Verifying Return-Addresses: Bounds on the Efficiency of Distributed Packet Filtering

Benjamin Armbruster[*]
Mentor: Cole Smith[†]
Undergraduate Research Assistantship
University of Arizona

May 6, 2002

Abstract

Denial of Service (DoS) attacks have become a great problem on the internet. The proliferation of such attacks is due to the ease with which packets can be sent anonymously. Park and Lee[1] have shown that Distributed Packet Filtering (DPF) holds the most promise of a variety of approaches and with simulation that it could be cost effectively deployed on the internet. As Distributed Packet Filtering is also of theoretical interest, we have shown that the problem of completely securing a network with a given number of filters can be transformed into the vertex-covering problem (which is NP-complete) and vice-versa. We further prove bounds on the security of a network with a given number of filters.

1 Introduction

There are many cases of communications networks where accountability is desired. Accountability generally means that the sender of information can be identified. Two examples of such networks are the mail system to identify for example the source of letters containing anthrax; and the internet to identify origin of intrusion attempts or to identify the origin of Denial of Service attacks where the victim's bandwidth and resources are consumed through bogus requests. We focus mainly on computer networks.

There have been many previous approaches to solving this problem. Some of these methods tailored toward preventing DoS attacks include ingress filtering, probabalistic packet filtering, and distributed packet filtering (DFP). According to Park and Lee[1], DPF holds the most promise of a variety of approaches and with simulation that it could be cost effectively deployed on the internet. Hence, in this paper we examine the theoretical potential of DPF.

[*]Department of Mathematics; email: barmbrus@email.arizona.edu
[†]Department of Systems and Industrial Engineering; email: cole@sie.arizona.edu

In Section 2 we describe the network model and DPF. In Section 3 we show that securing a network is an NP-Complete problem related to the well-known vertex covering problem. In Section 4 we present various metrics of how secure a network is. In Section 5 we solve various network design problems. In Section 6 we conclude and present areas of future research.

2 Model

Let $G = (V, E)$ be a simple, undirected, connected graph with a set of vertices V with $n := |V|$ and edges E. We assume that $n > 2$ to remove future complicating special cases. Let C be the set of distinct pairs of vertices: $C = \{(i, j) \in V^2 : i \neq j\}$. For all $(u, v) \in C$, let $L(u, v)$ be the set of all cycle-free paths in G from u to v. Let R be a function with domain C such that $R(u, v)$ returns the routing policy of the network from u to v. That is, $R(u, v)$ returns a set of paths from u to v that a packet in the network may take. For all $(u, v) \in C$, we must have that $R(u, v) \subseteq L(u, v)$ and $|R(u, v)| \geq 1$, that is, there exists at least one path for each origin-destination pair.

At the most basic level, a path p from u to v of length k is a finite sequence of vertices, $p(0), \ldots, p(k)$ such that $p(0) = u$ and $p(k) = v$, and for every $i \in [0 \ldots k-1]$, $(p(i), p(i+1)) \in E$. For notational convenience we define two auxiliary functions, $\mathcal{E}$ and $\mathcal{V}$, which act on paths and return a set of directed edges and vertices, respectively. $\mathcal{E}$ is defined such that $(a, b) \in \mathcal{E}(p)$ iff there exists an $i \in [0 \ldots k-1]$ such that $p(i) = a$ and $p(i+1) = b$. Similarly, $a \in \mathcal{V}(p)$ iff there exists an $i \in [0 \ldots k]$ such that $p(i) = a$. Likewise, we may define $\mathcal{E}$ and $\mathcal{V}$ on a set of paths $\mathcal{P}$, such that $(a, b) \in \mathcal{E}(\mathcal{P})$ iff $(a, b) \in \mathcal{E}(p)$ for some $p \in \mathcal{P}$, and $a \in \mathcal{V}(\mathcal{P})$ iff $a \in \mathcal{V}(p)$ for some $p \in \mathcal{P}$.

A Distributed Packet Filtering system involves placing filters on various nodes of the network. These nodes perform various checks on packets (of information) flowing through them. Let $F \subseteq V$ be the set of network nodes with filters, with $f := |F|$. To lend intuition as to how filters monitor packets, we will now define some relevant properties of a packet. Our procedures will utilize information about from where a packet came (on its previous arc), to where it is headed, and what its return address is. We thus characterize $P = \{(o, s, d) : o \in V \bigwedge (s, d) \in C\}$ as the set of packets that can be sent, where o is the puported origin (the return address), s is the (true) sender of the packet, and d is the destination.

To describe the function of packet filters, suppose a packet claiming to come from o destined for d arrives from edge $e = (a, b)$, such that node b has a filter. The filters we consider make a decision as to whether or not not o may equal s based only on the edge on which a packet arrives, its puported origin, and its destination. In particular, we consider networks with either *maximal* or *semi-maximal filters.* Maximal filters make use of all three pieces of information, while semi-maximal filters do not look at the destination. Semi-maximal filters are interesting in practice as the decision tree is reduced to $\mathrm{O}(n)$ instead of $\mathrm{O}(n^2)$ as for maximal filters. Given a set of such filters, let $P_A \subseteq P$ be the subset of those packets that are not dropped by our filters. One property of filters is that no node with a filter may send a packet with an incorrect return address. That is, for all $s \in F$, $(o, s, d) \in P_A$ implies $o = s$.

