**CluB**: A Cluster Based Proactive Method for Mitigating Distributed Denial of Service Attacks

**Abstract**

Distributed Denial of Service (DDoS) attacks are threats not only for the direct targets but also for the core of the network. They are also hard to detect in advance, hence methods to deal with them need to be proactive. By building on earlier work and improving on distribution of control aspects, we propose a proactive method, which is called CluB, to mitigate DDoS attacks; the method balances the effectiveness-overhead trade-off by addressing the issue of granularity of control in the network. CluB can collaborate with different routing policies in the network, including contemporary datagram options. We estimate the effectiveness of the method and also study a set of factors for tuning the granularity of control.

**Keywords**: Cluster-Based, Distributed Denial of Service, Proactive Method, Granularity, Distributed Filtering

1 **Introduction**

Distributed Denial of Service (DDoS) attacks can be so powerful that they can easily deplete the computing resources or bandwidth of the potential targets. Preventing or mitigating DDoS attacks is quite hard, since the key feature of today’s networks is openness —any pair of Internet end points can communicate freely. This leads to the lack of effective network infrastructures for distinguishing and dropping malicious traffics. When the attacker controls millions of zombie machines, it can easily launch a big volume of malicious traffic by having each of the zombie machines contribute only a small volume of packet flow.

This is challenging not only for the target of the attack, but also for the network, as large volumes of illegitimate traffic share the same network resources as legitimate traffic and can furthermore cause congestion phenomena and performance degradation. Although there is large amount of literature on congestion control [5], the methods do not apply well to the DDoS situation, as they are based on legitimate traffic characteristics, or aim at e.g. sharing the resources among traffic classes in a fair way, etc. Considering malicious traffic, we would like ideally to disallow it completely from consuming network resources. This is essentially the motivation of the recent research work on this theme (cf. section 2), focusing on controlling the malicious traffic as close to the attacker(s) as possible.

Ideally, in an imaginary world, if we have a global network monitor which can observe and control every flow between any pair of hosts, DDoS attacks could be mitigated by having this monitor identify every potential DDoS attack and stop the corresponding traffic as soon as possible before it reaches the target. However, it is well-understood that such a centralized counter-measure is practically impossible due to huge implied overhead —it would actually resemble a DoS attack itself! On the other hand, a completely distributed, low-overhead solution, where every entity has only local information, can have limited effect in stopping malicious traffic at some distance from the target. Viewing the problem from this perspective, we observe a trade-off in the achievable protection level of the network and the efficiency/overhead of the protecting method, depending on the granularity of control.

Roughly speaking, to balance this trade-off, we would wish to have some distribution of control, with some sort of distributed authority present for coordination purposes. This observation led us to think of an analogy in real-life; namely the exit and entry control problem between countries in the world. A citizen
of one country needs a passport and the corresponding visa to go to another country. The passport is a permission that the local country allows this person to go abroad and recognizes this person as a good citizen. The visa is the permission from the destination country to allow this person to enter in. If this person will transfer a flight in a third country, a transit permission for passing by this country may be needed.

Inspired by the above idea, we propose CluB: a Cluster-Based architecture against DDoS attacks. In CluB the network consists of a set of clusters (in the Internet, these can be e.g. Autonomous Systems (AS)), or neighborhoods of AS. Packets need permissions to exit, enter, or pass-by different clusters.

Several challenges need to be addressed, including what is the right level of distribution so that this can be feasible and scalable, how the permissions are issued, how the permission-control is carried out, how the permissions will be implemented so that they are hard to be faked by adversaries, and what is the level of protection that can be achieved. We describe and analyze solutions in CluB, addressing the above questions. Briefly, the properties of CluB are that each of the sending permissions is path-independent within the clusters, the packets forwarded through clusters are checked in a distributed way and only some particular routers will keep the states of the valid traffic flows. We also suggest implementation options, involving the appropriate cryptographic tools, and periodically updating components in the architecture as well; i.e. tokens of valid packets and checking entities change periodically. This limits the possibility of a powerful attacker to obtain means to launch direct attacks to the checking entities. In the analysis section we show the guarantees of CluB in filtering malicious traffic, as well as in limiting the attacker’s options and strategies. Furthermore, we study how to choose the size of the clusters, aiming at balancing the granularity-of-control trade-off.

CluB can be applicable also in the Internet context, with the following advantages:

- It treats the DDoS problem in a distributed manner, possibly at the granularity of Autonomous Systems level, which are natural administrative domains.

- Each cluster is responsible for its own traffic and cooperates with other clusters to prevent malicious packets flooding. Each cluster can adopt its own security policy.

- Routing algorithms used in today’s Internet are still applicable in the proposed architecture.

2 Related Work

Solutions proposed to defend DoS attacks can be mainly categorized into reactive solutions and proactive solutions. Generally speaking, reactive solutions usually rely on the targets to detect the attacks, and identify the malicious traffic by the path through which it is forwarded [23, 22, 17, 4, 16]. Proactive solutions aim at mitigating or preventing the DDoS attacks before the latter effect on the performance of targets. Some proposals of proactive defense are based on Secure Overlay Networks [14, 24]. The basic idea is to use a secure overlay network as a proxy for the server that needs to be protected. Communication between clients and the server is via secure overlay nodes, the legitimate traffic is spread and the corresponding service becomes DDoS-resilient. But this mechanism is expensive for deployment. Not every network service can afford getting enough overlay nodes for protection. However, this mechanism is suitable for protecting some very important services.

Capability based mechanisms [2, 25, 26] against DDoS were proposed as an alternative. The essential component in this kind of solutions is a token (capability) which indicates the validity of the messages. Every router along the path may check this capability or some part of it (if it is generated by the routers). Every capability is path-dependent, which implies that if the path is broken, the capability should be regenerated. Also, intermediate routers do not have the right to decide which traffic can pass through them. This implies that malicious hosts can allow each other to send traffic and flood a part of the network. Another important issue is that attackers can flood capability-request packets to the request channel to prevent the legitimate
requests being delivered. The attack against the initial capability requests was referred to as Denial of Capability (DoC) [3]; i.e. capabilities need to be used together with mechanisms against DoC [20, 12].

Other related work includes the methods for traffic control across different organization boundaries using visas in [9] and the Platypus routing architecture [21]. Although the basic metaphor is similar (i.e. use of visa-like permissions), the problems targeted in this work are different and the challenges mentioned in the introduction, which are necessary to be addressed in order to apply the intuitive metaphor into solving the DDoS problem, are not addressed in that work. In particular, the traffic control solution [9] assigns all the controlling tasks to the gateways of the organizations, which could handle the inter-organization traffic, but would make the solution infeasible for a DDoS solution (this could turn the gateways into hot-spots). Furthermore, visas used in the protocol are client-dependent which forces the gateways to keep one visa for each client that will pass through, and this, too, makes the system vulnerable to DoS attacks. Platypus uses delegated-capabilities for packets checking by a set of designated routers (waypoints) along the path, which relaxes the path-dependency of capabilities but is still coherently vulnerable to the DoC problem. Furthermore, the issue of how outbound traffic is controlled and how to deal with the case of attackers sending traffic directly to the target without being checked by the waypoints are not addressed, naturally, as they were not part of the problem addressed in that paper.

