# Scalable Anti-Censorship Framework Using Moving Target Defense for Web Servers

Vahid Heydari, *Student Member, IEEE*, Sun-il Kim, *Senior Member, IEEE*, and Seong-Moo Yoo, *Senior Member, IEEE*

***Abstract*— Although the Internet has become a hub around which every aspect of our lives—from commerce to leisurely activities—is centered, many around the world are not able to freely access information over the Internet. Some governments censor what the people can and cannot see. In this paper, regardless of the socio-political view points, we focus on the design of anti-censorship technology that can be implemented on the side of the information purveyors. The primary objective is to develop a framework for combating censorship. Our approach aims to make it too expensive and impractical for the adversary to censor Web sites. In particular, we propose the use of Mobile IPv6 to form a moving target defense strategy, where the Web servers logically behave as if they are the mobile nodes (without actually moving). The potential efficacy of this framework is modeled analytically. Probabilistic models are used to derive important metrics and parameters. One key factor termed swarming ratio enables hosting sites to reason about the amount of resources needed to force the adversary's costs over practical limits. This model is used to guide the performance goals and architectural setup of the prototype implementation (modifications are made on the server-side software and Kernel without changing the standard Mobile IPv6 protocol). Hence, the solution can be utilized without any changes to the existing network infrastructure. Furthermore, we introduce a novel, credit-based accounting strategy for grouping of users to drastically shift resource requirements in our favor. Lab-based tests are used to measure performance overheads, and based on the findings, targeted optimizations are performed to consider practical deployment scenarios. The end result is a solution that may also be combined with existing anti-censorship methods (that are end-user-based and/or assisted by friendly network assets) to form a robust anti-censorship solution.**

***Index Terms*— Anti-censorship, moving target defense, mobile IPv6.**

## I. INTRODUCTION

OVER the past decade, widespread access to the Internet has led to significant changes in the way people live. The power of having information that can be readily accessed via a computer or a mobile device has prompted never-before-seen rate of advances in science, technology, and cultural transformations. Though some may also argue the downsides of the Internet, it has enabled all communities around the globe to come much closer together than ever before. Unfortunately, the freedom to access information—which is considered a basic human right by most cultures and their respective laws—is not protected for some people in various parts of the world today. Freedom of information is key to a free, democratic society. In addition to fueling the transformations as described above, it symbolizes a fundamental struggle towards betterment of mankind. As such, some have even argued that the right to access information should be a welfare right: that the burden of guaranteeing such access to its citizens should be placed on the government [1]. Through the use of technology, it is possible to combat anti-censorship, and help people from all around the globe access information that can ultimately help them reach their potential in all areas of science, technology, digital culture, etc. This research work presents a framework through which web servers can combat censorship.

Manuscript received July 24, 2016; revised November 10, 2016; accepted December 7, 2016. Date of publication January 2, 2017; date of current version February 22, 2017. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Athanasios V. Vasilakos.

V. Heydari and S.-M. Yoo are with the Department of Electrical and Computer Engineering, The University of Alabama in Huntsville, Huntsville, AL 35899 USA (e-mail: vahid.heydari@uah.edu; yoos@uah.edu).

S. Kim is with the Department of Computer Science, North Central College, Naperville, IL 60565 USA (e-mail: skim@noctrl.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2016.2647218

The adversary may be a government body or a private group trying to prevent people from accessing certain types of information. Our goal is to make it difficult for any entity to prevent the public from accessing web-hosted information.

There are several ways information on the web is currently censored. Some common techniques include:

- Internet Protocol (IP) address blocking: Block access to certain sites as identified by their IP address(es).
- Domain Name System (DNS) filtering: Certain domain names are not resolved, preventing access.
- Uniform Resource Locator (URL) filtering: URL strings are scrutinized regardless of the domain name.
- Packet filtering: Network packet payloads are monitored for forbidden terms.
- Network Disconnection: Censorship is achieved by turning off the network infrastructure.

The last item—complete network disconnection for some—is seldom implemented in practice, but there are no solutions other than finding another way to get on to the Internet. Thus, complete disconnection is outside the scope of this paper. Most users go around the other four censorship methods by utilizing encrypted tunnels and proxies, such as VPNs [2]–[4] and Tor [5]. In response, the adversary typically attempts to find and block the hosts of these services.

In this paper, we present a solution based on a combination of the features present in Mobile IPv6 (MIPv6) and a moving target defense (MTD) strategy. This framework is designed to be implemented at the servers hosting the content. We use the basic ideas behind MTD approaches that were designed

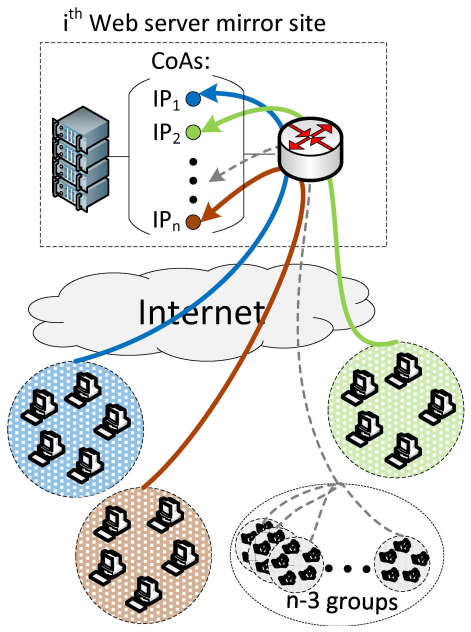


Fig. 1. For illustration purposes, $n$ different randomly chosen CoAs are assigned to different groups of users; three user groups assigned to $IP_1$ $IP_2$ and $IP_n$ are highlighted.

to prevent Distributed Denial-of-Service (DDoS) attacks by detecting flooding attacks where the intended victims (servers) are turned into moving targets for attack isolation [6]–[8]. Such methods were also used to improve user privacy [9]. (In the related work section, we discuss the lack of suitability of these existing MTD approaches for the anti-censorship problem.) Our approach is different from the existing attempts to thwart Internet censorship measures where the end-users or some entities in the network attempt to combat censorship efforts (either individually, or as a community). Our solution leverages the participation of the hosting server, and is orthogonal to the existing methods. In fact, we believe that the joint-use of the different schemes could yield an extremely robust solution.

Typically in MIPv6, a permanent IP (home address) is used to avoid disrupting TCP sessions and one or more care-of addresses (CoA) are used to connect to the other nodes [10]. We use MIPv6, and *treat the web server as if it were a mobile node*. Servers can then utilize dynamically changing IP addresses (based on the CoAs) to avoid filtering and blocking (and also from being attacked). End-users are assigned to random groups and provided with a CoA that they can use to access the website. After some time interval (called shuffling interval), we generate new sets of CoAs and re-randomize the user groups and update them with the new CoAs. In Figure 1, we illustrate the basic scheme. On the server-side, $n$ different CoAs are utilized with three IPs ($IP_1$, $IP_2$ and $IP_n$) explicitly shown. Each CoA is assigned to a randomly chosen group of users. After shuffling, the users will be re-grouped and will be connected via newly generated CoAs. The efficacy of the basic scheme depends on the number of CoAs utilized and the number of agents of the adversary pretending to be normal users. These agents will seek out the CoAs in order to block them. By changing the CoAs, and shuffling the users, we provide a moving target. By utilizing a novel, cost-based shuffling method, we make it cost-inhibitive for the adversary to try to out number the system. Our method leverages the existing MIPv6 technology, such that no modifications to existing MIPv6 standard/protocol is required.

