a cache entry is created and stored in the next available cache slot. The index number of this slot is communicated to the previous router in the success message for the avoidance path. That router, in turn, will create a cache entry, storing the index number of the next router. In this way, the source will receive a success message containing the index number of the first router along the path.

When the source uses the service, it places the index number in the IP header as an IP option. The first router receives the packet, and looks up the forwarding interface based on the index number. It then validates that the cache destination and the packet destination match. If so, it changes the current index number in the packet to that of the next router and forwards it along the specified interface. In this way, the avoidance path can be used without every packet containing the explicit avoidance criteria. Since the forwarding cache has small fixed-sized entries, we believe the cache can eventually sit directly on the router using high-speed memory. Therefore, the forwarding cache allows quick high-speed forwarding of packets.

One important optimization in the forwarding cache is that reverse entries are created when an avoidance request arrives. This cache entry is identical to a normal cache entry except it is set up in the reverse direction (to the source), unless routing policy does not permit the router to send packets in that direction. Logically, the packet could not have reached this router from the source unless an avoidance path exists to this router. The next router index number can be carried in the avoidance request.

The second cache is called the *avoidance cache*. This cache is organized by destination and stores the following information: the destination, the forwarding cache index, the avoidance criteria, the next router index number, and the forwarding interface. While it may appear that the avoidance cache makes the forwarding cache redundant, there are two important distinctions. First, avoidance cache entries are larger, and therefore result in less storage of avoidance paths per byte of high-speed memory. The second distinction is more critical. Since avoidance criteria are of variable length, it is impossible to know a priori the size of cache entries. This makes memory layout and management a more difficult problem. For these reasons, we deploy the forwarding cache which uses small fixed-sized entries, making memory layout and access far easier.

It may also appear that the avoidance cache is unnecessary given the forwarding cache. However, the avoidance cache serves several purposes. The first purpose is to handle a packet whose destination does not correspond to the forwarding cache entry based on index number. For example, this may occur if the forwarding cache entry was evicted. In this case, the avoidance cache is checked, the packet can be quickly forwarded, and the forwarding cache can be updated.

The next purpose, and possibly the most important, is to store the avoidance criteria. As stated above, the forwarding cache uses an index system so that individual packets do not have to contain the avoidance criteria. However, this criteria

**Algorithm 2** Using Caching to Forward a Packet.

```
 1: Receive packet using avoidance routing
 2: Get cache index from packet
 3: Get forwarding cache entry $fc_i$
 4: if $fc_i$ exists AND
    $fc_i$ destination = packet destination then
 5:   Forward along specified interface
 6: end if
 7: Get avoidance cache entry $ac_i$
 8: if $ac_i$ exists then
 9:   Forward along specified interface
10:   Create new forwarding cache entry
11: end if
12: Request previous hop to generate new avoidance request
13: run Algorithm 1
```

must be maintained in case there is a routing change. When a routing change occurs along an avoidance path, that path is no longer valid. The last still-connected router will be the first to detect the routing change, and thus the invalid path. At this point, this router will conduct a new search to the destination using the cached avoidance criteria.

This is also important for resurrecting avoidance criteria from a previous router. If a packet using avoidance routing arrives and it does not have a either forwarding cache entry or an avoidance cache entry, the router can request that the previous router conduct a new search. While this router has a valid avoidance path, it also has the avoidance criteria. In this way, it simply generates a new avoidance request and sends it along its cache's interface to the router that does not have any cache entries. This router now has the avoidance criteria and can conduct a new search.

The final purpose of the avoidance cache is that of search optimization. When an avoidance request arrives, the router checks to see if a cache entry to the destination with the specified avoidance criteria already exists. If so, it can use the existing forwarding interface to forward the message along without searching. This can also be used to store failure. In this case, a recent search for a specific avoidance path to a destination failed. This allows us to prevent search loops and prune branches that have already been used by a previous search. How the caches work is shown in some detail in Algorithm 2.