Consider again a packet $(o, s, d) \in P$, arriving from edge $e = (a, b)$, such that $b \in F$. Then a maximal filter will not drop the packet iff $e \in \mathcal{E}(R(o, d))$, that is there exists a path $p \in R(o, d)$ such that $e \in \mathcal{E}(p)$. From a path viewpoint, $(o, s, d) \in P_A$ iff there exists a path $p \in R(s, d)$ of the packet such that for any edge $(a, b) \in \mathcal{E}(p)$ with $b \in F$, $(a, b) \in \mathcal{E}(R(o, d))$. The only difference between the maximal and semi-maximal filters is that where the maximal filter test $e \in \mathcal{E}(R(o, d))$ the semi-maximal filter tests $e \in \mathcal{E}(\bigcup_{d' \in V} R(o, d'))$, that is, whether the maximal filter holds for some destination. Through the similarity of the definitions of maximal and semi-maximal filters, it can easily be shown that P_A for maximal filters is a subset of P_A for semi-maximal filters. In summary, the semi-maximal filter will not drop the packet $(o, s, d) \in P$ iff there exists a $d' \in V$ such that $e \in \mathcal{E}(R(o, d'))$. Or from a path viewpoint, $(o, s, d) \in P_A$ on a set of semi-maximal filters iff there exists a path $p \in R(s, d)$ such that for every edge $(a, b) \in \mathcal{E}(p)$ with $b \in F$, $(a, b) \in \mathcal{E}(\bigcup_{d' \in V} R(o, d'))$.

3 NP-Completeness and Vertex Covering

Consider the problem of completely securing a network with a limited number of filters. In this section we address if and how this can be done. A network is considered completely secure (or just "secure" as we will say in the sequel) iff for any packet $(o, s, d) \in P_A$, $o = s$. We may also examine the security between a given origin and destination. The communication between a pair of vertices $(u, v) \in C$ is considered secure iff $(o, u, v) \in P_A$ implies $o = u$.

Note that any well-formed packet $(o, o, d) \in P$ will arrive at its destination regardless of the type and placement of the filters. This is proven in the following lemma:

Lemma 1. *Given any assignment of maximal or semi-maximal filters to V, any packet $(o, s, d) \in P$ with $o = s$ is a member of P_A.*

Proof. As P_A for maximal filters is a subset of P_A for semi-maximal filters, it suffices to show this for maximal filters. Consider a packet $(o, s, d) \in P$ with $o = s$, and examine any arbitrary path $p \in R(s, d)$ (guaranteed to exist). For every edge $(a, b) \in \mathcal{E}(p)$ with $b \in F$, we need to show that $(a, b) \in \mathcal{E}(R(o, d))$. Since $(a, b) \in \mathcal{E}(p) \in \mathcal{E}(R(s, d))$ the lemma is proven. □

The general idea of the next definition and theorem is that if an unfiltered node u is on a routing path from o_1 to v, then u can send a packet to v pretending to be o_1. It is said that u *forges* or *spoofs* its address as o_1. We will call u a *corner-node* if there exists no such o_1. Similarly, if the second node of the path, call it o_2, from u to v does not have a filter, then u can send a packet to v pretending to be o_2. This necessarily assumes that the path from u to v is a subset of the path from o_1 to v

and a superset of the path from o_2 to v. This assumption is realistic since the routing table at each node can be of size $\mathrm{O}(n)$ by holding a pairing of destinations and the next node to which the packet should be sent. Otherwise the routing table would be of size $\mathrm{O}(n^2)$ since the subsequent node in the path would depend on the origin too. In the above example, u might conceivably pass a packet from itself to v onto a different node than it would route the packet a o going to v.

Assumption 1. Consider an arbitrary $(u, v) \in C$ and $w \in \mathcal{V}(R(u, v))$ with $w \neq v$. For any path $p \in R(w, v)$ there exists a path $q \in R(u, v)$ such that $\mathcal{E}(p) \subseteq \mathcal{E}(q)$. In the other direction, for any path $q \in R(u, v)$ there exists a path $p \in R(w, v)$ such that $\mathcal{E}(p) \subseteq \mathcal{E}(q)$. Taken together, this means that there exists a bijection from $R(w, v)$ to $\{p \in R(u, v) : w \in \mathcal{V}(p)\}$ with the property that for any $p \in R(w, v)$, $\mathcal{E}(p) \subseteq \mathcal{E}(q)$ or equivalently, there exists a $k \in \mathbb{Z}^{0+}$ such that for all $i \in \mathbb{Z}^{0+}$ for which $p(i)$ and $q(i + k)$ are defined, $p(i) = q(i + k)$.

Definition 1. A node $u \in V$ is a *corner-node* with respect to another node $v \in V$ iff for every node $o \in V$, $o \neq u$, we have that $u \notin \mathcal{V}(R(o, v))$. Furthermore, we simply say that u is a *corner-node* iff u is a corner-node with respect to all nodes $v \in V$ such that $u \neq v$.