In our earlier paper [11], we presented the basic framework and analysis.[1] In this paper, in addition to editorial changes, we also include the following significant extensions/updates:

- Updated explanation and analysis.
- Co-location of mirrors with other sites.
- User registration mechanism: (1) coalition with uncensored sites (2) multiple reCAPTCHA with random timing for user registration control.
- Use of IPSec with Internet Key Exchange (IKEv2).
- Behavior monitoring and intelligent user grouping based on credit/risk based accounting that allows significant reduction in the number of CoAs needed during operation.
- Signaling overhead reduction.

The remainder of this paper is organized as follows. Over the next two sections, we provide an overview of the related work in anti-censorship efforts, moving target defense, and some details of MIPv6. We then present the basic framework of our solution, including an analytical model, and results of testing with a prototype implementation. We also present improvements to the framework and explain the benefits. Finally, we offer some conclusions, and discuss our ongoing and future works.

## II. Related Work

In this section, we discuss two categories of related work: (1) Anti-censorship methods and (2) MTD methods.

### A. Anti-Censorship Methods

End users utilize proxies and encrypted tunnels to get around the problem of censorship. Some notable examples include VPNs [2]–[4], open HTTPS proxies, and different types of anti-censorship tools such as Tor [5]. Unfortunately, the adversary can thwart the use of these solutions by blocking the IP addresses of the systems on which these services run. For example, traffic to and from Tor relays can be easily filtered, since their IP addresses are publicly advertised. The maintainers of the Tor project have thus suggested the use of bridges, which through relay-like functionality, do not advertise their presence and reachability information through directory services. However, encrypted bridge traffic can also be identified [12]. Thus, systems such as Skypemorph [13] and Stegotorus [14], suggest ways of camouflaging Tor messages through various unrelated, cover protocols (e.g., VoIP). These camouflaging methods can also be detected [15].

Recently, a solution called decoy routing was introduced [16]. Unlike traditional methods where the proxies reside at the ends of the network paths, this solution relies on placing the proxies in the middle. There are several variations that can be found in the literature including Cirripede [17],

[1]ACM Cyber and Information Security Research Conference at the Oak Ridge National Laboratory in Oak Ridge, TN, April 6, 2016.

TapDance [18], and Telex [19]. Internet service providers (ISP) that participate in decoy routing place special routing equipment that filter out specially marked messages (which are destined to decoy destinations) and redirects them towards the actual destination (which is censored). The realization of these schemes is dependent on the ability of the participating ISPs to filter traffic between the end-user and the decoys. The filtering process is not computationally cheap, requiring the addition of hardware resources. This overhead also results in increased latency. Though TapDance is designed to function without this blocking at the ISPs, it is vulnerable to active attacks that do not affect other schemes. Current decoy routing systems are also vulnerable to traffic analysis or website fingerprinting. Finally, in [20], a scheme for routing adversaries against the decoy routing strategy was introduced. The authors showed that the end-users can be forced onto paths lacking the special routers from the friendly ISPs. The use of several different class of solutions (including the one we propose in this paper) in conjunction is likely to yield a stronger solution.

### B. MTD Methods

We briefly discuss two different MTD-based methods that protect servers against attacks (e.g., DDoS) and discuss why they cannot be used (at least in their existing form) to address anti-censorship.

First, a cloud-based defense mechanism was introduced in [8] for Internet services against DDoS attacks. This solution was based on performing selective server replication and intelligent client reassignment, where the victim servers were turned into moving targets for attack isolation. The attacked server instances are replaced with new replicas at different network locations, and the clients are migrated to the new server instances. The attacked servers are recycled after client migration is completed. The new locations are only known to clients that have been migrated to them. The DNS service is used to redirect incoming clients to the cloud domains where the protected servers are deployed.

To reach the protected server, a client first needs to go through the DNS for domain resolution. In this step, the DNS refers the client to the cloud domain, where load balancers are used to redirect the clients. Each load balancer keeps records about the active replica servers within the same domain and redirects new clients to the servers. To achieve redirection-based load balancing, the load balancer replies to each client's requests with the unique network location of the server that the client now belongs to. The server is also notified; the client is then whitelisted on the server. If there is a DDoS attack on some of the servers, new replicas are instantiated and the clients are reassigned across the set of replacement servers.

We now discuss the limitations in adapting this method for anti-censorship (instead of using it for DDoS protection). The shortcomings can be summarized as:

- The censors can perform (1) IP blocking and/or (2) DNS filtering and redirection to prevent access to the load balancers.
- Detecting who the censor is amongst the normal clients is difficult; All users grouped with a censor are blocked perpetually.
- DDoS requires large volume of attacks to target the servers, where as a single censor can bring down a replica upon detecting the true location; it leads to much more frequent migrations.
- The time overhead of the redirection operation associated with the migration causes packet losses.
- A large amount of spare cloud infrastructure capacity is needed to allow large scale replication of servers.

Using out-of-band methods, such as email, can help get around the problem with access to the load balancers. Dynamic group arrangements could be performed each time new IPs are needed for replica servers to help alleviate the issue of having some users constantly being blocked as a result of their grouping. The remaining three limitations, however, pose serious challenges in utilizing this approach. We leveraged some of the insight gained in looking at these problems to design our solution.

The second MTD scheme of interest is MT6D [9]. MT6D is a form of a dynamic, network layer MTD that rapidly changes IPv6 addresses of both the sender and receiver mid-session without dropping or renegotiating sessions. The design takes advantage of IPv6 for new address binding. MT6D creates dynamic Interface IDentifier (IID) obscuration to create dynamic IP addresses. These IIDs are comprised of three parts: (1) a value specific to an individual host (seed IID), (2) a secret (symmetric) key shared between both parties, and (3) some changing value that is agreed upon by both parties (e.g., time). Out-of-band is suggested for sharing of the seed IID and the shared key. MT6D encapsulates each packet using Unreliable Datagram Protocol (UDP) to hide the original IP addresses, and uses virtual IPs. Although some initial steps have been taken to extend MT6D to support client-server based networking, it was focused solely on peer-to-peer networks. The much more common client-server networks are left untreated [21]. The limitations of MT6D are as follows:

- Packet loss due to address collision. As the IP addresses of the hosts are dynamically changed, a host must check to see that the new IP address is free. If an address collision occurs, the source and the destination will remain disconnected during that rotation interval.
- The use of shared keys between server and all clients (out-of-band) has scalability implications.
- Relatively tight time synchronization is needed.
- Organization of users to virtual IPs of a server is the most limiting factor for MT6D. It has two options for this allocation: (1) one virtual IP per each user, which is not scalable, and (2) one virtual IP per each predefined group of users. In the second option, the users that get grouped with a censor will be perpetually blocked.

## III. Background

In this section we discuss how MIPv6 is used. We also discuss route optimization and multiple care-of address registration as related to our use.

### A. Use of Mobile IPv6

We utilize MIPv6 to take advantage of several of its features, but the more important part is the ability for it to change how a Mobile Node (MN) is reached with a changing IP address as it moves through the network. Although we do not assume the use of any mobile nodes in our intended application, we treat the server as if it were a mobile node. In MIPv6, a MN has a permanent IP address, Home Address (HoA), assigned by the Home Agent (HA). HA is a router on the MN's home link that functions similar to a proxy for the MN. A MN also has an alternate address: Care-of Address (CoA), which is used by others, called correspondent nodes (CN), to reach MN. The HA keeps track of the CoA, and performs the necessary forwarding. The MN is responsible for updating the HA using Binding Update (BU) messages that contain new CoAs. In response, Binding Acknowledgement (BA) messages are received. CNs contact MN via HoA, which is processed by the HA and tunneled to MN.