## 2.5 Path Invalidation

While routing in the Internet is more static than routing in other domains, such as ad hoc networks, routes do change. As stated in Section 2.4, the avoidance cache is designed to recover from a routing change by conducting a new search.

Unfortunately, the router will no longer have any information about previously attempted interfaces. Therefore it will once again start the search as if it had never done it before. If this router is able to discover an avoidance path to the destination, it will set up the appropriate cache entries and inform the previous router of success. The previous router will be unaware of the route failure. When it receives the success message, it will discover that no search is being conducted, update its next router index, and then discard the message. However, if the router receives a failure message, the router will also discover that no search is underway, but will not ignore the failure message. It will then check to see if the failure message corresponds to an avoidance cache entry. If this is true, then it determines that the next router on the path had some failure and is no longer usable for this avoidance path. Just like the first router, this router will begin a search. In this way, either a new avoidance path will be discovered or the source will eventually be notified of failure.

One negative effect of this approach is that these new searches will be conducted on existing data flows. This may result in a significant increase in delay, which in turn will result in a reduction of TCP throughput. Although not currently part of the design, a possible optimization would be to explicitly inform the sender or sending AS of an avoidance path invalidation. In this way, the AS could throttle or stop sending until the new path is constructed, at which point normal operation could be resumed.

## 2.6 Partial Deployment

Our system would, of course, benefit from total deployment. However, since total deployment is not likely to happen, the system can work in partial deployment, provided that deployment is contiguous. ASes that do not deploy the system must be recognized and treated as untrustworthy. Further, it may be possible to support non-contiguous partial deployment if a secure method of transit between one island of deployment to another is used. For example, if an anonymity network is deployed between islands, this can be used to protect data transitting between them. However, these methods are generally easily detectable, and may be attacked as away to cut off one island from another. For routing performed on behalf of nations, mandating sufficient partial deployment for their security seems realistic. ASes may also deploy the system as a value-added service.

## 3. PERFORMANCE CONCERNS

While there might seem to be major performance issues, closer examination shows that the avoidance routing system could be relatively cheap to use and maintain. There are three main cost issues that we analyze:

- Cost to perform the search
- Additional overhead for ongoing packet delivery
- Additional space and computational overhead for participating routers

## 3.1 Search Cost

The search algorithm's cost is best expressed in its time complexity. This cost can be determined by the total nodes traversed to find the path. Since our search algorithm reduces to depth-first search (DFS), we are bounded by the well-known complexities of DFS. DFS may result in infinite time complexity if loops exist in a graph. In general, this problem is overcome by marking or remembering which nodes have already been visited. Our search table entries and negative cache entries act as a form of marking. The search complexity is therefore bounded by the worst-case DFS complexity of $O(|V| + |E|)$, where V is the set of nodes in the graph and E is the set of edges between the nodes. This is the worst case where either no path exists or the last possible path searched is the correct one.

With total deployment and no optimization, the nodes in our graph are the routing ASes of the Internet and the edges are the links connecting them. Therefore, in the worst case, every Internet routing AS and link will be traversed. However, our search algorithm reduces complexity by using partial knowledge of the graph's topology. We do not explore paths that cannot possibly reach the destination, reducing our search space to only the paths from the source to the destination. Furthermore, we prune paths that cannot possibly avoid our target properties, including ASes with those properties. Our complexity is still bounded by $O(|V|+|E|)$, but our V is the set of nodes that lie on a possible avoidance path from the source to the destination, and E is the set of edges between the nodes in V. Finally, the next-hop selection heuristic may further reduce complexity.

## 3.2 Forwarding Cost

We assume that the vast majority of all packets forwarded using avoidance routing will find their forwarding information in the forwarding cache. Any packet not using the service will be forwarded normally, and should not experience any additional delay. With suitable indexing, the forwarding cache should be able to service forwarding requests as quickly as a standard forwarding table.

