Keeping Denial-of-Service Attackers in the Dark

Gal Badishi, Amir Herzberg, and Idit Keidar

Abstract—We consider the problem of overcoming (distributed) denial-of-service (DoS) attacks by realistic adversaries that have knowledge of their attack's successfulness, for example, by observing service performance degradation or by eavesdropping on messages or parts thereof. A solution for this problem in a high-speed network environment necessitates lightweight mechanisms for differentiating between valid traffic and the attacker's packets. The main challenge in presenting such a solution is to exploit existing packet-filtering mechanisms in a way that allows fast processing of packets but is complex enough so that the attacker cannot efficiently craft packets that pass the filters. We show a protocol that mitigates DoS attacks by adversaries that can eavesdrop and (with some delay) adapt their attacks accordingly. The protocol uses only available efficient packet-filtering mechanisms based mainly on addresses and port numbers. Our protocol avoids the use of fixed ports and instead performs "pseudorandom port hopping." We model the underlying packet-filtering services and define measures for the capabilities of the adversary and for the success rate of the protocol. Using these, we provide a novel rigorous analysis of the impact of DoS on an end-to-end protocol and show that our protocol provides effective DoS prevention for realistic attack and deployment scenarios.

Index Terms—Protocols, reliability, availability, serviceability.

1 Introduction

Denial-of-service *(DoS)* attacks have proliferated in recent years, causing severe service disruptions [6]. The most devastating attacks stem from *distributed DoS (DDoS)*, where an attacker utilizes multiple machines (often thousands) to generate excessive traffic [16]. Due to the acuteness of such attacks, various commercial solutions and off-the-shelf products addressing this problem have emerged. The main goal of all solutions is to provide lightweight packet-filtering mechanisms that are adequate for use in high-speed networks, where per-packet analysis must be efficient.

The most common solution uses an existing firewall/router (or protocol stack) to perform *rate limiting* of traffic and to *filter* messages according to header fields like address and port number. Such mechanisms are cheap and readily available and are therefore very appealing. Nevertheless, rate limiting indiscriminately discards messages, and it is easy to spoof (fake) headers that match the filtering criteria: An attacker can often generate spoofed packets containing correct source and destination Internet Protocol (IP) addresses and arbitrarily chosen values for almost all fields used for filtering.[1] Therefore, the only hope in using such efficient filtering mechanisms to overcome DoS attacks lies in choosing values that are *unknown to the adversary*. For example, the Transmission Control Protocol (TCP)'s use of a random initial sequence number is a simple version of this approach but is inadequate if the attacker has some (even limited) eavesdropping capability.

More effective DoS solutions are provided by expensive commercial devices that perform stateful filtering [19], [20], [21]. These solutions specialize in protecting a handful of commonly used stateful protocols, for example, TCP; they are less effective for stateless traffic such as the User Datagram Protocol (UDP) [21]. Such expensive solutions are not suitable for all organizations.

Finally, the most effective way to filter out offending traffic is using secure source authentication with message authentication codes (MACs), as in IP security (IPsec) [3]. However, this requires computing a MAC for every packet, which can induce significant overhead and, thus, this approach may be even more vulnerable to DoS attacks. Specifically, it is inadequate for use in high-speed networks with high volumes of traffic.

Our goal is to address DoS attacks on end hosts, for example, in corporate networks, assuming that the network leading to the hosts is functional. (A complementary solution protecting the end network can be deployed at the ISP.) In this paper, we focus on fortifying the basic building block of two-party communication. Specifically, we develop a DoS-resistant datagram protocol, similar to UDP or raw IP. Our protocol has promising properties, especially in overcoming realistic attack scenarios where attackers can discover some of the control information included in protocol packets, as also described in [1]. We assume that a realistic adversary can detect whether its attack is successful or not and adjust its behavior accordingly. However, this adjustment takes some time, as it involves gathering information from the system, processing it to decide on the proper adjustment and then notifying all the attacking nodes (massive attacks employ many nodes). We believe that our ideas, with some practical adjustments,

1. An exception is the TTL field of IP packets, which is automatically decremented by each router. This is used by some filtering mechanisms, for example, BGP routers that receive only packets with maximal TTL value (255) to ensure that the packets were sent by a neighboring router and the Hop Counter Filtering proposal [9].

• *G. Badishi and I. Keidar are with the Department of Electrical Engineering, Technion, Haifa 32000, Israel. E-mail: badishi@tx.technion.ac.il, idish@ee.technion.ac.il.*
• *A. Herzberg is with the Department of Computer Science, Bar Ilan University, Ramat Gan 52900, Israel. E-mail: amir.herzberg@gmail.com.*

Manuscript received 3 Aug. 2006; revised 23 Apr. 2007; accepted 22 May 2007; published online 18 June 2007.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-0115-0806.
Digital Object Identifier no. 10.1109/TDSC.2007.70209.

have the potential to find their way into future DoS protection systems. For example, these ideas can be integrated into IPsec [3]. Our formal analysis proves the effectiveness of our ideas and, thus, shows that their realization into a working system is highly beneficial.

The key to exploiting lightweight mechanisms that can filter high-speed traffic is using a *dual-layer approach*: On the one hand, we exploit cheap, simple, and readily available measures at the network layer; on the other hand, we leverage these network mechanisms to provide sophisticated defense at the application layer. The latter allows for more complex algorithms as it has to deal with significantly fewer packets than the network layer and may have closer interaction with the application. The higher layer dynamically changes the filtering criteria used by the underlying layer, for example, by closing certain ports and opening others for communication. It is important to note that the use of dynamically changing ports instead of a single well-known port does not increase the chance of a security breach, as a *single* application is listening on *all* open ports.