### B. Route Optimization

Route Optimization is used to route packets via the shortest possible path between a MN and a CN. It necessitates the CN to hold the MN's current binding information. Thus, the MN must update the CN with the CoA. A return routability procedure is used to verify the claimed CoA [10] and the permission over the use of HoA. This procedure involves four messages. Following this procedure, route optimization uses two additional messages (BU and BA). These overhead messages are reduced by requiring only CoA tests in [22].

In [23], all messages related to the return routability tests are eliminated via the use of a shared symmetric key. We propose the use of this method, which has some significant advantages for our framework. First, low signaling overhead for route optimization minimizes handoff delays, which in turn mitigates packet loss during shuffling. Second, more importantly, without the need for return routability procedure, the HA is not involved in route optimization. The MN can update a CN with a new CoA directly. As a result, the HA can be disconnected. The removal of HA is a new way to handle this process that we will use in our approach, and both of these points are further explained later. Along with these advantages, the static shared key method also has some limitations:

- The CN need to trust the actions of the MN, and needs to assume that the MN will not launch flooding attacks against a third party as described in [24].
- Shared symmetric keys between a MN and each CN are needed. Replay attacks are possible. To address this issue, we propose to use IPsec with Internet Key Exchange (IKE) between the MN and CNs.

After running the route optimization mechanism, packets will be routed directly to the MN's CoA by a CN. To send a packet to any IPv6 destination, type 2 routing and destination options headers are used to route the packet to/from the MN [25].

The type 2 routing header is a routing header type for MIPv6. This routing header is used by a HA or a CN to carry a MN's HoA when packets are sent to the MN's CoA. For example, if a CN knows the MN's CoA, the CN can send a packet to the CoA but the MN needs to see its HoA in the destination IP address. Therefore, the CN stores the MN's HoA in the type 2 routing header and then sends the packet with the MN's CoA as the destination IP address. When the MN receives this packet, it automatically replaces the destination IP address of the packet with the address stored in the type 2 routing header.

The destination options header is used to carry optional information that needs to be processed only by a destination node. The important part of this option is home address option. It is used in a packet sent by a MN while away from home, to inform the CN of the MN's HoA. In this situation, the source address of the packet is the CoA of the MN and the address in the home address option is the HoA of the MN. These addresses will be swapped when the validity of the addresses is verified. The pair of the CoA and the HoA of the MN must be registered as a binding cache entry.

### C. Multiple Care-of Address

One of the keys to moving target defense as used for anti-censorship purposes is the ability to have many IP addresses such that the censor cannot detect them in a reasonable amount of time and block them. (We use and discard these addresses as explained later in Section 4). With MIPv6, with the Binding Identification (BID) number extension, a MN can utilize multiple CoAs (over the same HoA) with its HA and/or CNs. An MN can setup multiple IPv6 global addresses and register them as its CoAs. To register multiple bindings, the MN can generate a unique BID per CoA. These BIDs are stored in the Binding Update List, and used to handle each binding independently. The MN can register its CoAs using BU messages via the Binding Identifier mobility option.

On the other hand, the Multiple Care-of Addresses registration can be disabled on the CNs. the CNs then do not understand the BID mobility option found in the BU messages they received. As such, according to RFC 5648 [26], the CNs can skip the unknown mobility option and simply update the binding cache and will send packets to the newly updated CoA of the MN.

## IV. Using MIPv6-Based MTD for Anti-Censorship

In this section, we first describe our MIPv6-based Moving Target Defense (MI-MTD) scheme and present an analytical model to help reason about the various factors that affect the effectiveness of the scheme. We then discuss a user registration scheme, and present an optimization scheme that aids in improving the effectiveness of our approach. Finally, we present our experimental results to demonstrate the feasibility of using MI-MTD.

As described briefly above, the core of this approach involves the use of multiple IPv6 CoAs. Unlike actual mobile applications, however, the host (web server) is treated as the mobile node: the HoA is used as the permanent address of the server and the CoAs are used as the dynamic addresses. The CoAs are assigned to groups of users called *access groups*. We generate pseudo-random IP addresses to dynamically rotate all CoAs of the server after each time interval. During each *shuffling interval*, each access group membership is randomly changed by shuffling and redistributing the users.

The binding update mechanism is used to update users with the new CoAs.

Note that HAs and real mobility are not needed in our scheme. We just need to set the MIPv6 parameters in the server and select an IP address for the HoA with a prefix different from that of the server's subnet. When MIPv6 is run, the server will receive the route advertisement message from its router, where the prefix of the home link will be different from the prefix of the HoA. As such, the server will think that it is in a foreign network and will register a CoA in this network. To randomly generate the CoAs, 64 bit addresses are created at random and combined with the home link prefix to generate the new CoAs. These new CoAs are checked against existing occupancy via neighbor solicitation messages before they are registered on the subnet. According to the multiple CoA registration rules of MIPv6, the server (acting as if it were the MN) will send BU messages to its users to inform them of the new CoAs. When each user receives the BU, the HoA and CoA of the server are inserted into the binding cache. The server also removes the previous CoAs.

On the server-side some changes are needed to the implementation of MIPv6 to (1) allow the allocation of users to different access groups and (2) update each group by its allocated CoA. No changes are needed on the user side, and the MIPv6 protocol standard is also not changed.

Two components are critical to the scheme we described: the allocation and shuffling of multiple CoAs. First, allocating different CoAs to each access group limits the impact of having a censor (an adversary who may be pretending to be a normal user) within a group. Once a censor discovers a CoA, it will utilize IP blocking to cut-off access to the server that may occur via the CoA in question. All users that are in the same access group as this censor would lose their connection to the server. Second, to alleviate this problem, shuffling is used and each user is randomly rotated through the different access groups during each shuffling interval. To eliminate packet loss during address shuffling, we propose to have the server send a BU message for its new CoAs before removing the previous CoAs. Therefore, during the handoff delay, packets sent by users will have the old CoA in their header and will be received by the server. Each of the previous CoAs can be removed by the server after receiving the respective BAs from all users in each access group or after a certain amount of time. In our analysis and initial test implementation, we used a static shuffling interval, but it is possible to use a dynamic value. Dynamic intervals may be utilized to decrease any potential overhead associated with the binding update mechanism, and is part of our planned research.

The adversary may attempt to circumvent our scheme by blocking the entire subnet for each of the servers. We envision a coalition amongst web content providers and/or network domain operators, such that web server mirrors can be placed in various subnets in conjunction with other unblocked content (that is desirable to the adversary). The drawback is that such cooperative agreement between independent entities may not be possible. If cooperation can be achieved, this participation of the friendly domain is relatively cheap compared to ISP participation in other methods (e.g., decoy-routing): it does not require the friendly domain to perform any special handling of the target servers' traffic; they just need to allow placement of some servers (that act as mirrors for the target website) in their subnet. There are several interesting implications and potential solutions (such as compensation models), which could be researched and modeled. We leave this for future work.