The system does change information within the packet header, i.e., the index number. This change will incur additional overhead per packet forwarded using avoidance routing. However, routers already change the TTL and checksum fields in every packet they forward. Further, an IP spoofing defense called BASE [17] which uses a marking scheme similar to ours shows negligible additional overhead.

## 3.3 Router Overhead

Router overhead is expressed in terms of both additional storage and additional computation. In terms of storage, the router must store cache entries, the search table, and route advertisements. The router makes two cache entries for each avoidance path it sets up, one for the forward path and another for the return path. Both cache entries are the same size and contain destination IP prefix (32 bits), interface (8 bits), and next router index (32 bits). This means that every avoidance path requires 144 bits.

The size of the both the search table and avoidance cache is important. However, this value is unknown since both of these structures store the avoidance criteria, which are of variable length. However, these structures do not need to be stored on the router, but instead can be placed on the route processor. Considering that the Cisco CSR-16-RP route processor has 4 GB of RAM [42], this seems quite reasonable.

The other important space parameter is advertisement storage. Our updates will be larger than standard BGP updates, as the security properties must be augmented to the updates. Similar to the search table and avoidance cache, the exact cost of this augmentation is unknown.

The computational cost is also important. If a path already exists, this cost is negligible, since it is a simple cache look-up, which, like all table look-ups, can be highly optimized for speed. The case where the path does not exist is more complex. First, the router needs to compare the avoidance criteria in the packet to the security criteria in the advertisements. If we assume each set is sorted, the result is an ordered set intersection problem, $O(2N)$. Since this has to be done for $M$ advertisements, it comes out to $O(M \cdot 2N)$. If all advertisements intersect the avoidance criteria, we pick a neighbor based on a heuristic. In general, the neighbors need to be ordered based on the heuristic, which is $O(N \cdot logN)$, although we could attempt to optimize this by performing some of the work in the first advertisement search.

The other computational expense involves cache maintenance. Insertion into the cache should be trivial using a good hash function and indexing ($O(1)$ in the ideal case). If the cache is full, we must pay the penalty of searching the entire cache for expired entries and removing them. A possible optimization is to periodically clean the cache when router load is low. Finally, there is the additional computational cost for receiving an update of a changed/new/withdrawn path. The primary additional cost is attributable to constructing the new advertisement sent to the neighbors. The router must update the security vector specified in the advertisement. Updating this vector requires adding the security properties of this AS to the path's existing security vector.

Not all storage and computational costs need to be paid directly by the router. Many of these costs can be offloaded to route processors, similar to how normal route updates are processed and generated. In principle, only the forwarding cache needs to be on the router. Running the search, storing the search table, storing the avoidance cache, and creating and processing advertisements can be done on route processors, allowing the router to continue to process and forward packets as normal.

## 4. TRUST AND SECURITY CONCERNS

Clearly, trust is a fundamental issue of any security based system. Avoidance routing has two forms of trust that have to be considered. The first involves the problem of trusting information that an AS provides. In Section 2.1, we discuss how some information can be validated by the next hop. However, this is only true if multiple ASes are not colluding. For example, let us assume that we only care about geopolitical location. AS X is in France, which we do not trust. AS Y, colluding with AS X, informs us that AS X is actually in Germany. AS Z, who gets the update from AS Y, has no way of validating if AS X is in fact in Germany.

There are several possible solutions. One possible solution is to deploy a signature-based scheme. ASes sign their security properties. As the path is built, the signatures can be validated. To do this in a truly trusted manner, each AS must determine its security properties in some verifiable way. For example, Cisco routers can generate a known signature which can be verified.

Another possible solution is to create a global reputation system for ASes. As ASes continue to send verifiable and accurate security criteria, their reputation increases. If they are known to send inaccurate security criteria, or participate in other nefarious activities, their reputation goes down. In this way, a reputation level may act as another security property. A user may request a path that has a certain minimum reputation level. Further, a trusted third party could broker the reputation level of all ASes. This would be similar to the clearing houses in the financial system. The solution to this problem is an open question.