Theorem 1. *Consider a pair $(u, v) \in C$ and assume the filters are maximal. The communication between (u, v) is secure iff $u \in F$, or simultaneously u is a corner-node with respect to v and for every path $p \in R(u, v)$, $p(1) \in F$.*

By removing the condition that the corner-node be with respect to v, we have an analogous theorem for semi-maximal filters.

Theorem 2. *Consider a pair $(u, v) \in C$ and assume the filters are semi-maximal. The communication between (u, v) is secure iff $u \in F$ or simultaneously u is a corner-node and for every path $p \in R(u, v)$, $p(1) \in F$.*

Proof. Node $u \in F$ automatically implies secure communication between (u, v). Now suppose that $u \notin F$, and $(o, u, v) \in P_A$. By contradiction suppose that $o \neq u$, First, let us examine the case in which the filters are maximal. Then u is a corner node with respect to v, and for every $p \in R(u, v)$, $p(1) \in F$. Since we know that $p(1) \in F$ for all paths $p \in R(u, v)$, by definition there exists a path $p \in R(u, v)$ such that $(u, p(1)) \in \mathcal{E}(R(o, d))$. Since $(o, u, v) \in P_A$, there exists a path $q \in R(o, d)$ such that $(u, p(1)) \in \mathcal{E}(q)$ and hence $u \in \mathcal{V}(q)$. However, by definition u is not a corner-node with respect to v, which contradicts our initial assumption. Now suppose that the initial assumptions are modified such that the filters are semi-maximal and that u is a corner-node. By the same argument as above, there exists a $v' \in V$ and a path $p \in R(u, v')$ such that $(u, p(1)) \in \mathcal{E}(R(o, d))$, contradicting the assumption that is a corner-node..

To prove the other direction, we demonstrate that communication between nodes u and v is not secure if the consequent is not satisfied by manufacturing a counterexample. The negation of the consequent is that $u \notin F$ and either u is not a corner-node with respect to v (or just a corner-node when dealing with semi-maximal filters) or there exists a path $p \in R(u, v)$ such that $p(1) \notin F$.

Consider the first possibility that $u \notin F$ and there exists a path $p \in R(u,v)$ such that $p(1) \notin F$. If $p(1) = v$ then (u,v) is not secure because for every $o \in V$, $v \neq o$, we have that $(o,u,v) \in P_A$ regardless of whether the filters are maximal or semi-maximal. Choosing an $o \in V$ such that $u, v \neq o$ (this can be done since $n > 2$), we have that the communication from u to v is not secure. Now assume $p(1) \neq v$. then since $(p(1),u,v) \in P_A$ for maximal filters and semi-maximal filters, we again have that communication from u to v is not secure. (Clearly $p(1) \neq u$ since R returns only cycle-free paths.) To show that $(p(1),u,v) \in P_A$, it suffices to show that for any edge $(a,b) \in \mathcal{E}(p)$ with $b \in F$, $(a,b) \in \mathcal{E}(R(p(1),v))$. Suppose p is length K. There exists a path $q \in R(p(1),v)$ such that for any integer $1 \leq k \leq K$, $p(k) = q(k-1)$. So choose a $k \in \{1, 2, \dots k-1\}$ and consider an edge $(p(k), p(k+1)) \in \mathcal{E}(p)$ with $p(k+1) \in F$ ($k = 0$ need not be considered since $p(1) \notin F$). Then $(p(k), p(k+1)) = (q(k-1), q(k)) \in \mathcal{E}(q)$, which is a subset of $\mathcal{E}(R(p(1),v))$.

Now consider the second possibility that both $u \notin F$ and u is not a corner-node with respect to v There must exist an $o \in V$, $o \neq u$, such that a path $q \in R(o,v)$ exists with $u \in \mathcal{V}(q)$. Define k such that $q(k) = u$, and consider any path $p \in R(u,v)$, such that $\mathcal{E}(p) \subseteq \mathcal{E}(q)$. For each edge $(a,b) \in \mathcal{E}(p)$ with $b \in F$, we have that $(a,b) \in \mathcal{E}(p) \subseteq \mathcal{E}(q) \subseteq \mathcal{E}(R(o,v))$. This completes the proof for theorem 1. Furthermore, since u cannot be a corner-node if it is not a corner-node with respect to v, and since semi-maximal filters use less information than do maximal filters, this completes the proof for theorem 2. □

Define $\mathcal{N}$ to be a function on V which returns the set of neighboring vertices. More precisely, for any vertex $u, v \in V$, $v \in \mathcal{N}(u)$ iff there exists a $d \in V$ and a path $p \in R(u,d)$ such that $v = p(1)$. Note that from the definitions of paths, $\mathcal{N}(u)$ is a subset of the neighbors of u on the graph G but it need not be equal. If v is a neighbor to u on the graph but $v \notin \mathcal{N}(u)$ then (u,v) is a redundant edge not needed (to show this one needs to make use of assumption (1)). For simplicity however, we will assume that they are equal.

Assumption 2. For any $u \in V$, $\mathcal{N}(u)$ is the of neighbors to u on G. That is, for any $v \in V$, $v \in \mathcal{N}(u)$ iff $(u,v) \in E$.