TABLE I
SUMMARY OF NOTATIONS

| | |
|---|---|
| $N$ | Total # of users (including censors) |
| $N_a$ | # of adversary's agents (censors) |
| $N_u$ | # of (innocent) users |
| $I$ | # of IPs (CoAs) used per interval |
| $A_j$ | # of users assigned to IP$_j$, $j = 1$ to $I$ |
| $P_j$ | Probability that IP$_j$ is **not** blocked |
| $N_{ub}$ | # of users grouped with censors |
| $N_{uf}$ | # of users in censor-less groups |
| $p$ | (Instantaneous) Access probability |
| $b_k$ | Blocking probability over $k$ intervals |
| $t$ | duration of one shuffling interval (time) |
| $\phi$ | swarming ratio |

As discussed earlier, our goal is to show that it is technically feasible and relatively low-cost to implement our scheme. At the same time, we will show that it is prohibitively expensive for the adversary. The *swarming ratio*, derived in the next section, helps us reason about these two issues.

## *A. Model and Analysis*

In order to understand the efficacy of the MI-MTD scheme, we need to analyze the effects of having the adversary's agents (hereinafter referred to as censors) masquerading as normal end-users. Each censor will receive BU messages with a new CoA after each shuffling interval; This CoA will then be blocked or attacked by the adversary. During each interval, end-users that reside in an access group that contain at least one censor (by random chance) would be blocked from accessing the server. We are first interested in the probability of being blocked at any given moment based on the number of CoAs and the number of censors.

A mathematical model of the shuffling process is used to help us reason about the performance. The notations used in this model are summarized in Table I.

The total number of users, $N$, is the sum of the number of censors ($N_a$) and normal users($N_u$): $N = N_a + N_u$. At each shuffling interval, the normal users can be divided into two categories—those that are grouped with censors ($N_{ub}$) and those that are not grouped with censors ($N_{uf}$): $N_u = N_{ub} + N_{uf}$. We compute access probability, $p$, which is the probability that a user has access to the server at any given moment. To compute $p$, we first need to compute the expected value of the number of users assigned to groups without censors ($N_{uf}$):

$$N_{uf} = \sum_{j=1}^{I} P_j A_j \tag{1}$$

We assume (1) equal-sized access groups, which means that the users are uniformly distributed over the available CoAs, and (2) that $N$ is divisible by $I$. Therefore,

$$A_j = \frac{N}{I} \quad \text{for all } j \tag{2}$$

and the probability that any given CoA, $IP_j$, is not blocked is:

$$P_j = \frac{\binom{N-N_a}{A_j}}{\binom{N}{A_j}} \tag{3}$$

where $\binom{N-N_a}{A_j}$ is the number of ways that censors are not in group $A_j$ and $\binom{N}{A_j}$ is the number of ways to select $A_j$ users from the population of $N$. Therefore, we have:

$$E\left[N_{uf}\right] = \sum_{j=1}^{I} \frac{\binom{N-N_a}{A_j}}{\binom{N}{A_j}} A_j \tag{4}$$

Based on the assumptions stated above, $A_j = A_i, \forall i, j \in (1, I)$ and $N = I \times A_j$, which leads to:

$$E\left[N_{uf}\right] = N \times \frac{\binom{N-N_a}{A_j}}{\binom{N}{A_j}} \tag{5}$$

Using an approach similar to [7], based on Stirling's Approximation, $n! \approx \left(\frac{n}{e}\right)^n \sqrt{2\pi n}$, and assuming $N_a \ll N$:

$$E\left[N_{uf}\right] = N\left(\frac{N-N_a}{N}\right)^{A_j} = N\left(1 - \frac{N_a}{N}\right)^{N/I} \tag{6}$$

Therefore, the ratio of users that have access, which represents the probability $p$ with which a user has access to the server at any given moment, is [2]:

$$p = \frac{N}{N_u}\left(1 - \frac{N_a}{N}\right)^{N/I} \tag{7}$$

For example, if we have 1,000,000 users along with 5,000 censors, and 10,000 CoAs are used at each shuffling interval, the probability of having access to the server during one shuffling interval for any given user is about 60.88%.

Given the probability of access for each interval, we can reason about other important metrics based on practical considerations such as website access patterns that occur in real-life. For example, a user accessing certain information on a website does not typically interact with the web server in a constant/continuous manner. Meaning, the user does not constantly click and load web pages (and related web resources such as images). The pattern of access typically involves interaction with the server (short duration) followed by actual reading of the information on the user's own machine (longer duration). As such, it is useful to investigate how much inconvenience the user will experience in trying to get to the information. Due to our shuffling methodology, the probability of access (at any given moment) is an independent probability, which allows us to easily compute the blocking probability over a time period $\delta$. The time period $\delta$ is some multiple of the shuffling interval ($t$). Hence, we derive the number of intervals in question, $k = \lceil \delta/t \rceil$, and denote the blocking probability over this period as $b_k$. We can compute $k$ independent trials during which the user is always grouped with a censor:

$$b_k = (1-p)^k, \quad k = \left\lceil \frac{\delta}{t} \right\rceil \tag{8}$$

For example, we can look at a period of one minute and calculate the blocking probability over that period. This measures the probability that the user is denied access to the server for the *entire period*. For shuffling interval, $t$, of 10 seconds, $b_6 = (1 - 0.6088)^6 \approx 0.358\%$. In other words, the user has ~99.6% chance of getting access (every minute).

The overall effect of having our solution in place depends on the relationship between the key parameters. Primarily, we define the ***swarming ratio***, $\phi$, as $(\frac{N_a}{I})$. The swarming ratio is a critical metric: it determines the access and blocking probabilities, and is not affected by the scale. In other words, changing the $I$ and $N_a$ values does not affect the access probability as long as the swarming ratio remains constant. Furthermore, changing $N_u$ also does not affect the worst case access probability for any given swarming ratio. It is trivial to see that if $N$ is less than $I$, users are guaranteed access. As $N$ increases, the access probability converges. For example, with $\phi = 0.5$ from the previous example, the access probability, $p$, is never less than 60.65%. Given a sufficiently large number of users, we can always compute the lower bound, which is a function of the swarming ratio. Hence, the swarming ratio is a key metric in our analysis.

Based on the equations introduced above, we also define what we term the *dominating point* for the swarming ratio, which is where $p$ becomes too low for the system to be useful. What exactly is considered useful depends on the intended use of the application. For instance, under extreme conditions, it may be good enough to be able to access the website just once after several hours of attempts. In contrast, typical web browsing for news articles may require some access every few minutes or so. Hence, we must ask the following question: What kind of censorship is the adversary interested in? Is it interested in complete blockage, or is it just trying to cause some level of inconvenience?

For illustration purposes, we choose 5% blocking over the period of one minute. Let us say that the system will be considered useless if the user cannot access the website with more than 95% probability after a minute of trying. In practice, it would only be seen as a mere inconvenience or nuisance. Users, especially under heavy censorship, will tolerate a much higher blocking rate. Also, as the period of consideration increases from our one-minute example, the odds become better for the user. We chose this scenario since it presents a more challenging case for our approach. *The goal of our proposed framework is to make censorship efforts much more expensive for the adversary.*

In the current working example, the dominating point occurs when $\phi = 0.9339$. Figure 2 shows the effect of changing the swarming ratio. With this result, we can explore the underlying problem space given IPv6 and the limitations of the hardware resources available for a website. The total number of

[2]We empirically validated the result against the expanded form of Equation 4, and the differences were negligible.