The main contribution of our work is in presenting a *formal framework* for understanding and analyzing the effects of the proposed solutions to the DoS problem. The main challenges in attempting to formalize DoS resistance for the first time are coming up with appropriate models for the attacker and the environment, modeling the functionality that can be provided by underlying mechanisms such as firewalls, and defining meaningful metrics for evaluating suggested solutions. We capture the functionality of a simple network-level DoS mitigation solution by introducing the abstraction of a *port-based rationing channel*. It is important to note that our use of ports just serves as an example. In fact, any field that appears on all packets can be used as the filtering criterion, and our analysis and suggested protocol apply to all such fields. For simplicity, we henceforth use the term "port" to refer to any filtering criterion that can be dynamically changed by the application level. Our primary metric of an end-to-end communication protocol's resistance to DoS attacks is the *success rate*, which is the worst case expected portion of valid application messages that successfully reach their destination, under a defined adversary class.

Having defined our model and metrics, we proceed to give a generic analysis of the communication success rate over a port-based rationing channel in different attack scenarios. We distinguish between *directed* attacks, where the adversary knows the port used, and *blind* attacks, in which the adversary does not know the port. Not surprisingly, we show that directed attacks are extremely harmful: With as little as 100 machines (or a sending capacity 100 times that of the protocol) the success rate is virtually zero. On the other hand, the worst-case success rate that an attacker can cause in blind attacks in realistic scenarios is well over 90 percent even with 10,000 machines.

Our goal is therefore to "keep the attacker in the dark" so that it will have to resort to blind attacks. Our basic idea is to change the filtering criteria (that is, ports) in a manner that cannot be predicted by the attacker. This *port-hopping* approach mimics the technique of a frequency-hopping spread spectrum in radio communication [22]. We assume that the communicating parties share a secret key unknown to the attacker; they apply a pseudorandom function [8] to this key in order to select the sequence of ports they will use. Note that such port hopping has negligible effect on the communication overhead for realistic intervals between hops and thus can be used even in high-speed networks. The remaining challenge is synchronizing the processes so that the recipient opens the port currently used by the sender. We present a protocol for doing so in a realistic partially synchronous model, where processes are equipped with bounded-drift bounded-skew clocks, and message latency is bounded.

The paper proceeds as follows: Section 2 details related work. Section 3 details our models for the communication channel and the adversary. Section 4 provides a generic DoS analysis. Section 5 describes our port-hopping protocol and analyzes its effectiveness. Section 6 concludes.

2 Related Work

Our work continues the line of research on prevention of DDoS attacks, which focuses on filtering mechanisms to block and discard the offending traffic. Our work is unique in providing a rigorous model and analysis, which constitute the first step in formally modeling and evaluating the effectiveness of possible filtering and rate-limiting mechanisms. Since our formal framework is not restricted to port-based filtering but rather operates with any filtering based on per-packet fields, our model and analysis can be used in evaluating future protocols and may assist in examining and comparing the solutions that exist now.

Most closely related is the work on Secure Overlay Services (SOS) [12], followed by the work on Mayday [1]. Both propose realistic and efficient mechanisms that do not require global adoption yet allow a server to provide services immune to DDoS attacks. These solutions, like ours, utilize efficient packet-filtering mechanisms between the server and predefined trusted "access point" hosts. The basic ideas of filtering based on ports or other simple identifiers ("keys") and even of changing them already appear in [1], [12] but without analysis and details. Additionally, Andersen [1] provides a discussion of attack types and limitations, justifying much of our model, including the assumption that the exposure of the identifier (port) number may be possible but not immediate. Furthermore, Andersen [1] mentions blind and targeted attacks (where blind attacks are attacks in which the adversary does not know the valid identifier) and asserts that the damage to the system is much more severe when targeted attacks are launched. We prove that this is indeed the case and give exact quantities for the maximum performance degradation in both attack scenarios. Both SOS and Mayday require the setup of an overlay network consisting of several nodes and use several levels of indirection to obscure the identity of the nodes that may prove to be a promising attack target. These levels of indirection may increase latency by a factor of 5 or even 10 [12]. In contrast, our solution does not require additional hosts, preserves communication characteristics, and is simple to construct and maintain.

Additional work [23] employs an overlay network similar to SOS, which uses spread-spectrum-like path diversity to counter DoS attacks. The system also uses secret keys to authenticate valid messages. Like SOS, it requires additional nodes to construct the overlay network, and the additional overhead has an impact on message throughput and latency.

There are other several proposed methods to filter offending DoS traffic. Some proposals, for example, that by Krishnamurthy et al. [13], [10], filter according to the source IP address. This is convenient and efficient, allowing implementation in existing packet-filtering routers. However, IP addresses are subject to spoofing; furthermore, using a white list of source addresses of legitimate clients/peers is difficult, since many hosts may have dynamic IP addresses due to the use of a network address translator (NAT), Dynamic Host Configuration Protocol (DHCP), and mobile-IP. Some proposals try to detect spoofed senders using new routing mechanisms such as "path markers" supported by some or all of the routers en route, as in Pi [25], Stateless Internet Flow Filter (SIFF) [26], Active Internet Traffic Filtering (AITF) [2], and Pushback [15], but global router modification is difficult to achieve. Few proposals try to detect spoofed senders using only existing mechanisms, such as the hop count (Time-to-Live (TTL)), as in Hop-Count Filtering (HCF) [9]. However, empirical evaluation of these approaches show rather disappointing results [5].