3 System Model

The network nodes (routers or end-hosts) are organized into disjoint clusters (e.g. autonomous systems (ASes) in the Internet). All end entities (or hosts) are associated with clusters where they are located. Each cluster $C_i$ has $n_i$ nodes, with $\sum_i n_i = N$, the number of nodes in the network. Each cluster has a small set of routers which have direct links to the neighbor clusters, thus handling the traffic in and out of the cluster. We call them border routers and can be e.g. AS gateways in the case of the Internet. We also assume that:

- The deviation of the clock values of any pair of different nodes is bounded in a small range, $\Delta$ time units; this helps synchronize the periodic updates; the algorithm can also be extended to work for bounded clock drifts (implying unbounded deviation of clock values, as described in [11]). Here the protocol is presented with the assumption of bounded offsets, for simplicity of the presentation.

- Every router participates in the protocol.

- The border routers are very difficult to be compromised. Since they are the gateways for traffic between clusters, it is possible to assume they are administrated and protected intensively.

- Clusters that forward traffic between other clusters have backbone routers which will do the job of forwarding transit traffic.

The attacker is modeled as an adaptive adversary which can eavesdrop and launch a bounded number of directed attacks (i.e. flooding attacks targeting a set of specific hosts/routers). There is an assumed bound in the time that it takes for the adversary between the moment it gets the information on a possible target (e.g. some important router) and the time it can launch a directed (distributed) attack to it. We call this exposure delay and bound it by $E$ time units.

We say a router is compromised, if the attacker can break into the router’s system and get all its information. For the sake of the presentation, we first consider that the attacker cannot control the behavior of the compromised routers. In section 8, we will present and discuss mechanisms against malicious behavior of the compromised routers.
Recall that the network is assumed to be partitioned into a set of clusters, and that neighboring clusters are connected via border routers. Depending on whether the source and the destination of packets are within the same cluster, traffic can be classified into intra-cluster traffic and inter-cluster traffic. Controlling intra-cluster traffic, due to the limited number of hosts and paths in one cluster, can be done e.g. via flow filtering based on some fairness[5]. Each cluster can adopt its own method to prevent intra-cluster traffic flooding. In this paper, we concentrate on controlling inter-cluster traffic.

Each cluster \( C_i \) will mainly control three types of inter-cluster traffic:

1. **outbound traffic** i.e. from \( C_i \) to other clusters;
2. **inbound traffic** i.e. from other clusters to \( C_i \);
3. **transit traffic** i.e. traffic that passes through \( C_i \) and goes to another cluster.

In order to go out/in or pass through a cluster, packets need to have some permissions which are called **authentication tokens**. Roughly speaking, these authentication tokens are like “passports” (permission to leave a cluster) and “visas” (permission to enter or pass through a cluster) for the traffic. To control the validity of the authentication tokens of outgoing and incoming packets, in each cluster there are designated **Egress Checking Routers (ECR)**, **Ingress Checking Routers (ICR)** (cf. section 4.3). Transit traffic is controlled by **backbone routers**. Given that the attacker can observe the tokens of legitimate packets and learn which routers are the ECRs and ICRs, these tokens are binded to specific IP addresses and checking routers, also tokens and the checking routers will be changed periodically; the latter actions are also called “hopping” in the paper. By controlling the length of each period not to exceed the exposure delay, CluB can avoid directed flooding attacks to the checking routers. Next we will present key components in CluB and describe how we cope with the challenges mentioned in the introduction.

### 4.1 Coordination authorities

Every cluster needs a coordination entity, which can be implemented in a centralized manner by a single node, protected by a secure overlay as in [14, 24]. For simplicity we present the proposed solution with a single coordinator per cluster. This coordinator is publicly trusted and well protected, e.g. via a secure overlay as mentioned above, to avoid turning it to a DoS attack target. The coordinator maintains security policies of its cluster and cooperates with coordinators in other clusters. Generally, each coordinator has the following tasks:

- It decides whether to allow a host in the cluster to send outbound traffic out of this cluster, or to allow a host of another cluster to send inbound and transit traffic to/via its cluster. The coordinator will grant the **authentication code** of the cluster to the hosts of the approved requests. The authentication code is used for computing the authentication tokens of legitimate packets. If a host wants to apply for permission for sending inbound traffic or transit traffic to another cluster, the coordinator of the local cluster will forward the host’s request to the coordinator of the corresponding cluster.

- It generates new authentication codes for the cluster periodically; and gives the codes to the corresponding routers for checking the validity of traffic.

- As mentioned in the preceding paragraphs, not only the tokens, but also the routers for checking them will be changed periodically. It is the coordinator that appoints the checking routers for each period of time. We describe this process in more detail in section 4.3.

Considering potential attacks to the permission requesting stage, these can be referred to the DoC problem (cf. section 2) and can be mitigated in two levels: First in intra-cluster level, a secure overlay is used to protect the coordinator, hence mitigating the potential attempts by the attacker to prevent the requests of legitimate hosts from being received by the coordinator. Second in inter-cluster level, coordinators may
control the sending rate when they forward requests. The rate may be varied according to policies agreed among clusters.

4.2 Authentication tokens

When a packet is being forwarded in/via a cluster, the latter could be source, intermediate or destination cluster. Hence Club has three kinds of authentication tokens for legitimate packets, which are computed using the corresponding authentication codes:

- **Outbound-authentication token** is used for packets going out of the source cluster. This kind of token is generated with the *outbound-authentication code* of the source cluster.

- **Inbound-authentication token** is used for packets being forwarded within the destination cluster. This kind of token is generated with the *inbound-authentication code* of the destination cluster.

- **Transient-authentication token** is used for packets passing through each intermediate cluster. This kind of token is generated with the *transient-authentication code* of the corresponding intermediate cluster.

Every cluster has its own authentication codes, and all authentication codes will be changed periodically. A host that wants to send packets to another cluster, to compute valid authentication tokens for its packets, it needs the current outbound-authentication code of the local cluster, the current inbound-authentication code of the destination cluster, and the current transit authentication code of each intermediate cluster. Considering that a host may get the authentication codes and share them with other hosts, which is referred as *colluding*, the authentication codes will be given in a hash format binding with specific IP addresses and checking routers: Each authentication code will be hashed together with the source and destination IP address and the number of period as well as the ID of the router who will check the validity of the corresponding token, as shown below in formula 1, (where “||” denotes concatenation, \( PID \) is number of period, \( RID \) is the checking router’s ID and \( HASH \) can be an one-way hash function, such as MD5, SHA.)

\[
HASH (SrcIP||DesIP||AuthToken||PID||RID). 
\] (1)

4.3 Egress/Ingress Checking Routers

Intuitively, letting the border routers to check the authentication tokens of all kinds of traffic would be a straight-forward option. But this is not practical, since the border routers can be the bottleneck of the cluster as far as performance and security are concerned. Hence, in Club, each cluster has one or more ordinary routers that act as the logical exit control of the cluster, called the Egress Checking Routers (ECRs). All the packets generated from this cluster to other clusters are supposed to be routed via some ECRs where the packets’ outbound-authentication tokens are checked. To prevent direct attacks to ECRs, these routers change periodically, and also the information about which routers are ECRs is not known publicly. How to achieve that is explained later in this section.