Theorem 3. *The network is completely secure iff for all $u \in V$, $u \in F$ when u is not a corner-node, and either $u \in F$ or $\mathcal{N}(u) \subseteq F$.*

Proof. This theorem is the direct result of applying the previous theorem to every possible pair of senders and destinations. Consider a node $u \in V$. Hence the communication from u to any $v \neq u$ must be secure. Now we can apply the above theorems and the definitions of $\mathcal{N}$ and corner-nodes. □

Remark 1. Notice that in the context of complete security maximal and semi-maximal filters are equivalent.

3.1 Decomposing the Network

As proven above, to secure a network completely all vertices which are not corner-nodes, need to be in F. Furthermore, if the neighbors of some corner-nodes are all not corner-nodes, then it is an easy decision not to place a filter on these corner-node as all its neighbors need filters anyway. It is less obvious what decisions should be made when corner-nodes have neighbors that are themselves corner-nodes. In this case, it is not clear on which of these corner-nodes filters must be placed so that the total number of filters placed is a minimum. Note however that this placement of filters only depends on the way in which these corner-nodes are neighbors of one another. Hence, we can think of a solution algorithm as follows.

Algorithm 1 (Decomposition Algorithm).

1. Place a filter on all nodes that are not corner-nodes.
2. Identify all of the disjoint groups of neighboring corner-nodes.
3. Secure each disjoint group separately using the least possible number of filters.

The first two steps have complexity $O(n^3)$ and $O(n^2)$, respectively. If this algorithm completes then the network can be completely secured while if one runs out filters (f is too small) then it cannot be. We will show that there is a one-to-one transformation from the problem of completely securing each group of neighboring corner-nodes with the smallest number of filters and the vertex covering problem.

From here on we will only consider such a group of neighboring corner-nodes in order to determine the complexity of step 3 of the above algorithm. Since no node in the group serves as an intermediate routing node, the routing structure can be represented with a directed graph. An edge from a to b means that a sends packets to b directly. Note that the existence of edges (a, b) and (b, c) does *not* mean that a can send a packet to c via b. In this case, a can only send a packet to c by either sending it via a noncorner-node or through the edge from a to c (if it exists).

Theorem 4. *For a graph of corner-nodes the direction of the edges is irrelevant in determining the minimum number of filters required for complete security.*

Proof. Consider a directed graph G and some edge $e = (a, b)$ in G. There are two cases:

1. Edge e is undirected. Then any solution to G (placement of filter that secures G) is a solution to G with the modification that e is directed.
2. Edge e is directed $a \to b$. Let F be a solution to G. In F, a cannot forge packets. Hence either a has a filter or all its neighbors do. In particular either a has a filter or b does. Now if we consider G', G with the modification that e is undirected, we only need to check whether F allows b to forge packets to its new neighbor, a. This is not possible since we know from F that either b or a has a filter. Hence F is also a solution to G'.

In summary any solution to G does not care about the direction of any particular edge. Therefore the optimal solutions are the same irrespective of the direction of the edges of G. $\square$

3.2 Equivalence to Vertex Covering

Given a graph G with undirected edges, the vertex covering problem tries to minimize the number of vertices that need to be covered such that each edge is incident to at least one covered vertex. A vertex covering problem can easily be turned into our problem. Consider all the vertices to be corner-nodes in our problem and let G' be the graph which represents the routing (as discussed above). Then consider all edges bidirectional and let the fact that a vertex is covered mean that it is in F. Similarly, securing a group of neighboring corner-nodes can be turned into a vertex covering problem. By the above theorem one makes all edges among this group bidirectional. Then one can just reverse the process used in the other direction.

Remark 2. Hence, if corner-nodes are not adjacent to each other, the cheapest way to protect a network completely is to to place filters on a node iff it is not an corner-node.

Remark 3. Picture the routing as a graph where there is an edge between u and v iff u may route to v or v may route to u (in effect a directed subset of G). There the corner-nodes are easy to see! They are preciely those node having a degree of 1. If the cost of securing a network is only proportional to the number of filters placed, the cheapest type of network to secure has as many corner-nodes as possible, realized by a star or tree topology with as great a branching factor as possible.

As the vertex-covering problem is NP-Complete, the problem of completely securing a network with f filters is also NP-Complete.

The question was posed whether the optimal filter placement on any network with shortest path routing is easy. The answer is a simple no. Consider a network with $n-1$ corner-nodes and one node with sufficient edges such that it routes between any other two nodes, which don't communicate directly. Placing filters on such a network is hard as the decision to place a filter on any of the $n-1$ corner-nodes is a vertex-covering problem.

4 Security Metrics

Above we examined how to completely secure a network. Now we turn our attention to defining, “secure-enough”. As noted above, all nodes which are not corner-nodes need to have filters. In practice this is a lot of the network. The question is whether we can, say achieve 90% security with 10% of the filters. Consider a network with n nodes and f filters. There are three main metrics of interest, Θ, $\mathbb{T}$, and $\mathbb{P}$. But before we define these we need to define the functions $\mathbb{S}$, $\mathbb{C}$ on C which return a subset of V. $\mathbb{S}(s,d)$ is the set of return-addresses s can use to send a packet to d. Similarly, $\mathbb{C}(o,d)$ is the set of nodes which could have sent a packet with return-address o to d:

$$\mathbb{S}(s,d) := \{o : (o,s,d) \in P_A\} \tag{1}$$
$$\mathbb{C}(o,d) := \{s : (o,s,d) \in P_A\}\,. \tag{2}$$