A different approach is to perform application-specific filtering for predefined protocols [11], [18]. Such protection schemes are cumbersome, only work for a handful of well-known protocols, and are usually restricted to attackers that transmit invalid protocol packets.

IPsec [3] performs filtering at the IP layer, by authenticating messages using MACs, based on shared secret keys. IPsec ensures that higher level protocols only receive valid messages. However, the work required to authenticate each message is invested for each incoming packet that has a valid security parameter index (SPI). Once the SPI, which is sent in the clear, is known, an attacker can perform a DoS attack by overloading IPsec with many bogus packets to authenticate. In contrast, our solution ensures that the authentication phase is reached only for packets that are valid with high probability by constantly changing the cleartext filtering identifier, for example, the SPI.

In earlier work, we have presented Drum [4], a gossip-based multicast protocol resistant to DoS attacks. Drum does not use pseudorandom port hopping, and it heavily relies on well-known ports that can be easily attacked. Therefore, Drum is far less resistant to DoS attacks than the protocol we present here. Finally, Drum focuses on multicast only, and as a gossip-based protocol, it relies on a high level of redundancy, whereas the protocol presented herein sends very little redundant information.

Independent of our work, Lee and Thing [14] examined the use of port hopping to mitigate the effect of DoS attacks. However, they concentrated more on implementation and empirical results, providing only a very brief analysis of their method. Even so, their empirical results do not state the strategy the attacker employs for its attack, and it is not clear whether the adversary cannot launch a better attack against their protocol. Conversely, we provide a thorough formal analysis of the environment and our protocol. We formally model the communication channel and the adversary and provide rigorous proofs for the correctness and effectiveness of our protocol under the best attack the adversary can possibly launch.

Wang et al. [24] provide simulation results for various DDoS attacks on general proxy networks and the applications protected by them. However, they do not provide any theoretical analysis and only deal with general proxy networks.

3 Model and Definitions

3.1 Overview

We consider a realistic *semisynchronous* model, where processes have continuously increasing local clocks with bounded drift Φ from real time. Each party may schedule events to occur when its local clock reaches a specific value (time). There is a bound Δ on the transmission delay, that is, every packet sent either arrives within Δ time units or is considered lost. Notice that although we assume that messages *always* arrive within Δ time, this is only a simplification, and our results are valid even if a few messages arrive later than that; therefore, Δ should really be thought of as the typical maximal round-trip time and not as an absolute bound on a message's lifetime (for example, 1 second rather than 60 seconds).

Our goal is to send messages from a sender A to a recipient B in spite of attempts to disrupt this communication by an adversary. The basic technique available to the adversary is to clog the recipient by sending many packets. The standard defense deployed by most corporations is to rate-limit and filter packets, typically by a firewall. We capture this type of defense mechanism using a *port-based rationing channel* machine, which models the first-in, first-out (FIFO) communication channel between A and B, as well as the filtering mechanism. To send a message, A invokes a *ch_send(m)* event, a message is received by the channel in a *net_recv(m)* event, and B receives messages via *ch_recv(m)* events. We assume that the adversary cannot clog the communication to the channel and that there is no message loss other than in the channel. The channel discards messages when it performs rate limiting and filtering.

The channel machine is formally defined in Section 3.2. We now provide an intuitive description of its functionality. Since we assume that the attacker can spoof packets with valid addresses, we cannot use these addresses for filtering. Instead, the channel filters packets using port numbers, allowing deployment using existing efficient filtering mechanisms. Specifically, let the set Ψ of port numbers be $\{1, \dots, \psi\}$. Our solutions can be used with larger values of ψ; however, this may require modified filtering mechanisms. The buffer space of the channel is a critical resource. The channel's interface includes the *alloc*action, which allows B to break the total buffer space of R messages into a separate allocation of R_i messages per port $i \in \Psi$, as long as $R \geq \sum_{i=1}^{\psi} R_i$. For simplicity, we assume that the buffers are read and cleared together in a single *deliver* event, which occurs exactly once on every integer time unit. If the number of packets sent to port i since the last *deliver* exceeds R_i, a uniformly distributed random subset of R_i of them is delivered.

We define several parameters that constrain the adversary's strength. The most important parameter is the *attack strength* C, which is the maximal number of messages that the adversary may inject to the channel between two *deliver* events.

As shown in [1], attackers can utilize different techniques to try to learn the port numbers expected by the filters (and used in packets sent by the sender). However, these techniques usually require considerable communication and time. To simplify, we allow the adversary to eavesdrop by exposing messages, but we assume that the adversary can expose packets no earlier than $\mathcal{E}$ time after they are sent, where $\mathcal{E}$ is the *exposure delay* parameter. The exposure delay reflects the time it takes an attacker to expose the relevant information, as well as to distribute it to the (many) attacking nodes, possibly using a very limited bandwidth (for example, if sending from a firewalled network). Our protocol works well with as little as $\mathcal{E} > 5\Delta$.

Since the adversary may control some behavior of the parties, we take a conservative approach and let the adversary schedule the *app_send(m)* events in which the application (at A) asks to send m to B. To prevent the adversary from abusing these abilities by simply invoking too many *app_send* events before a *deliver* event, we define the *throughput* $T \geq 1$, as the maximal number of *app_send* events in a single time unit. We further assume that $R \geq \Delta T$, that is, that the capacity of the channel is sufficient to handle the maximal rate of *app_send* events.