The other trust model that must be considered concerns the actual avoidance routing search. Many of our security properties may not be homogeneous throughout an AS. For example, an AS may have routers deployed in multiple countries. Should we give a packet to this AS and trust it to internally avoid the routers we do not trust, or should we not? To solve this problem, we propose two different levels of avoidance criteria: strict criteria and loose criteria. Strict criteria means that an AS should be avoided if it does not homogeneously (strictly) meet that criteria. Loose criteria allow an AS to participate in avoidance routing even if it does not homogeneously meet the criteria, but does heterogeneously. For example, AS X deploys routers in France and Belgium. If France is a strict criterion, AS X should not receive the avoidance packet. If France is a loose criterion, AS X can still receive the avoidance packet and it is assumed that AS X will attempt to internally conduct avoidance routing properly, i.e. only using Belgium. However, if AS X is only deployed in France, it should not receive the avoidance packet regardless of criteria level.

There are several other important security concerns that must be addressed for a system like this one to work. First, our avoidance criteria information is passed in BGP updates. We must ensure that our updates are not improperly altered, causing incorrect routing. This issue is less the core problem of S-BGP (whether the advertiser has the right to advertise a path to a destination) [15], and more whether BGP messages have been altered in transit. These issues can be handled by using IPSec between BGP routers. While some BGP speakers may not currently use IPSec, it would not be difficult to begin doing so. Using secure BGP transmission may be a criterion for trusting an AS.

Another security issue pertains to the search messages passed from one AS to another. Like the path information, search messages must be unaltered. This need, therefore, can be alleviated with IPSec. It is also important that routers not participating in the search cannot send false success or failure messages to other routers. Without protection, this flaw would allow one router to disrupt the search of another. However, in order for this to occur, the router must know the source and destination IP addresses and the avoidance criteria of the search. Without this information, any false search message would have no effect. Further, the router knows which interface is currently being searched. If a success or failure message arrives on another interface, it will have reason to believe this message is false.

A third possible security flaw is the potential for denial of service (DoS). It's possible that an adversary could continuously request different avoidance paths in order to constantly cause routers to conduct searches. These bogus requests will also flush caches, resulting in other existing avoidance paths being lost. We deal with this possible attack with admission controls. Only trusted users will be allowed to request the service, and they will be given an allowed rate of service for requests. ASes are allowed to ignore requests from other ASes they do not trust, and can rate-limit their peers in any manner they see fit to use.

## 5. ALTERNATIVE APPROACHES AND RELATED WORK

There are other routing approaches that attempt to avoid untrusted nodes, such as source routing [26]. But the majority of routers are not source-routing capable [9] and many recommend disabling source routing as it presents a different method for IP spoofing [12][45]. Further, source routing is limited to nine hops based on the maximum length of the IP header. Even if we ignore these problems, it is hard to determine which routers to use. One requires knowledge of the security properties of all routers as well as the topology of the network. Accurately determining the topology of the Internet is an extremely difficult problem [25]. By contrast, we disseminate the security properties via BGP updates and use the Internet topology knowledge present at each router to help make routing decisions.

Zlatokrilov and Levy [38][39] attempt to solve this problem by creating a virtual coordinate system using distance vectors. They retrieve distance vectors from specific reference nodes. Using these distance vectors, as well as the known position of the nodes relative to the source and destination, they generate a coordinate system. They use this coordinate system to find nodes maximally distant from the nodes they want to avoid, and source route through them. To perform avoidance routing using their approach, senders would already need to know the security properties of every router. The authors also do not discuss how to use their system in the current Internet. Khuruna et al. [16] use a similar technique in ad hoc networking to find paths that are farthest from the adversarial nodes. This technique suffers similar limitations to Zlatokrilov's, while also incurring a slow convergence time.