In addition to ECRs, Ingress Checking Routers (ICRs) also exist in each cluster. The task of ICRs is symmetric to ECRs: All the packets whose destinations are in this cluster are supposed to pass through some ICRs where the inbound-authentication tokens of the packets are checked. The set of ICRs of each cluster changes periodically, too.

Besides, recall that we have the backbone router(s) in each cluster, responsible for checking packets in transit. These are assumed to be part of the underlying routing protocol (e.g. they can be the same as the backbone routers in the autonomous systems in Internet) and so do not change explicitly in Club.
Dynamically changing ECRs and ICRs: In the beginning of every time period, the cluster coordinator computes the new ECRs and ICRs for this period using some pseudo-random functions, and then sends a notification message to every router in the cluster. The notification message to a specific router will only tell whether this router is a checking router or not in the current time period. If it is, then the outbound/inbound authentication token for the current time period will also be included in the notification message. Since a notification message should only be readable by the corresponding receiver, the coordinator encrypts it with the receiver’s public key and patches its digital signature in it. Due to the time offsets $\Delta$ and packet delivery latency $\mu$, each checking router has to serve $\Delta + \mu$ time units longer than a time period.

In each cluster there could be more than one checking routers, considering fault-tolerance and load-balancing issues. In each period, the non-checking routers do not know which router is checking router, while each checking router knows it is a checking router but will not tell this to other routers. In section 5 it is explained how traffic is routed via them without public knowledge of their status.

5 The Basic Protocol in CluB

In this section, the basic protocol in CluB will be presented. Algorithm 1 shows the pseudocode of ECRs and ordinary routers processing outbound packets. The pseudocode of ICRs and inbound packet processing is symmetric.

Briefly, the protocol consists of the following parts, which will be also explained in detail subsequently in this section:

i) **Permission requesting**: Before a host sends packets to other clusters, it should send a request to the coordinator in the local cluster. If it is allowed to send packets outbound, the coordinator will give the IDs of one ECR and one ICR of the local cluster and the destination cluster respectively. The authentication codes of the clusters that the packets will go through will also be given.

ii) **Packet encapsulation**: If the host gets all the information it needs for sending packets, it will form the packets as required by the protocol, essentially adding a new header.

iii) **Packet forwarding**: An outbound packet will be checked by one of ECRs in the source cluster, and will be routed out of the source cluster through some border router. Then it will pass through intermediate clusters, and will be forwarded into the destination cluster through some border router of that cluster. In the destination cluster, the packet will be checked by one of ICRs and finally be forwarded to the destination host. Each authentication token will be checked by the corresponding checking routers. If everything is valid, the packet will reach its destination.
5.1 Permission Requesting

Upon receiving a request from a host, the coordinator first checks whether this host misbehaved before according to some local policy, e.g. black-listing as used in the capability-based mechanisms. If everything is OK, the coordinator forwards this request to the coordinator of the destination cluster, while each coordinator of the intermediate clusters will also get this request. They too, will decide whether to grant the permissions to the host according to their local policies. If everything is OK, the destination coordinator provides the address of one of its ICRs and its inbound-authentication codes; similarly, all the intermediate coordinators provide their transit authentication codes. After the local coordinator gets the (encrypted) reply messages, it will hash each of the authentication codes as well as the outbound-authentication code in the way shown in formula 1. Then the local coordinator will put these hash values and the addresses of ECR and ICR into one message and return it to the requesting host, encrypted using the host’s public key.

5.2 Packet Encapsulation

After successfully getting the reply, the requesting host can compute the authentication tokens for its packets (using formula 2, cf. section 5.4) and send them to the desired destination. In CluB, the regular packets have some more header fields than the normal (e.g. IP) packets. These extra fields form a data structure, called routing vector, containing everything that a packet needs for going through the checking points in the process.

In more detail, the first entry of the vector contains the address of an ECR of the source cluster and the cluster’s outbound authentication token. The last entry of the vector contains one of the ICRs of the destination cluster and corresponding inbound-authentication token. The intermediate entries contain the corresponding transit-authentication tokens. Each entry also contains one bit, called checking bit, indicating whether the packet has been checked by the corresponding router.

At this point we can use an example to illustrate the routing vector (and continue using it for the packet forwarding procedure which is described subsequently). As shown in Figure 1, host $h_1$ wants to send packets to $h_3$ in cluster $C_3$. The routing vector in the packets will be like this:

$$\{(C_1, r_1^1, 0, tok_1), (C_2, border, 0, tok_2), (C_3, r_3^3, 0, tok_3)\}$$

where $tok_1$, $tok_2$, $tok_3$ are the outbound, transit and inbound authentication tokens, and $r_1^1$ is an ECR in $C_1$, $r_3^3$ is an ICR in $C_3$. Initially, all the checking bits are set to 0. We describe how the packets are routed from $h_1^1$ to $h_3^3$ in the next section.

5.3 Packet Forwarding

Consider the example is illustrated by Fig. 1: A regular packet from $h_1^1$ to $h_3^3$ will be first routed to router $r_1^1$ for checking the outbound authentication token, then it will be routed to $b_1^1$ and out of cluster $C_1$, with next hop $C_2$. In cluster $C_2$ the packet will be forwarded by the backbone router(s) where the transit authentication token is checked. After passing through $C_2$, the packet will enter into cluster $C_3$ via $b_3^1$ and be routed to router $r_3^3$ (ICR). After $r_3^3$ checks the inbound authentication token, the packet can be finally routed to its destination $h_3^3$. Below we detail some important steps of the forwarding process using our running example.

Forwarding within the source cluster  When an ordinary router gets an outbound packet (it can tell that from the source and destination addresses), it checks whether this packet has been already checked by the ECR in the routing vector (i.e. by $r_1^1$ in $C_1$, considering the example in Fig. 1). If the packet has not been checked by some ECR in the current cluster (i.e. its checking bit in the first entry of is 0), then the packet
will be routed to this ECR. If the checking bit is 1, then the packet will be routed to exit the cluster (i.e. via $b_1^1$ in the example).

When an ECR receives an outbound packet, it will check whether the router’s name in the first entry of the packet’s routing vector is equal to its own name. If not, then it will behave as an ordinary router. If yes, it will check the validity of the packet, i.e. redo the computing for the outbound authentication token and compare it with the corresponding value contained in the packet; if they match, then the router will flip the checking bit and route the packet towards out of the cluster. The packets having invalid hash values in the routing vectors will be dropped.