Since we focus on connectionless communication such as UDP, our main metric for resiliency to DoS attacks is its *success rate*, namely, the probability that a message sent by A is received by B.

Definition 1 (Success rate μ). *Let E be any execution of a given two-party protocol operating over a given port-based rationing channel with parameters Ψ, R, C, Φ, Δ, $\mathcal{E}$, and T, with adversary ADV. Let $end(E)$ be the time of the last* deliver *event in E. Let $sent(E)$ ($recv(E)$) be the number of messages sent (respectively, received) by the application, in app_send (respectively, app_recv) events during E, prior to $end(E) - \Delta$ (respectively, $end(E)$). The* success rate *μ of E is defined as $\mu(E) = \frac{recv(E)}{sent(E)}$. The success rate of adversary ADV is the average success rate over all executions of ADV. The success rate of the protocol, denoted $\mu(\Psi, R, C, \Phi, \Delta, \mathcal{E}, T)$, is the worst success rate over all adversaries ADV.*

Finally, a protocol can increase its success rate by sending redundant information, for example, multiple copies or error-correcting codes. We therefore also consider a system's *message* (*bit*) *complexity*, which is the number of messages (respectively, redundant bits) sent on the channel per each application message.

3.2 Formal Model and Specifications

We model the system as a collection of interacting state machines. Each state machine is defined by its state (variables), set of possible initial states, and deterministic state transitions associated with input and output events. To allow machines to make random choices, initial states include random tapes.

We model the adversary as one of the deterministic state machines of which the system is composed. The adversary controls, among other things, the scheduling of events. That is, it defines the next event that will occur in any system state, as well as the progress of time (via the *advance* event). Thus, an *execution* of the system is completely defined by its initial state and number of steps.[2] The possible choices of random tapes define a probability space on executions.

A *port-based rationing channel* models a FIFO-ordered rate-limited communication channel with port-based message filtering. Fig. 1 provides specifications for a channel from A to B; we assume that an equivalent channel is used from B to A. The *net_recv* event models the arrival of the next message from A (in FIFO order) to the channel's buffer, allowing the adversary control of network latency (up to Δ).

The recipient uses the *alloc* operation to designate ration values R_i for ports $i \in \Psi = \{1, \dots, \psi\}$. If $R_i > 0$, we say that port i is *open*. We use $In(i)$ to denote the set of messages in the input buffer designated with port i. The channel delivers all messages from $In(i)$ if $|In(i)| \leq R_i$ and a random subset of R_i messages from $In(i)$ if $|In(i)| > R_i$.

The adversary can inject messages directly into the buffer using *inj* events and can snoop on the contents of messages using *expose* events, under the restrictions above.

4 Analyzing the Success Rate in a Single Slot with a Single Port

This section provides a generic analysis of the probability of successfully communicating over a port-based rationing channel under different attacks, when messages are sent to a single open port p. This analysis is independent of the timing model and the particular protocol using the channel and can therefore serve to analyze different protocols that use such channels, for example, the one we present in the ensuing section. We focus on a single $deliver$ event and analyze the *channel's delivery probability*, which is the probability for a valid message in the channel's buffer to be delivered in that event. Since every *ch_send(m)* event eventually results in m being added to the channel's buffer, we can use the channel's delivery probability to analyze the success rates of higher level protocols.

Let R_p denote the ration allocated to port p in the last *alloc* event and let $In(p)$ be the contents of the channel's buffer for port p (see Section 3.2 for more details). Consider a $deliver$ event of a channel from A to B when A sends messages only to port p. We introduce some notations:[3]

- $R_p = R$ is the value of the channel's R_p when $deliver$ occurs.
- $a_p = a$ is the number of messages whose source is A in the channel's $In(p)$ when $deliver$ occurs. We assume that $a \leq R$. If $a_p < R_p$ (that is, $a < R$), we say that there is *overprovisioning* on port p.
- c_p is the number of messages whose source is *not* A in $In(p)$ when $deliver$ occurs.

Assume that $1 \leq a \leq R$. If $c_p < R - a + 1$, then B receives A's messages, and the attack does not affect the communication from A to B on port p. Let us now examine what happens when $c_p \geq R - a + 1$.