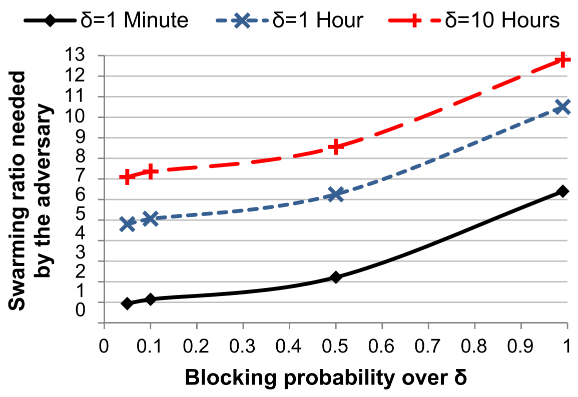


Fig. 2. Swarming ratio vs. Blocking over $\delta$.

IP addresses a web-server can utilize within a single subnet is $2^{64}$. The total number of CoAs a server can utilize during each shuffling interval (within each subnet) depends on two limiting factors. First, if all $2^{64}$ IPs are used, then the server will exhaust all possible addresses within one interval, and as a result, the access groups will not be able to receive new, never-before-assigned, CoAs. However, it is unlikely that a server system will utilize all of the $2^{64}$ IPs at the same time as the number is simply too large given current technology. (We provide a discussion of what is currently feasible with an example in the next section.) If 100,000 CoAs are used in each interval, it would take more than $5.849\times10^{7}$ years to exhaust all usable addresses. This figure represents the *lifespan* of the system. Second, the creation and binding performance of the IPs to be used as CoAs at each shuffling interval is limited by the hardware of the server and the router. Based on our experiments (and what others have experimented with in the literature [21]), we used the value $I = 10,000$ for our example because working with 10,000 simultaneous IPs did not incur large latencies. Given this value, to achieve a swarming ratio of 0.9339, $Na$ has to be $0.9339 \times 10,000 = 9,339$. The adversary must utilize 9,339 censors to cause a mere 5% blocking over each minute of users' access attempts. Using the same parameters, for complete blockage (greater than 99%) over 10 hours (such to prevent a user from downloading even a single news article over a period of 10 hours), the adversary needs to achieve a swarming ratio of 12.8 by deploying 128,000 censors. Using the 5-minute registration example from earlier, the adversary in this case needs 1,067 humans to complete the blocking process by the *end of the 10-hour period*, meaning before the 10 hours is up, people have chance to access the website. To achieve complete blockage for 10-hours, the adversary would need to setup 128,000 censors at the start of the period in question. If we give the adversary a one hour window, 10,667 people are needed. The process has to repeat when the server resets the users. Also, ramping up the number of CoAs across all mirrors should be (1) relatively trivial in terms of the man-power required, and (2) cost-efficient for the website operators. We aim to demonstrate these ideas in this paper. On the other hand, the cost for censoring should quickly become unmanageable for the adversary. This cost should become even more unmanageable with a large number of websites that the adversary tries to block. (Figure 2 illustrates the swarming ratio required for the adversary to achieve different levels of blocking for three different intervals of user access attempts.)

This discussion outlines the cost struggle between the two competing sides. The server must deploy a sufficiently large number of CoAs and the adversary must deploy a sufficiently large number of censors, with each side fighting to control the swarming ratio.

## B. User Registration Procedure

In practice, the adversary does not need $Na$ physical computers to create $Na$ censors. Similar to the server's configuration, an adversary may create a system by binding a large number of IPs, which in turn can be used as censors. According to the experimental results in [21], it is possible to have 55,000 IPv6 addresses bound to a single computer in suitable time. The example IP binding capability of 10,000 per each shuffling interval of 10 seconds was obtained from the same results. Their experiment was performed on a non-server-grade computer with Intel i7-4770 processor running at 3.4Ghz with 16GB of RAM and an Intel I217-LM Gigabit Ethernet network interface card. Using machines with similar computing power, the adversary can theoretically implement 55,000 censors. As discussed earlier, to combat automated censor deployment, a method for requiring human response should be utilized. For the end-user, this is not a problem since it only needs to be performed at initial registration. Some of the suggested out-of-band methods include difficult CAPTCHA-like tests or mini games that require a non-trivial amount of time for a human to solve.

The registration status should be reset at some time interval (e.g., every 12 hours), which forces the adversary to repeat the registration process for every single censor. Another important factor in censor deployment is the use-duration of a censor. If a censor was deployed immediate after the new registration cycle opens, it is potentially useful for the entire cycle. If a censor was deployed at the 11th hour, that censor is only useful for the next hour, after which the registration would be reset globally. Since humans cannot solve massive numbers of challenge tests in parallel (without equally multiplying the time it takes to do them), the average use-duration of a censor (if the adversary spends the entire cycle solving challenge questions without any breaks) would be half the cycle time. To cause a complete block over a 10 hour period, with a 12 hour registration cycle and a one-minute solving time for the challenge tests, the adversary could prepare 128,000 censors during the first two hours after registration resets. Then, during the next 10 hours, no user would be able to access the server. For this scenario, the adversary needs 1,067 human agents to work for the full two hours. On the server-side, we can relatively easily add additional combinations of computers, network interfaces, and routers. In practice, web server sites consist of many computers. To combat a 10 server site configuration each with 10 server-interfaces, the adversary needs 106,700 humans working in parallel. Furthermore, during the first two hours, a large number of users would still be able to access the server.

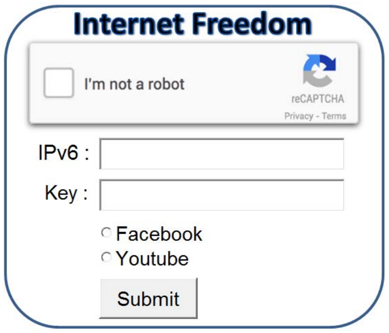


Fig. 3. Sample user registration window on uncensored sites.

Because there are no HAs (by design), a new user will not be able to start a connection to the server using the HoA of the server. Instead, the connection initiation is made by the server upon receiving a request from a user. On the user side, the host file needs to be set using the HoA and domain name of the server. To initiate a connection from a new user to the server, the user must send its IP address and a shared key to the server. One simple method is to utilize a secure email exchange for this issue. However, some users may not have access to secure email exchange. For example, users from certain countries may have access to Google's search engine, but cannot use Gmail. Therefore, other methods for registering new users are needed.

With the help of friendly web servers, we can address this issue as follows. In this approach, the friendly websites can volunteer to add a small script to assist in the key exchange/registration process. Figure 3 shows a mock-up of what would show up in the friendly websites' that are assisting with "Internet Freedom." A new user will be required to solve a CAPTCHA and put his IPv6 address through the "Internet Freedom" window. The user then needs to create a key that will be used for starting IKEv2, explained in the next subsection, and select one of the censored servers that he or she wishes to connect to (in the example figure, Facebook and Youtube are shown). The script will be designed to send a packet to the censored server. This packet will contain the IPv6 and Key of the user. The censored server can then use the standard MIPv6 procedure to start the route optimization mechanism and update the user with one of its active CoAs.

To determine whether the user is human or an automated bot a challenge puzzle that requires significant human intervention (e.g., CAPTCHA test) is needed. One suggestion that we have, as shown in Figure 3, is the use of the reCAPTCHA that is available for free, has API support and the test comprises of a simple click on a checkbox. On suspicious clicks, it diverts to more traditional CAPTCHA tests. To combat the adversary's use of large-scale automated censor operation, we need a method that makes it difficult for both a human and a computer to solve within a certain time window. Utilizing only the reCAPTCHA test, one such solution can be constructed. For example, we propose the use of a sequence of reCAPTCHAs with random down time, the time during which the user waits for the next reCAPTCHA to come up, and a short time window to perform the reCAPTCHA once it actually does come up. We can also use random screen location for each subsequent of reCAPTCHAs to prevent automatic clicking after the first human click. During this process, users have to quite actively engage themselves with the registration "puzzle" in order to not miss the time window. Changing the number of reCAPTCHAs in the sequence and the random down time can help adjust the time requirement for registration. This registration process should be configured such that the users are reset after a certain amount of time as explained previously. In this way, large-scale automated censor operation requires significant human resources to operate. For example, with five minutes spent per registration, a human working non-stop for 24 hours straight can only manage up to 288 censors. Our discussion of the numbers holds here as well. We later describe a novel approach to further shift the ratio in the server's favor, aiding in both the strength and the scalability of our framework.