Given that the attacker may not know which router is ECR, it may guess and fill an arbitrary router’s IP in the routing vector. So if a non-ECR router receives a regular outbound packet and finds its IP address in the first entry of the routing vector, this router will drop the packet.

**Forwarding in the intermediate cluster(s) and the destination cluster** When packets pass through an intermediate cluster, they will be forwarded by the backbone routers ($b_1^2$ and $b_2^2$ in $C_2$ in our running example). When a backbone router receives a transit packet, it will check the value for the transit-authentication token contained in the packet by recomputing the token and comparing the two values. If the two values match, then the corresponding checking bit will be set to 1, and the packet will be routed to the next cluster (maybe via other backbone routers).

After being forwarded through the intermediate cluster, the packets will reach the destination cluster ($C_3$ in our example). There, when an ordinary router receives an inbound packet, it will look into the last entry of the routing vector in the packet and do a symmetric process as the one treating an outbound packet. I.e. it will see whether this packet should be routed to its inscribed ICR (to get checked) or to the final destination (if it has been checked by the ICR).

### 5.4 Filtering replayed packets

In *CluB*, the authentication tokens can be implemented by directly using the hash values of the authentication codes (computed using formula 1). A malicious host cannot effectively reuse authentication tokens which are stolen from other packets, as these tokens are binded IP addresses. Alternatively, it may resend to the ECRs/ICRs packets that are eavesdropped from the network. Since the authentication codes are hashed with period numbers and IDs of ECRs/ICRs, the attacker can only “replay” packets to a specific ECR/ICR during the current period.

To prevent replaying packets in the current period, the authentication token should be packet-specific. We may compute the authentication tokens by hashing the entire packet with the hash values of authentication codes. However, for efficiency, when a legitimate host sends an outbound packet, it will assign a sequence number to the packet and compute the authentication tokens by hashing the sequence number with the hash values of the authentication codes computed by formula 1. In one period, each outbound packet of this host has a unique sequence number. To achieve that, the size of the number space should be big enough. e.g. 32 bits for $4kM$ packets per period. The new hash values will be put into the corresponding entries in the routing vector. So the authentication tokens contained in the routing vector are computed as below:

$$\text{HASH} (\text{hash} \oplus \text{SeqNumber}||\text{hash}), \text{where } \text{hash} = \text{HASH} (\text{SrcIP}||\text{DesIP}||\text{AuthCode}||\text{PID}||\text{RID})$$  \hfill (2)

The sequence number is sent with the packet. So a checking router will check the validity of a packet by redoing the hashing procedure as formula 2.

Identifying the replayed packets needs the checking routers to keep states for the received packets, which would be a big overhead. To reduce this overhead, each ECR/ICR can also employ a *Bloom Filter* [18] to check the “replaying” packets. A Bloom filter can check identical packets with bounded space overhead,
Algorithm 1 Algorithm for a router (can be ECR router) processing an outbound packet.

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>initiate</strong> isECR; /* This denotes whether this router is an ECR*/</td>
</tr>
<tr>
<td>2</td>
<td>clear the Bloom filter; /* Used for checking replay packets */</td>
</tr>
<tr>
<td>3</td>
<td>AuthToken ← Outbound authentication token for current period;</td>
</tr>
<tr>
<td>4</td>
<td>PID ← the number of current period;</td>
</tr>
<tr>
<td>5</td>
<td>–When receive an outbound packet PK_out</td>
</tr>
<tr>
<td>6</td>
<td>routerECR ← PK_out.ECR;</td>
</tr>
<tr>
<td>7</td>
<td>/<em>check whether itself is ECR and should it check this packet</em>/</td>
</tr>
<tr>
<td>8</td>
<td>if isECR = true &amp; myID == routerECR then</td>
</tr>
<tr>
<td>9</td>
<td>SeqNum ← PK_out.SeqNumber;</td>
</tr>
<tr>
<td>10</td>
<td>hash ← HASH(srcIP</td>
</tr>
<tr>
<td>11</td>
<td>if HASH(hash ⊕ SeqNum</td>
</tr>
<tr>
<td>12</td>
<td>if insert(SeqNum</td>
</tr>
<tr>
<td>13</td>
<td>/*Insert the value (SeqNum</td>
</tr>
<tr>
<td>14</td>
<td>PK_out.outBit = 1;</td>
</tr>
<tr>
<td>15</td>
<td>route the packet out of the cluster;</td>
</tr>
<tr>
<td>16</td>
<td>else</td>
</tr>
<tr>
<td>17</td>
<td>drop PK_out;</td>
</tr>
<tr>
<td>18</td>
<td>end if</td>
</tr>
<tr>
<td>19</td>
<td>else</td>
</tr>
<tr>
<td>20</td>
<td>drop PK_out;</td>
</tr>
<tr>
<td>21</td>
<td>end if</td>
</tr>
<tr>
<td>22</td>
<td>end if</td>
</tr>
<tr>
<td>23</td>
<td>if isECR != true &amp; myID == routerECR then</td>
</tr>
<tr>
<td>24</td>
<td>drop PK_out;</td>
</tr>
<tr>
<td>25</td>
<td>end if</td>
</tr>
<tr>
<td>26</td>
<td>if PK_out.outBit then</td>
</tr>
<tr>
<td>27</td>
<td>if PK_out is received directly from a host then</td>
</tr>
<tr>
<td>28</td>
<td>drop PK_out;</td>
</tr>
<tr>
<td>29</td>
<td>else</td>
</tr>
<tr>
<td>30</td>
<td>route the packet out of the cluster;</td>
</tr>
<tr>
<td>31</td>
<td>end if</td>
</tr>
<tr>
<td>32</td>
<td>end if</td>
</tr>
<tr>
<td>33</td>
<td>if !PK_out.outBit &amp; myID ≠ routerECR then</td>
</tr>
<tr>
<td>34</td>
<td>route PK_out to routerECR;</td>
</tr>
<tr>
<td>35</td>
<td>end if</td>
</tr>
</tbody>
</table>

and gives no false negative. When an ECR/ICR receives a valid outbound/inbound packet, if it received a packet with the same combination of (SeqNumber, SrcIP, DesIP) in the current time period, the newly received packet will be dropped. The probability that the Bloom filter may falsely consider there was an identical packet received in the current period can be arbitrarily small. If an ECR/ICR handles $W$ valid outbound/inbound packets for a period, and the Bloom filter uses $k$ hash functions and an $n$ bits array, then the false positive probability approximates to $\left(1 - e^{-kW/n}\right)^k$. The minimum value of this probability approximates $(0.6185)^{\frac{n}{W}}$ with an optimal $k$. Therefore if the ECR/ICR handles $1M$ valid outbound/inbound packets per period, a Bloom filter of $2MB$ is enough for checking identical packets with the false positive probability less than $5 \times 10^{-4}$.