2. We encapsulate all nondeterminism and randomness in the choice of random tapes.

3. Note that *for simplicity of notation only*, we remove the $_p$ subscript from R_p and a_p. All results are valid with the subscripts in place as well.

```
VARIABLES:            $rcvd$, initially 0                                   // Number of last received message (for FIFO)
                      $m(i)_{i\in\mathbb{N}}$, initially $\emptyset$        // $i^{th}$ message sent
                      $port(i)_{i\in\mathbb{N}}$, initially $\emptyset$     // port of $i^{th}$ message
                      $t(i)_{i\in\mathbb{N}}$, initially $\emptyset$        // time when $i^{th}$ message was sent
                      $time$, initially 0                                   // Current time
                      $\{\ In(port)\ \}_{port\in\{1,\dots,\psi\}}$, initially $\emptyset$   // Buffer of messages to processor $q$, per port
                      $\{\ R_{port}\ \}_{port\in\{1,\dots,\psi\}}$          // Ration of each port, set by recipient
                      $sent$, initially 0                                   // Count of messages sent
                      $inj$, initially 0                                    // Count of messages injected since last deliver
HANDLING OF EVENTS:
On $ch\_send(m, port)$:   $m(\text{++}sent) \leftarrow m$,   $port(sent) \leftarrow port$,   $t(sent) \leftarrow time$
On $net\_recv$:           add $m(\text{++}rcvd)$ to $In(port(rcvd))$
On $alloc(port, r)$:      if $(R \geq r - R_{port} + \sum_{i=1}^{\psi} R_i)$ then $R_{port} \leftarrow r$
On $deliver$:             $inj \leftarrow 0$
                          for $port \in \{1,\dots,\psi\}$ do:               // Deliver up to $R_{port}$ messages from $In(port)$
                              let $M$ be random $R_{port}$ messages from $In(port)$
                              for $m \in M$ do: $ch\_recv(m, port)$
                              $In(port) \leftarrow \emptyset$               // Clear buffer
On $inj(m, port)$:        if $inj\text{++} \leq C$ then add $m$ to $In(port)$
On $expose(i)$:           if $time \geq t(i) + \mathcal{E}$ then return $\langle m(i), port(i)\rangle$
On $advance(\delta)$:     $time \leftarrow sent > rcvd\ \ ?\ \min\{time + |\delta|, t(rcvd+1) + \Delta\}\ :\ time + |\delta|$
```

Fig. 1. Port-based rationing channel for given Ψ, R, C, Φ, Δ, and $\mathcal{E}$.

Lemma 1. *If* $c_p \geq R - a + 1$, *then the channel's delivery probability is* $\frac{R}{c_p+a}$.

Proof. The channel delivers $m \in In(p)$ if it is part of the R messages read uniformly at random from the $c_p + a$ available messages. Thus, the delivery probability is $\frac{R}{c_p+a}$.□

If the attacker knows that B has opened port p, it can direct all of its power to that port, that is, $c_p = C$, where we assume that $C \geq R - a + 1$. We call this a *directed attack*.

Corollary 1. *In a directed attack at rate C on B's port p, the delivery probability for messages sent to the attacked port p is* $\frac{R}{C+a}$, *assuming that* $1 \leq a \leq R$, *and* $C \geq R - a + 1$.

Lemma 2. *For fixed R and* c_p *such that* $1 \leq a \leq R$ *and* $c_p \geq R - a + 1$, *the probability of B receiving only invalid messages on port p decreases as a increases.*

Proof. The channel delivers only invalid messages if no message of the a valid messages is read. The corresponding probability is $\frac{c_p}{c_p+a} \cdot \frac{c_p-1}{c_p+a-1} \cdots \frac{c_p-R+1}{c_p+a-R+1}$, which clearly decreases as a increases. □

4.1 Blind Attack

We define a *blind attack* as a scenario where A sends messages to a single open port p, and the adversary cannot distinguish this port from a random one. We now analyze the worst-case delivery probability under a blind attack.

In general, an adversary's strategy is composed of both timing decisions and injected messages. The timing decisions affect a, the number of messages from A that are in the channel at a given delivery slot. Given that a is already decided, we define the set of all strategies of an attacker with sending rate C as

$$S(C) \triangleq \Big\{ \{c_i\}_{i\in\Psi} \mid \forall i \in \Psi\ :\ c_i \in \mathbb{N} \cup \{0\}\ \wedge\ \sum_{i=1}^{\psi} c_i = C \Big\}.$$

Each strategy $s \in S$ is composed of the number of messages the attacker sends to each port. Note that since the adversary wishes to minimize the delivery probability, we restrict the discussion to the set of attacks that fully utilize the attacker's capacity for sending messages.

Consider some fixed a, C, and R. We define $\mu_B(a, C, R, s)$ as the channel's delivery probability under attack strategy $s \in S$. Since S is a finite set, μ_B has at least one minimum point, and we define the delivery probability to be that minimum:

$$\mu_B(a, C, R) \triangleq \min_{s\in S(C)} \mu_B(a, C, R, s).$$

We sometimes use μ_B instead of $\mu_B(a, C, R)$ when a, C, and R are clear from context. We want to find lower bounds on μ_B, depending on the attacker's strength. We say that port p_i is attacked in strategy s if $c_{p_i} > 0$. We partition $S(C)$ according to the number of ports being attacked as follows:

$$S_k \triangleq \{ s \in S(C)\ \mid\ \text{Exactly } k \text{ ports are being attacked in } s \}.$$

Consider a fixed $s_k \in S_k$ and denote by $p_1, p_2, \dots, p_k$ the ports that the attacker attacks under strategy s_k at rates of $c_{p_1}, c_{p_2}, \dots, c_{p_k}$ messages, respectively, where $\sum_{i=1}^{k} c_{p_i} = C$, $c_{p_i} > 0$. Then, we assume that $\forall i\ c_{p_i} \geq R - a + 1$ (otherwise, even if $p_i = p$, the probability of B receiving A's messages is exactly 1).

We now find a lower bound on μ_B as follows: We first derive a lower bound on $\{\mu_B(a, C, R, s_k) | s_k \in S_k\}$; this lower bound is given as a function of k in Corollary 2. Incidentally, the worst degradation occurs when the attacker divides its power equally among the attacked ports, that is, when it sends $\frac{C}{k}$ messages to each attacked port (this is proven in Lemma 3). Then, we show lower bounds on $\mu_B(a, C, R)$ by finding the k that yields the minimum value.