Another alternative is to use an overlay network. Overlay networks have been used for a variety of purposes, including multi-cast routing [4][7], content replication [8][40], content distribution [6][19][44], and privacy and anonymity [46]. In general, two types of overlays could be used to accomplish our goal. A generic overlay network could be used, where the overlay nodes are set up to avoid the routers we distrust. This requires the creator of the network to be positive that the paths between nodes do not go through untrusted routers, which may be extremely difficult if the avoidance criteria is sufficiently complex. Setting up an overlay can be costly. Also, unless routing is static, routes between overlay nodes may change, causing data to go through untrustworthy ASes. Furthermore, overlays are more fragile than our scheme. An attacker may only need to initiate a DoS attack on a small number of overlay nodes in order to bring down the network [3].

An anonymity-type overlay network like Tor [46] could be used to provide avoidance routing. These networks use encryption and onion routing [2] to hide the identity of the sender from the receiver or anyone intercepting the communication. In principle, being able to determine who is talking to whom requires compromising the entire network. The problem with these networks is that they can be heavyweight and easy to detect [22]. An adversary country could kill Tor traffic, denying service at a critical moment. This approach also assumes that the attacker cannot break the cryptography, which is an assumption we prefer not to make.

A final alternative approach would be to design a completely new routing protocol. In this case, a link state routing protocol would be optimal, and would be able to support many different types of security parameters, such as country, router type, etc. Each node in the link state graph would specify all of its parameters. When avoidance routing was conducted, any node that did not meet the requirements would be removed from the graph. Then one would simply run Dijkstra's algorithm on the graph to find a path to the destination. While this solution is viable, it relies on being able to get global information about the ASes on the Internet, which we believe is infeasible. Further, it requires the deployment of a new routing protocol, toward which the industry has shown reluctance.

In the ad hoc realm, Yi et al. [36] propose a routing protocol to avoid untrusted nodes. The protocol assigns each node a trust value a priori. Messages are then encrypted based on trust level, where nodes with higher trust levels can decrypt and forward, while nodes with lower trust levels cannot. This technique is not practical for the Internet. Each node would have to have different trust information for different groups of senders, and the cryptography is too expensive for routers. The TARP approach [1] is close to ours, as decisions are made based on route attributes rather than global trust level. TARP avoids nodes based on power consumption or software capabilities, not necessarily security properties. TARP is based on Dynamic Source Routing [13]; instead of all nodes broadcasting a message, only nodes that meet the specified properties will. However, since TARP is built in the ad hoc realm, it has no way of preventing an untrusted node from overhearing a message. Also, TARP relies on each node to truthfully inform its neighbors of its capabilities.

Other ad hoc networking techniques dynamically adjust trust based on observed behavior [11][20][35]. If the trust is too low, you do not route towards these nodes. These techniques are vulnerable to passive attacks (such as interception). Passive attacks are difficult to observe and are one of the core motivations for our work. An attacker can also game this system to gain a high trust before committing his attack.

Depth-first search has been used before to make routing decisions for both quality-of-service (QoS) and wireless routing. Shin et al. [30] use DFS to find routes that meet certain QoS constraints. This work's use of DFS is different than ours, introducing a set of problems that we do not have to deal with. How messages arrive at a node is important for this approach in determining the next hop. In their paper, the search is optimized to solve this problem, while ours is optimized for different purposes. Stojmenovic et al. [32] use DFS to route in wireless networks, using GPS and QoS information to determine the next hop. Their system must rediscover routes each time, since their work is in the MANET area where the routes change quickly. Furthermore, their decisions are based on GPS, which is information we cannot expect to have for Internet routers. Finally, they also make decisions based on QoS information, which has the same limitations as Shin's system.

Finally, avoidance routing is an instance of policy-based routing [5]. Policy-based routing is any routing scheme where per-packet routing decisions are made based on some specific policy. Current policy-based routing mechanisms [43] generally work on parameters currently carried in the packet, such as source or destination. This allows for the enforcement of blacklists but does not provide the dynamics we desire. Our routing decisions are made based on the avoidance criteria specified by the host. Younis and Fahmy [37] explore constraint-based routing, a combination of policy-based routing and quality-of-service routing. However, they do not explore avoidance-based policies or the problem of how to disseminate or enforce such policies across domains.