Now to the metrics, Θ is the average size of $\mathbb{C}(o,d)$ or equivalently the average size $S-set(s,d)$. $\mathbb{T}$ is a metric of reactive security; after an attack, can you determine who is responsible. $\mathbb{T}$ is the maximum size of $\mathbb{C}(o,d)$. $\mathbb{P}$, on the otherhand, is a metric of proactive security; how many attacks can you not stop from getting to their destination. $\mathbb{P}$ is the maximum size of $\mathbb{S}(s,d)$, a measure of how hard it is for an attacker to find an invalid return-address that reaches his victim. In summary,

$$\Theta := \mathrm{avg}_{(o,d)\in C} |\mathbb{C}(o,d)| = \mathrm{avg}_{(s,d)\in C} |\mathbb{S}(s,d)| \tag{3}$$
$$\mathbb{T} = \max_{(o,d)\in C} |\mathbb{C}(o,d)| \tag{4}$$
$$\mathbb{P} = \max_{(s,d)\in C} |\mathbb{S}(s,d)|\,. \tag{5}$$

Note that if the network is completely secure, Θ, $\mathbb{T}$, and $\mathbb{P}$ all equal 1.

To optimize any of these metrics for an arbitrary network is NP-Complete as it is a superset of the problem of whether one can completely securing a network. And for any particular network, the best option is probably some kind of integer programming or backtracking algorithm. However we will demonstrate tight bounds on these metrics by solving various network design problems: e.g. what is the lowest possible $\mathbb{T}$ how do you achieve it off all possible networks with n nodes and f filters.

5 Network Design Problems

5.1 Optimum for Examples

Before considering general network design problems it is informative to go through the optimal filter placement to completely secure a variety of common graph types. From these examples we later construct optimal solutions to the network design problems.

Tree Consider a network where G has the topology of a tree. For a tree the routing is uniquely determined by the graph structure.

Lemma 2. *If G is a tree, then the network is completely secure iff for any node u with degree greater than one, $u \in F$.*

Hence to completely secure the network, we place filters only the nodes with degree greater than one (the branch nodes of the tree). In particular if the network has a star topology (is a graph with only two layers) then only one filter at the center is needed to secure the whole network. Later we will show that this optimizes various network design problems.

Clique Consider a network where G is a clique and where every packet goes directly from its sender to its destination (i.e., packets are not routed through intermediaries). Then the network is completely secure iff $n-1$ nodes have filters.

Proof. If two nodes do not have a filter then the route connecting them is unprotected and one of the two nodes can spoof the address of the other in a packet to the other. □

This is an example of a possible worst-case.

Loop Consider a network where G has the topology of a loop and where routing is in both directions (as in an ATM network). This is an example of a network with redundant links: if one link fails then the network can route around it. Then the network is completely secure iff all nodes have filters, $F = V$.

Proof. Consider a node without a filter. The it is in the middle of the path from its left neighbor to its right neighbor or vice versa. Hence it needs a filter since otherwise it can spoof a packet to the destination of the path. □

This is another example of a worst-case usage of filters.

5.2 Network Design Problems

A network design problem involves finding the network which minimizes a given cost function subject to some constraints. What kind of costs and constraints are possible? The cost could depend on the number of edges, the specific edges constructed (it costs more to go from LA to Pasadena than to New York), the bandwidth on the edges, the bandwidth through the nodes, the bandwidth through the filters, and the degree of the nodes. There could also be some redundancy constraints on the network to ensure network survivability. Furthermore, some issues can be simplified by viewing them as a constraint instead of a cost; instead calculating a high cost when say bandwidth through a filter exceeds a certain limit, we may just require a priori that the in our solution the bandwidth is below a certain limit.

Cost depends on f, $|E|$ For simplicity we start with a cost function that is a linear combination of the number of edges and filters,

$$\text{Cost} = c_1 e + c_2 f, \tag{6}$$

where e is the number of edges ($|E|$) and c_1 and c_2 are some nonnegative constants.

Theorem 5. *A network where G has a star topology minimizes the cost function in equation 6 by having only one filter at the center.*

Proof. As this is optimal irrespective of the values of c_1 and c_2, this topology must independently minimize e and f. That it is optimal is not surprising since it requires $n-1$ edges and only 1 filter (at the center).

we will first prove that a network with n nodes always needs at least $n-1$ edges (the number of edges needed by the proposed tree/star configuration). This can be proven inductively. For $n=2$, $e \geq 1$ trivially. So assume for $n=k$, $e \geq k-1$. Now if we add another node, we must connect it to the rest, this takes at least one edge. Hence for $n=k+1$, $e \geq k-1+1=k$.

Now, we need to prove that the proposed configuration uses the least number of filters. we will show that one node always needs at least one filter. Take any node in the network, either it is not corner-node and hence has a filter or it is an corner-node and hence either it or all its neighbors have filters. □

Constraining Degree Somehow, making the network a star topology is not feasible in practice, especially with a large number of nodes. This means that the above network design problem is too simplified; we have overlooked some variable which should either create a constraint or appear in the cost function. Superficially, a star topology is unrealistic as the center node has too great a degree. So to produce more realistic results we will constrain the degree of all nodes in the network while keeping the same cost function (6).