Proposition 1. *Consider some fixed k, a, C, R, and $s_k \in S_k$ and denote the ports attacked under s_k by $p_1, p_2, \ldots, p_k$ with attacking rates of $c_{p_1}, c_{p_2}, \ldots, c_{p_k}$, respectively. Then, $\mu_B(a, C, R, s_k) = \frac{\psi - k}{\psi} + \frac{1}{\psi}\sum_{i=1}^{k} \frac{R}{c_{p_i}+a}$.*

Proof. The probability that B does *not* deliver A's message is

$$\sum_{i=1}^{k} Pr[p_i = p] \cdot \left(1 - \frac{R}{c_{p_i}+a}\right) = \frac{1}{\psi}\sum_{i=1}^{k}\left(1 - \frac{R}{c_{p_i}+a}\right) = \frac{k}{\psi} - \frac{1}{\psi}\sum_{i=1}^{k}\frac{R}{c_{p_i}+a}.$$

Thus, the delivery probability is $\frac{\psi - k}{\psi} + \frac{1}{\psi}\sum_{i=1}^{k} \frac{R}{c_{p_i}+a}$. □

The proofs of the following lemmas appear in Appendix A.

Lemma 3. *Consider some fixed k, a, C, R, and $s_k \in S_k$ and denote the ports attacked under s_k by $p_1, p_2, \ldots, p_k$ with attacking rates of $c_{p_1}, c_{p_2}, \ldots, c_{p_k}$, respectively. Then, under a blind attack with strategy s_k, the worst (that is, minimal) expected delivery probability of the system is achieved when $\forall i\ c_{p_i} = \frac{C}{k}$.*

From Proposition 1 and Lemma 3, we get the following corollary:

Corollary 2. *Under a blind attack, if k, a, C, and R are fixed, then the expected delivery probability for $s_k \in S_k$ is bounded from below as follows:*

$$\mu_B(a, C, R, s_k) \geq \frac{\psi - k}{\psi} + \frac{1}{\psi}\cdot\sum_{i=1}^{k}\frac{R}{\frac{C}{k}+a} = \frac{\psi - k}{\psi} + \frac{1}{\psi}\cdot\frac{kR}{\frac{C}{k}+a} = \frac{\psi - k}{\psi} + \frac{k^2 R}{\psi(C+ka)}.$$

We now define $\mu_B(k) \overset{\Delta}{=} \min_{s_k \in S_k} \mu_B(s_k)$. We get that for each k

$$\mu_B(k) = \frac{\psi - k}{\psi} + \frac{R}{\psi}\cdot\frac{k^2}{C+ka}.$$

To find a lower bound, we continue this analysis as if k is continuous. The derivative of $\mu_B(k)$ is then

$$\begin{aligned}\mu'_B(k) &= \frac{-1}{\psi} + \frac{R}{\psi}\cdot\frac{2k(C+ka) - k^2 a}{(C+ka)^2}\\ &= \frac{R}{\psi}\cdot\frac{2kC + k^2 a}{(C+ka)^2} - \frac{1}{\psi}\\ &= \frac{R-1}{\psi} - \frac{R}{\psi}\cdot\frac{C^2 + (2kC + k^2 a)(a-1)}{(C+ka)^2}.\end{aligned}$$

We now state two lemmas that show that $\mu_B(a, C, R)$ is bounded from below by the function $f(a, C, R)$ presented in (1).

Lemma 4. *Let $R = a$. Then, an adversary with $C \geq \psi$ cannot decrease the expected delivery probability lower than $\frac{\psi a}{C + \psi a}$, and an adversary with $C \leq \psi$ cannot decrease the expected delivery probability lower than $1 - \frac{C}{\psi(1+a)}$.*

Lemma 5. *Let $a < R$. Then, an adversary with $C \geq \frac{\psi a}{\sqrt{\frac{R}{R-a}}-1}$ cannot decrease the expected delivery probability lower than $\frac{\psi R}{C+\psi a}$, and an adversary with $C \leq \frac{\psi a}{\sqrt{\frac{R}{R-a}}-1}$ cannot decrease the expected delivery probability lower than*

$$\frac{\psi a - C\left(\sqrt{\frac{R}{R-a}} - 1\right)}{\psi a} + \frac{R}{\psi}\cdot\frac{C\left(\sqrt{\frac{R}{R-a}} - 1\right)^2}{a^2\sqrt{\frac{R}{R-a}}}.$$

We conclude the following corollary:

Corollary 3. *$\mu_B(a, C, R)$ is bounded from below by the following function $f(a, C, R)$:*

$$f(a,C,R) = \begin{cases} \frac{\psi a}{C+\psi a} & \text{if } R = a \text{ and } C \geq \psi, \\ 1 - \frac{C}{\psi(1+a)} & \text{if } R = a \text{ and } C < \psi, \\ \frac{\psi R}{C+\psi a} & \text{if } R > a \text{ and } C \geq \frac{\psi a}{\sqrt{\frac{R}{R-a}}-1}, \\ \frac{\psi a - C\left(\sqrt{\frac{R}{R-a}}-1\right)}{\psi a} + \frac{R}{\psi}\cdot\frac{C\left(\sqrt{\frac{R}{R-a}}-1\right)^2}{a^2\sqrt{\frac{R}{R-a}}} & \text{if } R > a \text{ and } C < \frac{\psi a}{\sqrt{\frac{R}{R-a}}-1}, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Corollary 3 provides us with some insights of the adversary's best strategy and of the expected degradation in delivery probability. If no overprovisioning is used (that is, $R = a$), then the adversary's best strategy is to attack as many ports as possible. This is due to the fact that even a single bogus message to the correct port degrades the expected delivery probability. When the adversary has enough power to target all of the available ports with at least one message, it can attack with more messages per attacked port, and the delivery probability asymptotically degrades much like the function $\frac{1}{C}$. When not all ports are attacked, the adversary would like to use its remaining resources to attack more ports rather than target a strict subset of the ports with more than one bogus message per port. The degradation of the expected delivery probability is then linear as the attacker's strength increases.