## 6. CONCLUSION

In this paper we provide an overview of an avoidance routing system that allows users to protect their data by requesting routes that avoid adversarial or untrustworthy ASes based on security properties. We believe that this is an important concern in the current global political and economic climates. Encryption alone cannot guarantee the confidentiality of data. Instead, it should be used in concert with other methods to secure sensitive data. We have discussed, in detail, the design of such a system, and analyzed the relevant performance and security issues. Finally, we evaluated all alternative approaches, and concluded that none can adequately accomplish the goals of our system.

## 7. REFERENCES

[1] L. Abusalah, A. Khokhar, G. BenBrahim, and W. ElHajj. TARP: Trust-Aware Routing Protocol. In *International Wireless Communications and Mobile Computing Conference*, 2006.

[2] R. Anderson. Hiding Routing Information. *Information Hiding*, 1174:137–150, 1996.

[3] B. Awerbuch and C. Scheideler. Toward Scalable and Robust Overlay Networks. In *International Workshop on Peer-To-Peer Systems*, 2007.

[4] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable Application Layer Multicast. In *ACM SIGCOMM*, 2002.

[5] H-W. Braun. Models of Policy Based Routing. RFC 1104, 1989.

[6] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making Gnutella-like P2P Systems Scalable. In *Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2003.

[7] Y. Chu, S. Rao, and H. Zhang. A Case for End System Multicast. In *ACM SIGMETRICS*, 2000.

[8] I. Clarke, T. W. Hong, S. G. Miller, O. Sandberg, and B. Wiley. Protecting Freedom of Information Online with Freenet. *IEEE Internet Computing*, 6(1):40–49, January-February 2002.

[9] R. Govindan and H Tangmunarunkit. Heuristics for Internet Map Discovery. In *IEEE Infocom 2000*, 2000.

[10] P. Gutman. Lessons Learned in Implementing and Deploying Crypto Software. In *Usenix Security Symposium*, 2002.

[11] L. He. A Novel Scheme on Building a Trusted IP Routing Infrastructure. In *International Conference on Networking and Services*, 2006.

[12] D. Hoelzer. The Dangers of Source Routing. http://enclave forensics.com/Blog/files/dbe04629c14a2d07495a38bbf2fc98d9-5.html, 2009.

[13] D. Johnson. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. RFC 4728, 2007.

[14] D. Kahn. *The Codebreakers*. MacMillan, 1967.

[15] S. Kent, C. Lynn, and K. Seo. Secure Border Gateway Protocol (Secure BGP). *IEEE Journal on Selected Areas in Communication*, 18(4), April 2000.

[16] S. Khurana, N. Gupta, and N. Aneja. Minimum Exposed Path to the Attack (MEPA) in Mobile Ad hoc Network (MANET). In *International Conference on Networking*, 2007.

[17] H. Lee, M. Kwon, G. Hasker, and A. Perrig. BASE: An Incrementally Deployable Mechanism for Viable IP

Spoofing Prevention. In *ASIAN ACM Symposium on Information, Computer and Communications Security*, 2007.

[18] J. A. Lewis. Securing Cyberspace for the 44th President. Center for Strategic and International Studies, 2008.

[19] J. Li, G. Popek, and P. Reiher. Resilient Self-Organizing Overlay Networks for Security Update Delivery. *IEEE Journal on Selected Areas in Communications, Special Issue on Service Overlay Networks*, 22(1), January 2004.

[20] Z. Liu, A. Joy, and R. Thompson. A Dynamic Trust Model for Mobile Ad Hoc Networks. In *IEEE International Workshop on Future Trends of Distributed Computing Systems*, 2004.