**Dealing with malicious packets whose checking bits are set** The adversary may let the compromised hosts send packets whose checking bits are 1. Then these packets may reach the destination without being checked. Since the checking bits should be changed only by the corresponding checking routers, if a router
receives a packet whose checking bits are 1 directly from a host, this packet is definitely malicious and will be dropped. Given the assumption that the routers do not misbehave, this mechanism can prevent the malicious packets from avoiding the checking process. In section 8.1, we extend the method and discuss solutions against malicious behaviors of compromised routers.

**Compatibility** Routing a regular outbound packet from its host to its assigned ECR can be done using contemporary routing protocols, such as RIP and OSPF, by using the ECR routers in the routing vector as target for this itinerary. As a matter of fact, this is true for each part of the itinerary; i.e. BGP can be used for inter-cluster routing, by using the cluster IDs as the targets.

5.5 Token-Refreshing

Since the ECRs, ICRs and all authentication codes change periodically, a host sending outbound packets may have to request them when a new period begins, thus initial steps repeat. To care for enhanced efficiency considering the cost in the requesting process, we let the coordinator of each cluster give the information about its ICRs, inbound and transit authentication codes for the new period to the coordinators of other clusters. So upon receiving a refresh request from a host who got the sending permission before, the coordinator does not have to forward the request to the other coordinator(s) that may be involved. If the local cluster did not receive any report of possible misbehaviors of this host, new “hopping” information will be granted to the host.

6 Analysis and Evaluation

In this section, we will analyze the proposed methods from two perspectives. First, the effectiveness of filtering distributed malicious traffic will be studied, also in connection to the capability-based mechanism. Second, we will analyze the influence of malicious packets flooding to legitimate traffic.

6.1 Filtering effectiveness

Since the compromised hosts can be distributed among different clusters, the malicious traffic generated from different clusters has different probability to reach the target. Here we consider that the attacker, who does not know any of the authentication codes, randomly piggybacks some hash values in the routing vector. The expectation of the amount of malicious traffic reaching the target is estimated below.

**Proposition 1.** Assume that the proportion of the malicious traffic that is generated at the clusters $h$ hops (at cluster-level) away from the target is $\lambda_h$. Suppose the total amount of the malicious traffic is $M$, then the expectation of the amount of malicious traffic reaching the target is:

$$\sum_{h=1}^{\delta} \lambda_h \cdot M \cdot p_{in} \cdot p_{eg} \cdot p_{tr}^{h-1},$$

where $p_{in}$, $p_{eg}$, $p_{tr}$ are the probabilities that the attacker finds out one cluster’s outbound, inbound and transit authentication code respectively and $\delta$ is the diameter of the network in cluster-level.
To reflect on what the above implies, let us suppose that the clusters in the CluB are the autonomous systems (AS) in the Internet. Suppose also that the amount of the malicious traffic generated from each AS to a specific AS is the same, and \( p_{in} = p_{eg} = p_{tr} = p \). We use the data of neighbor ASes distribution of a specific AS mentioned in [6], which is shown in Fig. 2. Then the proportion of the malicious traffic that can reach the target is estimated as: \( \sum_{h=1}^{6} p^{h+1} p_h \), where \( p_h \) is the probability that an AS resides exactly \( h \) hops away from the given AS. Here we choose the diameter as 6, since most of the ASes reside within 6 hops away from the reference AS.

Consider an example, if each kind of authentication code has 32 bits, then the probability to guess out each of the authentication codes is \( \frac{1}{2^{32}} \). Recall that due to the time offsets and packet delivery latency, at a given time there could be more than one codes valid for each kind of authentication tokens. Let \( x \) be the number of outbound/inbound/transit authentication codes valid in each cluster at a given time. So in this example, \( p = 1 - \left(1 - \frac{1}{2^{32}} \right)^x \) which equals to the probability that the attacker guesses out at least one of the valid codes for each kind of authentication tokens in one cluster at a given time. Consider that usually the time offsets and the packet delivery latency are quite small, e.g. we choose \( x = 2 \). Therefore, in this specific example, the percentage of the malicious traffic that can reach the target will be smaller than \( 10^{-18} \).

6.2 Filtering effectiveness comparison with the capability-based mechanism

To put the filtering properties of CluB in perspective, it is worthwhile to study them in connection to the capability-based method filtering properties. As discussed in the related work section, the capability-based method assigns all the responsibility for issuing permissions (capabilities) to the end-server, while in CluB the responsibility is distributed in the network, via the clusters. Assuming CluB can use the same procedure of black-listing as that used in the capability-based mechanisms, we can show the potential effect of the difference between the two distributions of control. Via a couple of scenarios based on our running example, it is observed that CluB can offer more efficient filtering.

![Figure 3](image-url)

(a) Each compromised host in cluster \( C_1 \) has probability 0.2 to attack each of the other clusters
(b) Each compromised host in cluster \( C_1 \) has probability 0.5 to attack each of the other clusters

Figure 3. Filtering efficiency comparison between CluB and the capability-based mechanism. The number of compromised hosts in cluster \( C_1 \) (in fig.1) that can attack hosts in cluster \( C_3 \) in different periods is shown.

In the case study, we create 4 clusters with topology as shown in Fig.1. There are 1000 hosts in cluster \( C_1 \), all of them are considered as compromised. In every simulation period, each of the compromised hosts attacks each of the other three clusters with probability 0.2 (shown in Fig. 3(a)) or 0.5 (shown in Fig. 3(b)). Initially, none of the compromised host are prevented from sending packets to the other three clusters. We count the number of compromised hosts that can send traffic from cluster \( C_1 \) to cluster \( C_3 \) for 5 periods. From Fig.3, we can see that this number drops faster in CluB than that in the capability-based mechanism. This is because when some compromised hosts attack cluster \( C_2 \) and \( C_4 \) before attacking cluster \( C_3 \), they cannot send traffic to cluster \( C_3 \) later, since they are blacklisted in \( C_2 \) and \( C_4 \) and cannot get the transit
authentication codes. In the capability mechanism, those compromised hosts (who attacked \( C_2 \) and \( C_4 \) before) can still get capabilities from cluster \( C_3 \) and send traffic to it as long as they are not blacklisted in \( C_3 \).

### 6.3 Effect of packets flooding to the checking routers

By keeping the length of the periods less than the exposure delay, we may prevent the adversary from attacking the checking routers directly. For the unlikely event that an attacker is able to launch a directed attack in a shorter time, we would like to know the harm that this may cause. Therefore we wish to analyze the effect of the directed flooding attacks to the checking routers. We show the analysis for the outbound packets flooding; the situation of inbound packets flooding is symmetric. Let us first give the following definition:

**Definition 1.** The pass-through probability \( P_{E-th} \) is the probability that a legitimate outbound packet will be processed by the ECR whose ID is given in the routing vector of the packet.