When overprovisioning is used $(R > a)$, it affects the attack and its result in two ways. First, the attacker's best strategy may not be to attack as many ports as it can, since a single bogus message per port does not do any harm now. Second, for an adversary with a given strength, the degradation in delivery probability is lower when overprovisioning is used than when it is not employed. We can see in (1) that if the attacker has enough power to attack all the ports, the overprovisioning ratio $\frac{R}{a}$ is also the ratio by which the delivery probability is increased, compared to the case where $R = a$.

4.2 Actual Values

Fig. 2 shows the expected worst-case delivery probabilities for various attack scenarios on a single port. For directed attacks, we show the actual delivery probability, and for blind attacks, the lower bound $f(a, C, R)$ is shown. We

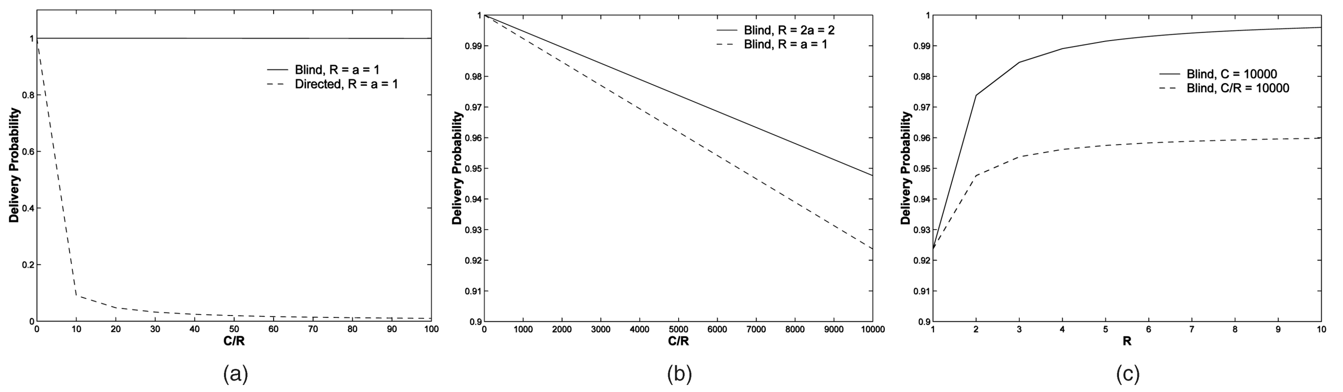


Fig. 2. Delivery probability per slot in various attack scenarios on a single port, $\psi = 65,536$. (a) Blind versus directed, $R = a = 1$. (b) Blind, $a = 1$. (c) Blind, $a = 1$.

chose $\psi = 65,536$, the number of ports in common Internet protocols, for example, UDP. Fig. 2a illustrates the major difference between a directed attack and a blind one: Even for a relatively weak attacker $(C \leq 100)$, the delivery probability under a directed attack approaches 0, whereas under a blind attack, it virtually remains 1.

Fig. 2b examines blind attacks by much stronger adversaries (with C up to 10,000 for $R = 1$ and up to 20,000 for $R = 2$). We see that the delivery probability gradually degrades down to a low of 92.5 percent when $R = 1$. If we use an overprovisioned channel, that is, have $a = 1$ (one message from A) when $R = 2$, the delivery probability improves to almost 95 percent for $C = 20,000$. (The ratio $\frac{C}{R}$ is the same for both curves.) Fig. 2c shows the effect of larger overprovisioning. We see that the cost-effectiveness of overprovisioning diminishes as $\frac{R}{a}$ increases.

The idea of hopping can essentially be applied to any changeable header field. For instance, other than the port numbers used in the TCP and UDP, one may decide to use the SPI field of IPsec, which consists of 32 bits, or the Key field of the Generic Routing Encapsulation (GRE), as suggested in WebSOS [17]. Fig. 3 shows the effect of hopping using IPsec's SPI field instead of using TCP/UDP ports. We can see that doubling the number of bits used for the filtering index has a substantial effect on the delivery probability. Using IPsec also has the added bonus of protecting all higher level protocols, for example, Internet Control Message Protocol (ICMP), TCP, UDP, and so forth.

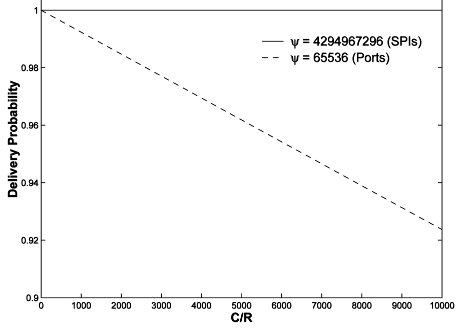


Fig. 3. Blind mode delivery probability per slot for different values of ψ, $R = a = 1$.