[21] J. Menn. Expert: Cyber-attacks on Georgia websites tied to mob, Russian government. http://latimesblogs.latimes.com/technology/ 2008/08/experts-debate.html, August 13 2008.

[22] S. Murdoch and G. Danezis. Low-Cost Traffic Analysis of Tor. In *IEEE Symposium on Security and Privacy*, 2005.

[23] US Department of Justice memo. Chinese National Sentenced For Committing Economic Espionage With the Intent to Benefit China Navy Research Center. http://www.cybercrime.gov/mengSent.pdf, June 2008.

[24] R. Oliveira, M. Lad, B. Zhang, and L. Zhang. Geographically Informed Inter-Domain Routing. In *IEEE ICNP*, 2007.

[25] R. Oliveira, D. Pei, W. Willinger, B. Zhang, and L. Zhang. the (in)Completeness of the Observed Internet AS-level Structure. In *IEEE/ACM Transactions on Networking*, 2010.

[26] J. Postel. Internet Protocol. RFC 791, 1981.

[27] R. Rohozinski and R. Deibert, editors. *Tracking GhostNet: Investigating a Cyber Espionage Network*. Information Warfare Monitor, 2009.

[28] F. Schneider. Network Neutrality versus Network Trustworthiness? *IEEE Security and Privacy*, 6(4), July-August 2008.

[29] B. Schneier. *Secrets and Lies*, pages 34–35. John Wiley and Sons, Inc., 2000.

[30] D. Shin, E. K. P. Chong, and H. Siegel. A Multiconstraint QoS Routing Scheme using the Depth-First Search Method with Limited Crankbacks. In *IEEE Workshop on High Performance Switching and Routing*, 2001.

[31] S. Singh. *The Code Book*. Anchor Books, 2000.

[32] I. Stojmenovic, M. Russel, and B. Vukojevic. Depth First Search and Location Based Localized Routing and QoS Routing in Wireless Networks. In *Computers and Informatics*, 2000.

[33] I. Traynor. Russia accused of unleashing cyberwar to disable Estonia. http://www.guardian.co.uk/world/2007/may/17/topstories3.russia, May 17 2007.

[34] B. Tuchman. *The Zimmerman Telegram*. MacMillan, 1966.

[35] R. Yang, Q. Pan, W. Wang, and M. Li. Secure Enhancement Scheme for Routing Protocol in Mobile Ad Hoc Networks. In *IEEE International Conference on Distributed Computing Systems Workshops*, 2005.

[36] S. Yi, P. Naldurg, and R. Kravets. A Security-Aware Routing Protocol for Wireless Ad Hoc Networks. In *ACM Symposium on Mobile Ad Hoc Networking & Computing*, 2001.

[37] O. Younis and S. Fahmy. Constraint-Based Routing in the Internet: Basic Principles and Recent Research. In *IEEE Communications Surveys and Tutorial*, 2003.

[38] H. Zlatokrilov and H. Levy. Navigation in Distance Vector Spaces and Its Use for Node Avoidance Routing. In *IEEE Infocom 2007*, 2007.

[39] H. Zlatokrilov and H. Levy. Area Avoidance Routing in Distance Vector Networks. In *IEEE Infocom 2008*, 2008.

[40] Akamai. http://www.akamai.com, 2008.

[41] BBC News. Sweden approves wiretapping law. http://news.bbc.co.uk/2/hi/europe/7463333.stm, June 19 2008.

[42] Cisco, Inc. http://www.cisco.com/en/US/prod/collateral/routers/ ps5763/ps5862/product_data_sheet09186a008022d5f3.html, 2008.

[43] Configuring Policy-Based Routing. http://www.cisco.com/en/US/ docs/ios/12_0/qos/configuration/guide/qcpolicy.html, 2008.

[44] Gnutella. http://www.gnutella.com, 2008.

[45] Source Address Spoofing. http://technet.microsoft.com/en-us/library/cc723706.aspx, 2008.

[46] Tor. http://www.torproject.org, 2008.