Next we will analyze the pass-through probability when the attacker launches outbound packets flooding to the ECRs.

Consider that there are \( \Phi \) ECRs in a cluster and each ECR can process \( B \) outbound packets per time unit. If an ECR has to process more than \( B \) outbound packets per time unit, then the ECR will randomly process \( B \) packets from them (by sampling over small time intervals). When the attacker knows the information about the current ECRs in the cluster, it may control the compromised hosts to attack \( \psi \) ECRs, \( \psi \leq \Phi \). We use \( s_i \), \( 1 \leq i \leq \psi \), to denote the number of malicious outbound packets that are sent to the \( i^{th} \) ECR that is under attack. And we assume that \( \forall i, s_i \geq 1 \). For the sake of the analysis, we assume that in each time unit, each ECR receives the same number of legitimate outbound packets. We use \( Q \) to denote this number. Here to avoid non-trivial options, we assume \( Q \leq B < Q + s_i \).

Given the \( i^{th} \) ECR that is under attack, the pass-through probability for a legitimate outbound packet assigned to this ECR is \( \frac{B}{Q + s_i} \). When a legitimate outbound packets is sent to an ECR that is not under attack, the pass-through probability of this packet is 1. Since each ECR receives the same amount of legitimate outbound packets, we assume the probability that a legitimate outbound packet is send to a specific ECR is \( \frac{1}{\Phi} \). We have the following lemma:

**Lemma 1.** For fixed \( \Phi \) and \( \psi \), the pass-through probability is:

\[
P_{E-th} = \frac{\Phi - \psi}{\Phi} + \frac{1}{\Phi} \sum_{i=1}^{\psi} \frac{B}{Q + s_i}. \tag{4}
\]

**Proof.** The probability that a legitimate outbound packet is sent to one of the ECRs, say \( ECR^* \), is \( \frac{1}{\Phi} \). If \( ECR^* \) is under attack (i.e.\( ECR^* \) is one of the \( \psi \) ECRs that are under attack), then the probability that this packet is not processed by \( ECR^* \) is \( 1 - \frac{B}{Q + s_i} \), if \( ECR^* \) is the \( i^{th} \) ECR that is under attack. So the probability that a legitimate outbound packet is not processed by the corresponding ECR is

\[
Pr[\text{not processed}] = \frac{1}{\Phi} \sum_{i=1}^{\psi} \left(1 - \frac{B}{Q + s_i}\right) = \frac{\psi}{\Phi} - \frac{1}{\Phi} \sum_{i=1}^{\psi} \frac{B}{Q + s_i}.
\]

So we have

\[
P_{E-th} = 1 - \left(\frac{\psi}{\Phi} - \frac{1}{\Phi} \sum_{i=1}^{\psi} \frac{B}{Q + s_i}\right) = \frac{\Phi - \psi}{\Phi} + \frac{1}{\Phi} \sum_{i=1}^{\psi} \frac{B}{Q + s_i}.
\]

\( \square \)
We use $m$ to denote the number of compromised hosts in the cluster. Each compromised host can at most send $b$ outbound packets per time unit. It can be shown that the attacking strategy to minimize $P_{E-th}$ with \( \psi \) ECRs under attack is to let $s_i = \frac{mb}{\psi}$ for all $i \in \{1, \cdots \psi\}$. Furthermore, when the capacity of each ECR for processing the outbound packets is just enough for processing the legitimate outbound packets, that is $B = Q$, the attacker’s option for minimizing $P_{E-th}$ is to attack all the ECRs, i.e. $\psi = \Phi$, and $\forall i : s_i = \frac{mb}{\Phi}$. The corresponding $P_{E-th}$ is $\frac{\Phi B}{mb + \Phi B}$. When each ECR has more capacity for processing outbound packets, that is $B > Q$ (i.e. there is over provision in each ECR’s capacity by $B - Q$), the strategy to minimize $P_{E-th}$ for the attacker is to attack $\frac{mb(\sqrt{\frac{B}{B-Q}}-1)}{Q}$ ECRs. Note that, since $\psi$ is at most $\Phi$, if $\frac{mb(\sqrt{\frac{B}{B-Q}}-1)}{Q} > \Phi$ then $P_{E-th}$ is minimized when $\psi = \Phi$. We have the following lemmas as summary and the proof will be shown in the appendix:

**Lemma 2.** If $B = Q$, then $P_{E-th} \geq \frac{\Phi B}{mb + \Phi B}$.

**Lemma 3.** If $B > Q$, then

- If $\frac{mb(\sqrt{\frac{B}{B-Q}}-1)}{Q} \leq \Phi$, then $P_{E-th} \geq \frac{B - \sqrt{B(B-Q)}}{Q}$.
- If $\frac{mb(\sqrt{\frac{B}{B-Q}}-1)}{Q} > \Phi$, then $P_{E-th} \geq \frac{\Phi B}{\Phi Q + mb}$.

### 7 Controlling the Granularity of Clusters

One cluster in CluB can be naturally mapped to an autonomous system or a group of neighbor autonomous systems in the Internet. Optimizing the granularity of clusters involves many factors. In this section, we will discuss the factors that play significant roles in choosing the granularity of the clusters.

**Security and Processing Load** Note that if some of the ECRs/ICRs are compromised, then the outbound/inbound authentication code may be revealed. If we assume that each router has a certain probability to be compromised, then each cluster should not have many ECRs/ICRs, since this may increase the probability of the revelation of the authentication codes. Given a router, we use $\eta$ to denote the probability it is compromised (assuming that each router has the same probability to be compromised). Then the probability that at least one of the ECRs is compromised is:

$$P_{comp} = 1 - (1 - \eta)^\Phi,$$

Figure 4 shows how $P_{comp}$ grows with the number of ECRs. If we want to control $P_{comp}$ under a threshold, say $\Gamma$, then $\Phi \leq \log_{1-\eta}(1 - \Gamma)$.

However, one cluster should have enough ECRs to handle its outbound traffic. If we use SHA-1 as the hashing function for the authentication tokens, according to Crypto++5.6.0 benchmarks [1], it can be processed 153 MiB/sec on a 1.83 GHz Inter Core2 machine. Some off-the-shelf routers such as cisco 7600 series have processors of 1.2 GHz, taking a conservative estimation that they can process the outbound packets with speed 50 MiB/Sec. If the volume of legitimate outbound traffic is 10 Gbps, then there should be $\frac{10Gbps}{50MiB/Sec} \approx 25$ ECRs. Considering the threshold of $P_{comp}$, we should control the volume of legitimate outbound traffic not bigger than $B \log_{1-\eta}(1 - \Gamma)$. This is a constraint that we have to consider, since generally the amount of outbound traffic grows with the size of the cluster.