## C. IPsec With IKEv2

When a node (N1) wants to send a data packet to another node (N2), it will have already supplied the HoAs as the source and destination addresses in the packet header. Next, N1 checks the binding update list to see if it has already sent a BU to N2, and search for the CoA from N2's entry. If it is found, N1 includes its CoA in the home address option. N1 then checks its Binding Cache to see if N2 has sent it a BU. If it is found, N1 constructs a type 2 routing header and places the N2's CoA inside it. The packet with HoAs in the source and destination addresses of the header will reach IPsec. After encrypting and adding headers, the home address option is swapped with the source address and the type 2 routing header is swapped with the destination address of the packet header. When the packet is received by N2, the headers are processed in the order they appear in the packet, and the IPsec implementation always sees HoAs in the source and destination addresses of the packet header [27, p. 116–119]. IPsec does not encrypt the type 2 routing header and the home address option that show HoA of the source and destination.

To prevent any censoring we can resolve this problem by removing the destination option header (and the type 2 routing header) from all packets. Note that the Security Parameter Index (SPI) found in the IP Encapsulating Security Payload (ESP) header is sufficient to get access to the HoA (the real source/destination of the packet). The packet format before and after removing destination option header/the type 2 routing header are shown in Figure 4.

We also propose the use of standard key management techniques. IKEv2 can be used to increase the security and scalability, and prevent replay attacks. To start IKEv2, some negotiations need to be performed between peers, and then SPIs will be defined (a total of four messages). After those steps, we can remove the HoA and use SPIs. The friendly server, the uncensored server used for user registration, needs to encapsulate the first packet of IKEv2 between a user and the server. Note that the key the user needs to put in the Internet Freedom window is to be used for IKE_AUTH Exchange, which is the first step for IKEv2. The details are out of the scope of this paper, and can be found in the standards document for IKEv2.

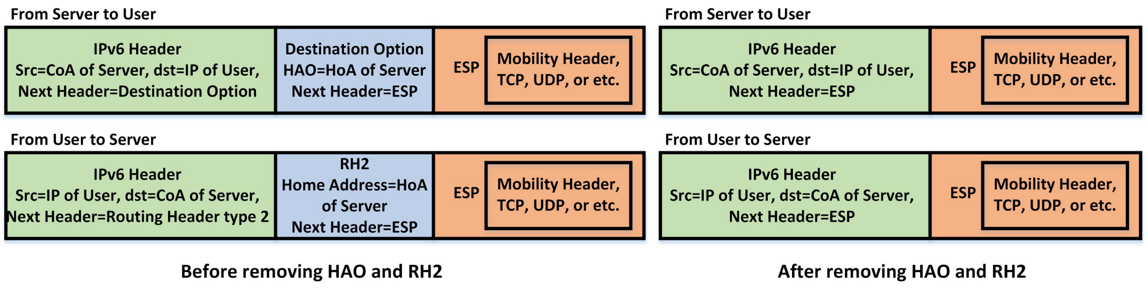


Fig. 4. Packet format before and after removing destination option header/type 2 routing header.

### *D. Optimization: Behavior Monitoring and Credit Based Accounting Method*

In this section, we introduce a novel approach to address the scalability issue mentioned above. We reduce the number of CoAs and utilize dynamic shuffling interval(s). Behavior monitoring is also performed, where we organize the users and group them according to the country that they belong to. This information can be obtained via the IPv6 prefix. The server can then check the status of one of its CoAs by sending ping commands to a pre-established point of contact (or to some webservers) in the same country that the users of the particular CoA group belong to. If a response is not received, the IP is blocked by a censor.

Instead of competing only through the strength of the computer/network system to create and operate with a large number of CoAs, we dynamically shift group membership based on a notion of credit assigned to each user. Basically, the credit value represents the amount of "trust" the system can place in a user.

By default, the credit ($C$) for each new user starts at zero. After each shuffling interval, the server checks each of its CoAs to see if they are blocked. If a CoA is not blocked, $C$ is increased for each user in the group associated with the CoA. The amount by which $C$ is increased is based on the number of users in the group ($C_{new} = C_{old} + (G - 1)$).

We utilize $C$ to combine groups together, which results in the creation of a larger group. When we combine groups, we are assuming that the group members are innocent users. Another change from the basic framework is that the server no longer shuffles groups whose IP addresses are not blocked. The server can maintain a lookup table with two columns—group size ($G$) and the *risk* of each group ($R(G)$). New users with zero credit are assigned to small size groups with size $G_{min}$, and the risk for these smallest groups is zero. At each shuffling interval any (larger) group that is blocked is divided into two groups. This process is continued, and could go on until reaching the minimum group size for some group.

The risk of a group is defined as the maximum cost of having a censor in this group. For example, assume that $G_{min} = 2$ and we have a group of eight users consisting of seven innocent users and one censor. Assume that the censor waited a while and starts to block (at which point the group size has reached eight). The censor will block seven innocent users $(8 - 1)$ for one shuffling interval. After the server splits the group, the censor will be assigned to one of the two groups of four users, at which point it can block three innocent users. In the next shuffling interval, the group with the censor inside will reach the minimum size, with one innocent victim. Therefore the cost of having a censor in a group of eight users is $(7+3+1 = 11)$. This cost is used as the risk of groups. The notations are summarized in Table II. With some calculations we can obtain that if the size of a group ($G$) is $G_{min} \times 2^j$, the risk of this group is:

$$R(G) = \sum_{i=0}^{j-1} \left( \frac{G}{2^i} - 1 \right) \tag{9}$$

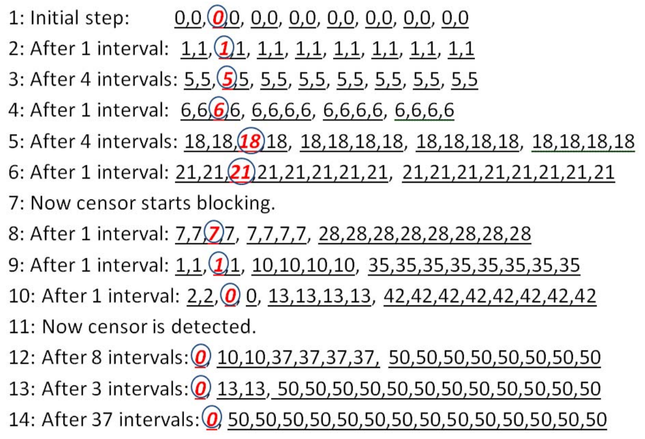
TABLE II
CREDIT BASED METHOD NOTATIONS

| | |
|---|---|
| $C$ | Credit for each user |
| $G$ | Group size |
| $G_{min}$ | Minimum group size |
| $G_{max}$ | Maximum group size |
| $R(G)$ | Risk of group $G$ |

Note that the size of the lookup table the server has to maintain is relatively small. For example, if $G_{min} = 100$ and the maximum group size ($G_{max}$) equals 1,638,400 then the number of rows in the table is equal to $log_2\left(\frac{1638400}{100}\right) = 14$ that is small enough for quick search to find suitable group for a user with a specific credit value.