Constraint 1. For all $u \in V$, the degree of u in the graph G is less than or equal to k.

The main result is that a tree where nodes have the maximum degree where possible solves this kind of network design problem.

Theorem 6. *A network where G has a tree topology which minimizes depth subject to the constraint* (1) *minimizes the above cost function* (6).

Proof. We will separately prove that the proposed configuration has the least number of edges and the least number of filters. This implies that the cost is minimized no matter what the constants are. We will first prove that a network with n nodes always needs at least $n-1$ edges (the number of edges needed by the proposed tree/star configuration). This was proven in the previous theorem.

Now we will prove that this topology minimizes f. Take an arbitrary graph G with n nodes. There is subset of the edges T that connects all nodes and is a tree. The number of filters needed to secure T is not greater than the number needed to secure G since it is a subset. Now we only need to compare the number of filters that the arbitrary tree T needs to the number that a tree with minimum depth needs. If T does not have a minimum depth then there is a subtree rooted at node i that can be moved up. As this subtree is moved *up*, its new parent already needed a filter. However, its old parent now has one less child nodes and may even be a leaf node. If it is a leaf node, then it doesn't need a filter anymore. In anycase, the number of filters needed does not increase. Hence relocating this subtree may reduce the number of filters needed. When this cannot be done anymore, then the tree has a minimum depth. □

Constraint: Redundancy Another limitation of the above network design problems is that unlike networks such as the internet, the optimal solutions (a star or tree topology) provide zero redundancy: if a link fails (i.e. an edge in G is removed), then the network becomes disconnected.

One can imagine the existence of backup links, or routes, in case another fails for some reason (e.g. it might physically break or it might become congested). If these routes may be used at any time, then obviously they are not special should be treated like any other. However, we can image that these can only be used if certain other routes fail. We are interested in this scenario because we still want to maintain security of the network. These backups can be realized as special routing table entries that are activated when certain other routing table entries fail. One question that needs to be answered is whether the filters know about the failure of certain routes and hence which backup routes may be used. Hence it is surprising that it does not matter: that these special routing table entries should be treated like any other in order to optimally place filters.

The argument is as follows. Consider a network with some backup links. Consider a vertex v in this network. In the case that the normal routes make it not an corner-node, then it needs a filter. Also, if there is any failure that activates backup links that make it not an corner-node then it needs a filter. Hence if the normal and backup routes together make it not an corner-node, then it needs a filter. Now consider one that is an corner-node with respect to both its normal and backup routes. If there is not a filter on v, then of course all of its neighbors from normal routes need to have filters. But any neighbor from resulting from a backup route also needs a filter in case this backup route becomes activated. Hence all its neighbors from normal and backup routes need to have filters (if v does not have one).

Now we will examine the network design problem characterized by the cost function used previously ((6)) and the constraint that all paths are redundant to the failure of any one edge. However as the optimum network structure depends on the constants in the cost function, we will solve an equivalent problem from which one can easily deduce the solution to the original problem. We will only have the cost function depend on the number of edges but with the number of filters being an arbitrary fixed number.

Theorem 7. *We want to find the network with f filters which minimizes the following cost function*

$$Cost = |E| .$$

A network where G is a combination of a loop and a fan where the loop contains f nodes all of which have filters optimally solves this problem with a cost of $2n - f$.

5.2.1 Cost function: how secure is secure?

One implicit constraint in the above network design problems is that the network must be completely secure. However as the metrics presented in the previous section show, security is not a binary property. Hence as we usually do not need the network to be completely secure, only secure enough, we can find more cost effective solutions by adding information about how secure it needs to be to our network design problem, There are multiple equivalent ways to do this. One way is to ask what is the most secure network one can make using a given number of filters. Hence our cost function is one security metrics presented in the previous section, and our a constraint is that we have only f filters. Note that for simplicity we do not consider the number of edges as we did above.

Reactive Security ($\mathbb{T}$) Now consider the network design problem characterized by the following cost function

$$\text{Cost} = \mathbb{T} \tag{7}$$

and the constraint that the number of filters, f, is fixed. What follows are the best and worst case solutions to this network design problem. The worst case is also interesting as it provides a lower limit to the security of a network already in place.

We will start with the worst case as it is the simplest. It turns out that $\mathbb{T}$ is roughly $n-f$. This makes sense since in a badly constructed network the only benefit from a filter on a particular node is that it prevents that particular node from sending a packet with an invalid return address. This is developed in detail in the following theorem.

Theorem 8. *A clique with indirect routing for all pairs of vertices in C maximizes the cost ((7) for any given number of filters. The resulting cost is the following*

$$Cost = \mathbb{T} = \begin{cases} n-1 & \textit{if } f = 0, 1, \\ n-f+1 & \textit{if } 1 < f < n-1, \\ 1 & \textit{if } f = n-1, n. \end{cases} \tag{8}$$

Proof. I claimed that if $f = 0$ then over the set S, $\max C = \min C = n-1$. Proof follows: Consider a network in S. Consider a packet arriving at a node. Since connectivity is complete, any other node could have sent it. Hence $\min C = n-1$. Since the trivial filter prevents the node from spoofing itself, $\max C = n-1$.