Another security issue is that the coordinator’s incoming link could be flooded by the compromised machines within the cluster. The secure overlay mechanism can resist attacks that render up to 40% of the nodes inoperable [24]. Since the potential number of compromised machines may grow with the size of the cluster, if CluB use secure overlay to protect the coordinators, the size of the clusters should be controlled so that the attacker can hardly assemble enough compromised machines to attack 40% of the overlay nodes.
Traffic Stretch In the source cluster, an outbound packet is forwarded to the border router via one of the ECRs. The shortest path length from the source to the border router via one ECR is usually bigger than the original shortest path length (which is the distance) between the source and the border router, unless the ECR is on the original shortest path. We use traffic stretch to measure this phenomenon. Here we define the stretch for the outbound traffic being forwarded in the source cluster (the situation for inbound traffic is symmetric).

**Definition 2.** Traffic stretch is the ratio between the shortest path length from the source of the packet to the border router via one ECR and the original shortest path length from the source to the border router.

Intuitively, the more ECRs a cluster has, the less the average traffic stretch will be. This is because more ECRs imply higher probability that the shortest path length via one of them is close to the original shortest path length. The actual relationship between the traffic stretch and the number of ECRs depends on the intra-cluster topology. To illustrate this issue, we present a study on several intra-cluster topologies. In particular, we illustrate the traffic stretch as a function of the density of ECRs in the cluster. The first topology is a balanced tree (height of 4 and each internal node has 6 child nodes, 1555 nodes in total). The second topology is a (33 × 33) grid. The third topology is a power-law topology (with 1000 nodes). The fraction of ECRs among all the routers of the cluster is between 0.001 and 0.03. Figure 5 depicts the stretch of different intra-cluster topologies. Figure 6 shows the stretch of power-law topologies with different sizes. In both of the figures, the average stretch is decreased when the fraction of ECRs grows. It is interesting to see that in both of the figures, the decreasing speed of the average stretch is quite fast before the fraction of ECRs reaches 0.005; and after the fraction of ECRs reaching 0.01 the decreasing speed of the average stretch is quite slow. Given the trade-off between security and the traffic stretch, one could use such diagrams to decide on the ECR density (e.g. with the studied system settings, the optimal fraction of ECRs is between 0.005 to 0.01.)

Path Diversity In CluB, if the packets from one cluster to another have to be forwarded along a new path via different clusters, then the sending hosts have to require transit authentication codes of the new clusters in the path. This might be caused by the link failures or routing policy changes. Next we will discuss the relation between the cluster sizes and the probability of path changing as a result of link failures.

We assume that the number of physical links between two neighbor clusters is proportional to the size of the clusters. So if we assume that every cluster has the same size (measured as the number of routers in the
cluster), then we can use an increasing function, e.g. \( f(n) = \log \log n \), to denote the number of physical links between two neighbor clusters. We say the connection between two neighbor clusters fails when all the physical links between them fail. Given a physical link, we use \( \theta \) to denote the probability that this link fails (we assume that each link has the same probability to fail). Let us define a path from one cluster to another cluster as an ordered sequence of clusters: \( \{C_0, C_1, \ldots, C_L\} \), such that for any \( i \neq j \) we have \( C_i \neq C_j \), and there exists at least one physical link between \( C_i \) and \( C_{i+1} \) for \( i = 0, \ldots, L - 1 \). \( L \) is the length of the path. We say that a path fails when there exists at least one pair of two consecutive clusters in the path whose connection fails. We use \( P_{\text{path-f}} \) to denote the probability that a path fails. Hence, we have:

\[
P_{\text{path-f}} = 1 - \prod_{i=0}^{L-1} (1 - \theta f(n))
\]

If assume that every cluster has the same size, we have:

\[
P_{\text{path-f}} = 1 - (1 - \theta f(n))^L.
\]

Note that generally the length of a path will decrease when the size of the clusters increases (for example if we cluster \( C_0 \) and \( C_1 \) together then the length of the path will be \( L - 1 \)) and since \( a^x \) is a monotonically decreasing function when \( 0 < a < 1 \) and \( x > 1 \), the value of \( (1 - \theta f(n))^L \) will increase when \( n \) grows. Previous work (such as [10]) shows that the AS-level topology is complied to power-law. Chung et al. [7] showed that in common Internet graphs, where the topology is power-law with powers ranging from 2.1 to 2.45, the average distance between two ASes is \( \log \log (N) \), where \( N \) is the number of autonomous systems. Since a cluster can be naturally mapped to one or a group of autonomous systems, if we assume the topology of cluster-level also follows power-law, then \( L \) can be estimated as \( \log \log (N_C) \), where \( N_C \) is the number of clusters. Figure 7 illustrates the trade-off between \( P_{\text{comp}} \) and \( P_{\text{path-f}} \) using formula 5 and 6. We let the total number of routers equal to 100000, \( n \) ranges from 100 to 3000. From Figure 7 we observe that \( P_{\text{comp}} \) decreases when \( P_{\text{path-f}} \) increases, this is because increasing the size of a cluster will increase the amount of outbound traffic; then more ECRs/ICRs are needed, which leads to the decrease of \( P_{\text{comp}} \). Fig. 7(a) illustrates the trade-off with fixed value of \( \theta \) and varying \( \eta \); while Fig. 7(b) illustrates the trade-off with fixed value of \( \eta \) and varying \( \theta \).

![Figure 7](image)

(a) \( \eta \) ranges from 0.001 to 0.0001, \( \theta \) is fixed to 0.1  
(b) \( \eta \) ranges from 0.1 to 0.05, \( \theta \) is fixed to 0.001

Figure 7. Trade-off between the probability that at least one ECR is compromised and the probability of path changing in cluster level. X axis presents the logarithmic value of \( P_{\text{path-f}} \).

8 Further Discussion

In this section we consider options of a stronger adversary model, in particular, where the adversary can control the behavior of the compromised routers (besides obtaining secret information from them as assumed in section 3).
8.1 Controlling the checking process

Taking algorithm 1 into consideration, if the compromised routers can set the checking bits to 1 in the malicious packets, these packets will be forwarded to the destination without being checked by the corresponding routers. In such a more aggressive threat model, to make sure that each packet is checked by the routers indicated in the routing vector, we may use signatures. When a checking router finishes checking a valid packet, it will fill its signature in the packet header. When other routers get this packet, they will verify this signature, and the packet will be forwarded out of the cluster or to the destination if the signature is valid.

Consider a situation where a compromised router pretends to be an ECR and gives its digital signature to non-legitimate outbound packets. So non-legitimate outbound packets with a valid signature of a compromised router can be routed out of the cluster, as other routers have no idea about which routers are the ECRs. The hopping of checking routers can help to deal with this problem. The coordinator can broadcast in its cluster the checking router list for the previous period in the notification messages. Now, the identifications of the compromised routers who pretended to be checking routers can be revealed and they can be black-listed. In addition to that, border routers may only forward the outbound traffic that passed ECRs in the local cluster; and only forward the inbound traffics that are indicated to be passed through ICRs. This requires that at the beginning of each period, each coordinator contains the information about ECRs and ICRs in the notification messages to the border routers of the cluster.