A censor may pretend to be an innocent user in order to get into a bigger group before starting to block. To make it expensive for censors to wait, a user is added to a group only if the user's credit is twice the risk of the group. When the server detects that the IP of a group is blocked and divides the group as described above, the credit for each user is decreased by $2 \times (G-1)$. Note that the credit starts from zero and increases until twice $G_{max}$.

In the example shown in Figure 5, we have one censor (whose credit value is circled and highlighted in red, bold/italic font) that is waiting to be in a large group. First the server generates eight CoAs for the groups. Note the server does not need to create new IPs when it is merging two groups. Assume the censor starts blocking when it is in a group of eight users (line 6 in the example). At this point, the server will create two IPs and divide the group in question into two groups and continue this process until finding the censor.



1: Initial step: 0,0,0,0, 0,0, 0,0, 0,0, 0,0, 0,0, 0,0
2: After 1 interval: 1,1,1,1, 1,1, 1,1, 1,1, 1,1, 1,1, 1,1
3: After 4 intervals: 5,5,5,5, 5,5, 5,5, 5,5, 5,5, 5,5, 5,5
4: After 1 interval: 6,6,6,6, 6,6,6,6, 6,6,6,6, 6,6,6,6
5: After 4 intervals: 18,18,18,18, 18,18,18,18, 18,18,18,18, 18,18,18,18
6: After 1 interval: 21,21,21,21,21,21,21,21, 21,21,21,21,21,21,21,21
7: Now censor starts blocking.
8: After 1 interval: 7,7,7,7, 7,7,7,7, 28,28,28,28,28,28,28,28
9: After 1 interval: 1,1,1,1, 10,10,10,10, 35,35,35,35,35,35,35,35
10: After 1 interval: 2,2,0 0, 13,13,13,13, 42,42,42,42,42,42,42,42
11: Now censor is detected.
12: After 8 intervals: 0, 10,10,37,37,37,37, 50,50,50,50,50,50,50,50
13: After 3 intervals: 0, 13,13, 50,50,50,50,50,50,50,50,50,50,50,50
14: After 37 intervals: 0, 50,50,50,50,50,50,50,50,50,50,50,50,50,50

Fig. 5. Credit based method example.

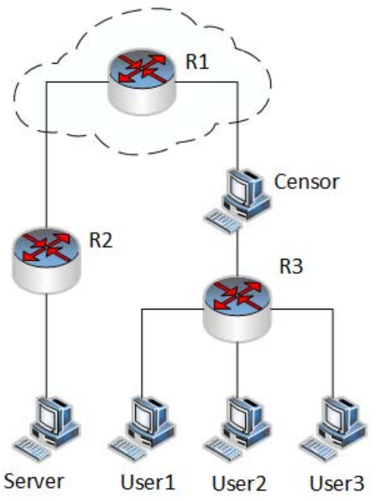


Fig. 6. The network topology of the testbed.

## E. Prototype Implementation

A proof of concept prototype of the first version of our method was implemented to illustrate the validity of the basic mechanics of the approach and to evaluate the performance of the design. To implement our method, we used an IPv6 testbed configured as shown in Figure 6. We used three routers (R1, R2, and R3) and five computers (2.4GHz Intel Core 2 Duo CPU with 4GB DDR2 800MHz RAM) running Ubuntu 14.04. Linux kernel (version 3.8.2) with the mobility options enabled was compiled and installed on the computers. An open source implementation of MIPv6 (UMIP) for Linux was used. Router R1 is used to emulate the heart of the Internet. Packet forwarding is enabled on the censor to work like a router. In this first version we used one CoA for the single access group of users (user1, user2, and user3). In this setup, the censor is not in the group.

After preparing the UMIP configuration files (mip6d.conf) to setup parameters such as IPsec (with static keys between the server and the users) and acceptance of route optimization, for the server and the users, the mobility daemons were run. Note that the server's HoA does not have the same prefix with the advertised prefix of R2. The server registered a CoA on R2 and updated all users with its CoA. We used a program to create a new CoA every 10 seconds (and remove the previous one). According to the MIPv6, the server sends the BU message to its users to inform them of its new CoA. The ACK bit of BU messages forces users to send back acknowledgment messages in response to BU messages. This process ensures that the users receive the BU messages and update their binding cache entries. Therefore, in every 10-second interval, we have two overhead packets between the server and each user as shown in Figure 7. Normally, during this update procedure, the users are unable to access the server until they receive the BU messages. We resolved this problem by using the multiple CoA registration on the server-side. That is, the CoA of one interval is kept alive until the next CoA has been received by all users. It is removed once all users are rotated on to the new CoA, and the process repeats.

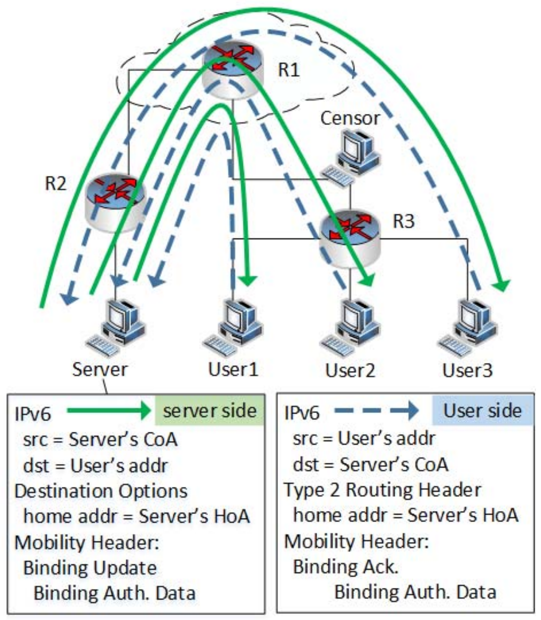


Fig. 7. The Binding Update process.

There are two types of overheads of the proposed method—overhead caused by the updating procedure because of the BU and BA messages (signaling overhead) and overhead in each data packet transmission between the server and a user (transmission overhead):

- *Signaling Overhead:* A complete correspondent node registration needs two message transmissions at the server (BU and BA messages) with each being 110 bytes (using IPsec and removing destination option and the type 2 routing headers). Note that in the original MIPv6, the length of each BU and BA is 110 bytes, and because of using return routability mechanism, we have four extra messages. Therefore, the overhead of route optimization in our method is only about one-third of the original MIPv6. Also, to minimize the number of messages generated and reduce the overhead, it is possible to operate the system without using BAs. If data is received from the client to the new CoA, the server knows the client received the BU without needing an explicit BA. Since the standard protocol already has a flag that lets the other side know whether or not to return a BA, no modifications are needed to utilize this method.

- *Transmission Overhead:* For each data packet, we have 24 bytes of overhead due to the use of IPsec (ESP). Note that without using our method we still need to use IPsec to have secure connection between the server and users so the real overhead caused by our method per each data packet is zero.

## V. Conclusion and Future Work

In this paper, we presented an anti-censorship framework using moving target defense designed with Mobile IPv6. Our analysis showed, based on an important metric called swarming ratio, that it is possible use our scheme to aid in making it difficult or impractical for the adversary to deploy censorship techniques. Our approach to shuffling optimization with a novel user grouping strategy further strengthens our framework. The solution leverages the standard MIPv6 protocol and does not require changes in the network protocol nor use third party network elements. We also demonstrated the feasibility of deployment with a lab-based test of MIPv6 protocol with the home agent completely removed from the update process.