Another trivial case I forgot to deal with before, if $f = n$ then over the set S, $\max C = \min C = 1$. Proof follows: Since all nodes have filters, no packet can be spoofed so C=1 for all routes.

Consider a network in S which is a clique with routing being the set of all 3 nnode paths. Consider a packet arriving at a node. If this node is a filter and the puported origin is not a filter then $|C_(s,t)| = (n-1)-(f-1) = n-f$. Now consider the case where the puported origin is also a filter. Then the puported origin could have sent it too. Hence $|C_(s,t)| = 1+(n-2)-(f-2) = n-f+1$. Since $f < n-1$, consider the case where this destination node does not have a filter. If the purported sender does not have a filter, then any other node except those with filter could have sent it. Hence $C_(s,t) = n-1-f$. However if the purported origin is a node with a filter, then the purported origin and all other nodes (there are $n-2$ of those) which are not filters could have sent it. Hence $C_(s,t) = 1 + (n-2) - (f-1) = n - f$. Hence for a clique $C = n - f + 1$.

Now I have to prove that there is no network in S with a greater C. Consider a network in S. Consider a packet arriving at a node. Assume the purported origin is not a filter. Then any other node except those with filters (there are at least $f-1$ of those since the destination node could have a filter) could possibly (depending on the topology of the network) have sent it. Hence $C_(s,t) <= (n-1) - (f-1) = n - f$. Assume the purported origin is a filter. Then the purported origin or any other node except the purported origin, the destination, or other filters could possibly have sent it. Hence if the destination is not a filter $C_(s,t) <= 1 + (n-2) - (f-1) = n - f$ and if the destination is a filter $C_(s,t) <= 1 + (n-2) - (f-2) = n - f + 1$. Hence $C <= n - f + 1$. □

The optimal solution to this problem is attained by having the nodes with filters form a shallow tree (subject to constraints on the maximal degree of any node). Then we let the rest of the nodes form small clusters of roughly equal size, each of which is attached to one node with a filter. So the general idea is that we take the optimal solution for perfect security, a shallow tree, and add then attach the "imperfections". These are spread as evenly as possible as the insecurity originates from communication between members of the same group of nodes with out filters since this communication never passes through a filter. The following theorem makes this precise,

Theorem 9. $\mathbb{T} = \left\lceil \frac{n-f}{(k-2)f+2} \right\rceil$

Proof. I claimed that if $f = 0$ then over the set S, $\max C = \min C = n-1$. Proof follows: Consider a network in S. Consider a packet arriving at a node. Since connectivity is complete, any other node could have sent it. Hence $\min C = n - 1$. Since the trivial filter prevents the node from spoofing itself, $\max C = n - 1$.

Another trivial case I forgot to deal with before, if $f = n$ then over the set S, $\max C = \min C = 1$. Proof follows: Since all nodes have filters, no packet can be spoofed so C=1 for all routes.

Proof for $0 < f < n$: I will first determine characteristics of networks in S with small C. These characteristics will be easy to satisfy (giving existence). I will then show that these characteristics imply $C = \left\lceil \frac{n-f}{(k-2)f+2} \right\rceil$. As these are networks with small C, this implies that this is $\min C$.

For simplicity assume networks have small C if the subgraph of filters forms a tree. Hence the other nodes form subgraphs hanging from various points of this tree. C is obviously smaller if these subgraphs are disjoint. Consider a subgraph K dangling from the tree of filters. Any node in K can spoof the address of any other node in K. But it can't spoof any other address. In particular it cannot spoof the address of a node with a filter. Hence the traceback algorithm is as follows for a packet (s,t): if s has a filter then the packet is genuine. $\mathbb{C}(s,t) = 1$. Else, let K be the subgraph in which s is. Due to the disjointness, K is unique. The sender can be any node in that subgraph (except t if it happens to be in that subgraph). Hence for t not in K, $\mathbb{C}(s,t) = |K|$. Hence $C = \max|K|$. Let m be the number of places where we can connect subgraphs to the tree of filters. Since we want to minimize $\max|K|$, we try to distribute these $n-f$ nodes evenly among the m possible subgraphs. Hence by the pigeon hole principle, $\min C = \max|K| = \lceil (n-f)/m \rceil$. The trick is to find m, the number of possible places to connect subgraphs to the tree of filters. This is constrained by the maximum degree a node may have. The root of these subgraphs must be neighbors of the tree of filters. All but the root of the tree of filters can have $k-1$ children (the root can have k). Hence the number of children is $m = (k-1)(f-1) + k - (f-1)$. This last term $f-1$ is because we were double counting the already existing children: all filters except the root are children of somebody in the tree of filters. Hence $\min C = \left\lceil \frac{n-f}{(k-2)f+2} \right\rceil$. This proof is a bit hand-wavy in the beginning but that can be fixed (the missing 2%). ☐

Note that the cost of this optimal solution is approximately $(n/f - 1)(k-2)^{-1}$. This form emphasizes the various roles n, f, and k play. The n/f factor shows the benefit of smaller clusters of nodes without filters, the more filters you have to attach them too. The inverse relationship to k is the same seen when optimizing for complete security. The higher the branching factor of your tree of filters, the less nodes are just there just to make the tree stay together.