Using digital signatures implies some computational overhead. After a regular packet has been checked by a checking router, that router’s digital signature will be checked by every router who will then forward the packet. As an alternative to exhaustive checking, we may use probabilistic verification: when a router receives a regular packet which has already been signed by the checking router inscribed in its routing vector, the router tries to verify that signature with probability \( \sigma_h, \sigma_h \in (0, 1) \) and \( h \) is the number of hops between the current router to the checking router. To ensure high probability for each packet to be checked at least once, \( \sigma_h \) can be inversely proportional to \( h \). It is also possible to use one time signature in conjunction with a hash tree to achieve fast verification and a small size of the public key as used in [13].

8.2 Dealing with valid packets flooding

When a router is compromised, all the secret information of this router will be revealed to the attacker. So if the attacker compromises an ECR, it will know the outbound authentication code. Hence, in such situation the attacker may flood “valid” outbound packets without replaying legitimate packets. This kind of attack is quite hard to defend, since checking routers can not distinguish these “valid” packets from legitimate packets. However, this may happen only during isolated periods, as the checking routers and authentication codes are changed periodically.

If source IP address spoofing is limited, a checking router can deal with this flooding by only processing up to a certain rate of valid outbound packets per each host. E.g. each ECR could maintain a rate-counter (e.g. like a token-bucket) for each of the hosts whose outbound packets it routes; when that exceeds some threshold, packets from that source will be dropped. However, if the adversary can influence the behavior of the compromised routers, the absence of IP spoofing can not be always guaranteed.

As a matter of fact, most IP spoofing defense mechanisms that are not vulnerable to DDoS attacks are router-based [8]. They may rely on routers to filter packets whose source addresses are not in the addresses space of the domains that they are directly connected [19]. Or they may rely on the routers to filter packets with spoofed addresses according to their incoming directions [15]. However, if we assume that the adversary can control the behaviors of the compromised routers, then these mechanisms may not be suitable for preventing IP spoofing. So valid packets flooding with spoofed IP addresses is more complicated to defend against, and further investigation is needed to mitigate the problem.
9 Conclusion and Future Work

CluB is a proactive method proposed here for mitigating DDoS attacks. It is a distributed method that requires clustering in the network and can combine with various underlying routing solutions. This paper also gives an estimation of the effectiveness of the method under adaptive eavesdropping adversaries. The limitations implied for the adversary options by the methods in CluB, are shown; moreover the effect of malicious traffic on legitimate traffic is limited by bounding from below the probability of successful forwarding of legitimate traffic. In such a framework, there are several factors that play role in the overhead-effectiveness trade-off; adjusting the granularity of control contributes to tuning such trade-off. The paper offers a study of some important factors, including cluster size, density of ECRs/ICRs, confidentiality risk, processing load, traffic stretch, path connectivity and diversity. A useful continuation can be a holistic study that combines factors, such as the distribution level of the coordinating authority, hopping periods, clock drifts, signature methods. It is also worth investigating the option of different clusters adopting different policies, which can be modeled as a game.

References

Appendix

Proof of lemma 2

Proof. In formula 4, we take $s_i$ as variables and use the Lagrange theorem to get the minimum value of $P_{E-th}$. The constrained condition is $\sum_{i=1}^{n} s_i = mb$. Then it can be shown by letting the partial derivatives of formula 4 be zero, that when $\forall i : s_i = \frac{mb}{\psi}$, $P_{E-th}$ will reach its minimum value. By replacing $Q$ with $B$ and replacing $s_i$ with $\frac{mb}{\psi}$ in formula 4 we can get:

$$P_{E-th} = 1 - \frac{mb}{\Phi(B + \frac{mb}{\psi})}.$$  \hspace{1cm} (7)

It’s easy to see that $P_{E-th}$ will decrease as $\psi$ increases, so the minimum value of $P_{E-th}$ will be achieved when $\psi = \Phi$. The lemma follows after replacing $\psi$ with $\Phi$ in formula 7. \hfill \Box

Proof of lemma 3

Proof. (sketch) By replacing $s_i$ with $\frac{mb}{\psi}$, formula 4 can be rewritten as:

$$P_{E-th} = 1 + \frac{\psi(B - Q) - mb}{\Phi Q + \frac{mb}{\psi}}.$$  \hspace{1cm} (8)

It is easy to see that $\psi(B - Q) - mb < 0$, this is because

$$B < Q + s_i \Rightarrow \psi(B - Q) < \psi s_i = mb$$

Take $\psi$ as the variable, and identify the value which makes the derivative of equation 8 be zero. We get i.e.

$$Q\psi^2 + 2mb\psi - \frac{m^2b^2}{B - Q} = 0,$$

whose positive solution is $\psi = \frac{mb(\sqrt{\frac{B^2}{Q} - 1})}{Q}$. 

18
By further checking, in formula 8 \( P_{E-th} \) is minimized when \( \psi = \frac{mb(\sqrt{\frac{B}{Q}} - 1)}{Q} \). Consider that \( \psi \leq \Phi \), so if \( \frac{mb(\sqrt{\frac{B}{Q}} - 1)}{Q} \leq \Phi \), then we can get the following formula by replacing \( \psi \) with \( \frac{mb(\sqrt{\frac{B}{Q}} - 1)}{Q} \) in formula 8:

\[
P_{E-th} = 1 + \frac{mb(\sqrt{\frac{B}{Q}} - 1)}{Q} \cdot \frac{B - mb\sqrt{\frac{B}{Q}}}{\Phi Q + \frac{\Phi Q}{\sqrt{\frac{B}{Q}}}}.
\]  

(9)

Formula 9 can be rewritten as

\[
P_{E-th} = 1 - \frac{mb(\sqrt{B} - \sqrt{B - Q})^2}{\Phi Q^2}.
\]

Since \( \frac{mb(\sqrt{\frac{B}{Q}} - 1)}{Q} \leq \Phi \), we have \(-mb(\sqrt{B} - \sqrt{B - Q}) \geq -\Phi Q \sqrt{B - Q}\). So we have

\[
-\frac{mb(\sqrt{B} - \sqrt{B - Q})^2}{\Phi Q^2} \geq -\sqrt{B - Q} \cdot \frac{\sqrt{B} - \sqrt{B - Q}}{Q},
\]

which leads

\[
P_{E-th} \geq 1 - \frac{\sqrt{B - Q} \left( \sqrt{B} - \sqrt{B - Q} \right)}{Q} = B - \sqrt{B - Q}.
\]

If \( \frac{mb(\sqrt{\frac{B}{Q}} - 1)}{Q} > \Phi \), since formula 8 is a monotonically decreasing function when \( 0 < \psi < \frac{mb(\sqrt{\frac{B}{Q}} - 1)}{Q} \), the minimum value of \( P_{E-th} \) will be achieved when \( \psi = \Phi \). The second item of the lemma follows after replacing \( \psi \) with \( \Phi \) in formula 8.