We are currently experimenting with various optimizations, such as the use of multicasting, on the server-side implementation to further minimize overhead and improve performance. To this end, our future experiments will include large-scale testing with multiple interfaces. We believe that the anti-censorship measure would be extremely robust if our scheme was used in conjunction with other anti-censorship methodologies. Specifically, since our approach is host-side, it lends itself well to a combination with end-user or end-to-middle schemes. With participation and support from friendly websites across the globe, we feel that our framework can add tremendous strength to the effort of making the Internet accessible for all.

## References

[1] R. Davies, *Broadband as a Universal Service*. Apr. 2016. [Online]. Available: http://www.europarl.europa.eu/RegData/etudes/BRIE/2016/581977/EPRS_BRI(2016)581977_EN.pdf
[2] *F-Secure Switch on Freedom*, accessed on Apr. 10, 2015. [Online]. Available: https://www.f-secure.com/en_US/welcome
[3] *Free VPN Service Free VPN Software—Hotspot Shield VPN*, accessed on Apr. 10, 2015. [Online]. Available: http://www.hotspotshield.com/
[4] *Psiphon Uncensored Internet Access For Windows and Mobile*, accessed on Apr. 10, 2015. [Online]. Available: https://psiphon3.com/en/index.html
[5] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proc. USENIX Secur. Symp.*, 2004, p. 21.
[6] A. Stavrou, A. D. Keromytis, J. Nieh, V. Misra, and D. Rubenstein, "Move: An end-to-end solution to network denial of service," in Proc. Netw. Distrib. System Secur. Symp. (NDSS), 2005.
[7] H. Wang, Q. Jia, D. Fleck, W. Powell, F. Li, and A. Stavrou, "A moving target DDoS defense mechanism," *Comput. Commun.*, vol. 46, pp. 10–21, Jun. 2014.
[8] Q. Jia, H. Wang, D. Fleck, F. Li, A. Stavrou, and W. Powell, "Catch me if you can: A cloud-enabled ddos defense," in *Proc. IEEE/IFIP Dependable Syst. Netw.*, Jun. 2014, pp. 264–275.
[9] M. Dunlop, S. Groat, W. Urbanski, R. Marchany, and J. Tront, "Mt6d: A moving target IPv6 defense," in *Proc. AFCEA/IEEE MILCOM*, Nov. 2011, pp. 1321–1326.
[10] C. Perkins, D. Johnson, and J. Arkko, *Mobility Support in IPv6*, document RFC 6275, Internet Requests for Comments, Jul. 2011.
[11] V. Heydari, S.-I. Kim, and S.-M. Yoo, "Anti-censorship framework using mobile IPv6 based moving target defense," in *Proc. ACM Cyber Inf. Secur. Res. Conf.*, 2016, Art. no. 7.
[12] P. Winter and S. Lindskog, "How the great firewall of China is blocking tor," presented at the 2nd USENIX Workshop Free Open Commun. Internet., Berkeley, CA, USA, 2012. [Online]. Available: https://www.usenix.org/conference/foci12/workshop-program/presentation/Winter
[13] H. M. Moghaddam, B. Li, M. Derakhshani, and I. Goldberg, "Skypemorph: Protocol obfuscation for tor bridges," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 97–108.
[14] Z. Weinberg *et al.*, "Stegotorus: A camouflage proxy for the tor anonymity system," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 109–120.
[15] A. Houmansadr, C. Brubaker, and V. Shmatikov, "The parrot is dead: Observing unobservable network communications," in *Proc. IEEE Symp. Secur. Privacy*, May 2013, pp. 65–79.
[16] J. Karlin *et al.*, "Decoy routing: Toward unblockable internet communication," in *Proc. USENIX Workshop Free Open Commun. Internet (FOCI)*, 2011, pp. 1–6.
[17] A. Houmansadr, G. T. Nguyen, M. Caesar, and N. Borisov, "Cirripede: Circumvention infrastructure using router redirection with plausible deniability," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2011, pp. 187–200.
[18] E. Wustrow, C. M. Swanson, and J. A. Halderman, "TapDance: End-to-middle anticensorship without flow blocking," in *Proc. USENIX Secur. Symp.*, 2014, pp. 159–174.
[19] E. Wustrow, S. Wolchok, I. Goldberg, and J. A. Halderman, "Telex: Anticensorship in the network infrastructure," in *Proc. USENIX Conf. Secur.*, 2011, pp. 1–15.
[20] M. Schuchard, J. Geddes, C. Thompson, and N. Hopper, "Routing around decoys," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 85–96.
[21] C. Morrell, J. Ransbottom, R. Marchany, and J. Tront, "Scaling IPv6 address bindings in support of a moving target defense," in *Proc. Internet Technol. Secur. Trans. (ICITST)*, Dec. 2014, pp. 440–445.
[22] J. Arkko, C. Vogt, and W. Haddad, *Enhanced Route Optimization for Mobile IPv6*, document RFC 4866, Internet Requests for Comments, May 2007.
[23] C. Perkins, *Securing Mobile IPv6 Route Optimization Using a Static Shared Key*, document RFC 4449, Internet Requests for Comments, Jun. 2006.
[24] P. Nikander, J. Arkko, T. Aura, G. Montenegro, and E. Nordmark, *Mobile IP Version 6 Route Optimization Security Design Background*, document RFC 4225, Internet Requests for Comments, Dec. 2005.
[25] D. Johnson, C. Perkins, and J. Arkko, *Mobility Support in IPv6*, document RFC 3775, Internet Requests for Comments, Jun. 2004.
[26] R. Wakikawa, V. Devarapalli, G. Tsirtsis, T. Ernst, and K. Nagami, *Multiple Care-Of Addresses Registration*, document RFC 5648, Internet Requests for Comments, Oct. 2009.
[27] H. Soliman, *Mobile IPv6*. Reading, MA, USA: Addison-Wesley, 2004.

**Vahid Heydari** (S'12) received the B.S. and M.S. degrees in computer engineering and the M.S. degree in cybersecurity from the University of Science & Culture, Tehran, Iran, in 2005, Payame Noor University, Tehran, in 2013, and The University of Alabama in Huntsville in 2016, respectively. He is currently pursuing the Ph.D. degree in electrical and computer engineering with The University of Alabama in Huntsville. His research interests include moving target defenses, mobile adhoc, sensor, and vehicular networks security. He is a Student Member of the ACM and the IEEE Computer Society and Communications Society.

**Sun-il Kim** (SM'15) received the B.S. degree from Binghamton University, State University of New York, in 2000, and the M.S. and Ph.D. degrees in computer science from the University of Illinois at Urbana–Champaign in 2001 and 2008, respectively. He was with The University of Alabama in Huntsville, the University of Alaska Anchorage, and the University of St. Thomas. He is currently an Associate Professor of Computer Science with the North Central College, Naperville, IL. His primary research interests center on reliability and security in networked systems.

**Seong-Moo Yoo** (SM'03) received the M.S. and Ph.D. degrees in computer science with The University of Texas at Arlington. He was an Assistant Professor with Columbus State University, Columbus, GA, USA. He is currently an Associate Professor of Electrical and Computer Engineering with The University of Alabama in Huntsville. He has co-authored over 100 scientific articles in refereed journals and international conferences. His research interests include computer network security and wireless network routing. He is a member of the ACM.