Total Security (Θ) Now we examine a similar situation to the above except that

$$\text{Cost} = \Theta. \tag{9}$$

Starting with the worst case again, it turns out that $\Theta = \mathrm{O}(n^2(n-f))$. That it is between n^3 and n^2 makes sense as $|P|$ exemplifying the worst case is $\mathrm{O}(n^3)$ while $|C|$ exemplifying the optimal case is $\mathrm{O}(n^2)$. In summary this means that this network constraint problem is underconstrained.

Theorem 10. *1. for $f = 0$, for all members of S, $\Theta = (1/2)(n-1)^2 n$. Hence $\Theta = \mathrm{O}(n^3)$.*

2. for $f = n$, for all members of S, $\Theta = (1/2)(n-1)n$. Hence $\Theta = \mathrm{O}(n^2)$.

for $0 < f < n$, over the set S, $\max\Theta = (1/2)(n-f)(n-1)n$.

Proof. Proof of claim 1: As there are no filters and connectivity is complete, for any (s,t) with $s \neq t$, $|\mathbb{C}(s,t)| = n-1$, i.e. all nodes but t. As there are $n(n-1)/2$ possible (s,t), $\Theta = \Theta = (1/2)(n-1)^2 n$.

Proof of claim 2: As there are filters on every node, no node can forge packets. Hence as connectivity is complete, for every (s,t), $\mathbb{C}(s,t) = \{s\}$. Hence $|\mathbb{C}(s,t)| = 1$. As there are $n(n-1)/2$ possible (s,t), $\Theta = (1/2)(n-1)n$

Proof of claim 3: Consider a network in S. It was shown previously that over S, $\max C = n-f$. Hence for all (s,t), $\mathbb{C}(s,t) \leq C \leq n-f$. As there are $n(n-1)/2$ possible (s,t), $\Theta \leq (1/2)(n-f)(n-1)n$. Therefore, I only need to find a network in S such that this maximum is reached. Consider a network in S which is a clique with routing being the set of all three node routes. The intermediate unfiltered node prevents a filter on the receiving node from being able to draw any conclusions. $\square$

The following theorem discusses the optimal solution.

Theorem 11. *1. for $f = 0$, for all members of S, $\Theta = (1/2)(n-1)^2 n$. Hence $\Theta = \mathrm{O}(n^3)$.*

2. for $f = n$, for all members of S, $\Theta = (1/2)(n-1)n$. Hence $\Theta = \mathrm{O}(n^2)$.

for $0 < f < n$, over the set S, $\min \Theta$ is complicated though the most relevant

Proof. Proof of claim 1: As there are no filters and connectivity is complete, for every (s, t) with $s \neq t$, $|\mathbb{C}(s, t)| = n - 1$, i.e. all nodes but t. As there are $n(n-1)/2$ possible (s, t), $\Theta = \Theta = (1/2)(n-1)^2 n$.

Proof of claim 2: As there are filters on every node, no node can forge packets. Hence as connectivity is complete, for every (s, t), $\mathbb{C}(s, t) = \{s\}$. Hence $|\mathbb{C}(s, t)| = 1$. As there are $n(n-1)/2$ possible (s, t), $\Theta = (1/2)(n-1)n$

the basic idea for the interesting case is that the optimal structures for minimizing and maximing C are optimal here too. $\square$

6 Conclusion

Our main result is that securing a network with distributed packet filters is an NP-Complete problem. However, as the network design problems showed it is not hard to design networks which can be secured efficiently. Now we will discuss ideas that have not born fruit yet. There are currently three such ideas, meaningful predictors of cost, heuristic algorithms, and the average cost.

As mentioned before the optimal solutions of the network design problem can be seen as bounds on the cost of securing an arbitrary network. However these bounds are far from tight: it takes $n - 1$ filters to secure a clique and one filter to secure a network with a star topology. This means that the number of nodes alone is a bad predictor of cost. We would like an easily computable characteristic of a network which provides good bounds on the cost.

In this paper we have only noted that optimizing the filter placement on an arbitrary network is computationally hard. The question remains how well a particular heuristic algorithm works. One promising possibility is a greedy algorithm. This algorithm would place filters first on those nodes which have the the most neighbors without filters. Then filters are placed until we run out.

Another interesting question is what the average cost is to secure a network. This is a broad topic as we could examine random networks, or networks with power-law structure. Or we could examine the average effectiveness of the greedy algorithm on a particular class of networks.

Finally, there are interesting extensions to the network design problem considered. Particularly more complex and realistic cost functions should bring out interesting behavior. For example the cost of placing a filter on a node might be proportional to the amount of traffic it needs to filter. With this kind of cost function the solutions considered earlier that involve a tree with filters become suboptimal as a few nodes at the base of the tree filter all the traffic.

References

[1] K. Park and H. Lee. On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets. In Proc. ACM SIGCOMM '01, pp. 15-26, 2001. see http://www.cs.purdue.edu/nsl/recent-pubs.html

[2] M. Weigt and A. K. Hartmann, "Typical solution time for a vertex-covering algorithm on finite-connectivity random graphs", Phys. Rev. Lett. 86, 1658 (2001). see http://www.theorie.physik.uni-goettingen.de/˜weigt/publications.html