5 DoS-Resistant Communication

We now describe a protocol that allows for DoS-resistant communication in a partially synchronous environment. The protocol's main component is an acknowledgment (ack)-based protocol. B sends acks for messages it receives from A, and these acks allow the parties to hop through ports together. However, although the ack-based protocol works well as long as the adversary fails to attack the correct port, once the adversary identifies the port used, it can perform a directed attack that renders the protocol useless. By attacking the found data port or simultaneously attacking the found data and ack ports, the adversary can effectively drop the success rate to 0, and no port hopping will occur. To solve this matter, there is a time-based proactive reinitialization of the ports used for the ack-based protocol, independent of any messages passed in the system.

5.1 Ack-Based Port Hopping

We present an *ack-based port-hopping* protocol, which uses two port-based rationing channels, from B to A (with ration R_{BA}) and vice versa (with ration R_{AB}). For simplicity, we assume that $R_{AB} = 2R_{BA} = 2R$. B always keeps two open ports for data reception from A, and A keeps one port open for acks from B. The protocol hops ports upon a successful round-trip on the most recent port used, using a *pseudorandom function* PRF^*.[4] In order to avoid hopping upon adversary messages, all protocol messages carry authentication information, using a second pseudorandom function PRF on $\{0,1\}^\kappa$. (We assume that PRF and PRF^* use different parts of A and B's shared secret key.)

The protocol's pseudocode appears in Fig. 4. Both A and B hold a port counter P, initialized to some $seed$ (for example, 1). Each party uses its counter P in order to determine which ports should be open and which ports to send messages to. B opens port p_{old} using the $(P-1)$th element in the pseudorandom sequence and p_{new}, using P. A sends data messages to the Pth port in the sequence and

4. Intuitively, we say that $f_{key}(data)$ is *pseudorandom function* (PRF^*) if for inputs of sufficient length, it cannot be distinguished efficiently from a truly random function r over the same domain and range by a probabilistic polynomial-time (PPT) adversary that can receive $g(x)$ for any values of x, where $g = r$ with probability half and $g = f$ with probability half. For definition and construction, see [8].

```
PROTOCOL FOR SENDER $A$:

On $ack\_init(seed)$:
    $P = seed$
    $p_{ack} = PRF^*_{S_{AB}}(P|\text{“}ack\text{”})$
    $alloc(p_{ack}, R_{BA})$

On $app\_send(data)$:
    $m = data|PRF_{S_{AB}}(P|\text{“}data\text{”})$
    $ch\_send(m, PRF^*_{S_{AB}}(P|\text{“}data\text{”}))$

On $ch\_recv(ack, p_{ack})$:
    if $ack.auth = PRF_{S_{AB}}(P|\text{“}ack\text{”})$ then
        $alloc(p_{ack}, 0)$
        $p_{ack} = PRF^*_{S_{AB}}(P+1|\text{“}ack\text{”})$
        $alloc(p_{ack}, R_{BA})$
        $P = P + 1$

PROTOCOL FOR RECIPIENT $B$:

On $ack\_init(seed)$:
    $P = seed$
    $p_{old} = PRF^*_{S_{AB}}(P-1|\text{“}data\text{”})$
    $p_{new} = PRF^*_{S_{AB}}(P|\text{“}data\text{”})$
    $alloc(p_{old}, R_{AB}/2)$
    $alloc(p_{new}, R_{AB}/2)$

On $ch\_recv(m, p_{old})$:
    if $m.auth = PRF_{S_{AB}}(P-1|\text{“}data\text{”})$ then
        $app\_recv(m.data)$
        $ack = PRF_{S_{AB}}(P-1|\text{“}ack\text{”})$
        $ch\_send(ack, PRF^*_{S_{AB}}(P-1|\text{“}ack\text{”}))$

On $ch\_recv(m, p_{new})$:
    if $m.auth = PRF_{S_{AB}}(P|\text{“}data\text{”})$ then
        $app\_recv(m.data)$
        $alloc(p_{old}, 0)$
        $p_{old} = p_{new}$
        $p_{new} = PRF^*_{S_{AB}}(P+1|\text{“}data\text{”})$
        $alloc(p_{new}, R_{AB}/2)$
        $ack = PRF_{S_{AB}}(P|\text{“}ack\text{”})$
        $ch\_send(ack, PRF^*_{S_{AB}}(P|\text{“}ack\text{”}))$
        $P = P + 1$
```

Fig. 4. Two-party ack-based port hopping.

opens the Pth port in the second pseudorandom sequence designated for acks. When B receives a valid data message from A on port p_{old}, it sends an ack to the old ack port. When it receives a valid message on port p_{new}, it sends an ack to the Pth ack port and then increases P. When A receives a valid ack on port p_{ack}, it increases P. We note that several data messages may be in transit before a port hop takes place, since it takes at least one round-trip time for a port hop to take effect, and in a high-speed network, multiple messages are sent within this time span. The proof of the next theorem is given in Appendix B.

Theorem 1. *When using the ack-based protocol, the probability that a data message that A sends to port p arrives when p is open is 1 up to a polynomially negligible factor.*[5]

In order to compute the throughput that the protocol can support in the absence of a DoS attack (that is, when $C = 0$), we need to take latency variations into consideration. Since messages sent up to Δ time apart can arrive in the same delivery slot, a throughput $T \leq R/\Delta$ ensures $a \leq R$. Since the protocol uses two incoming ports with the same rations, we require $T \leq \frac{R}{2\Delta}$, that is, $a \leq \frac{R}{2}$.

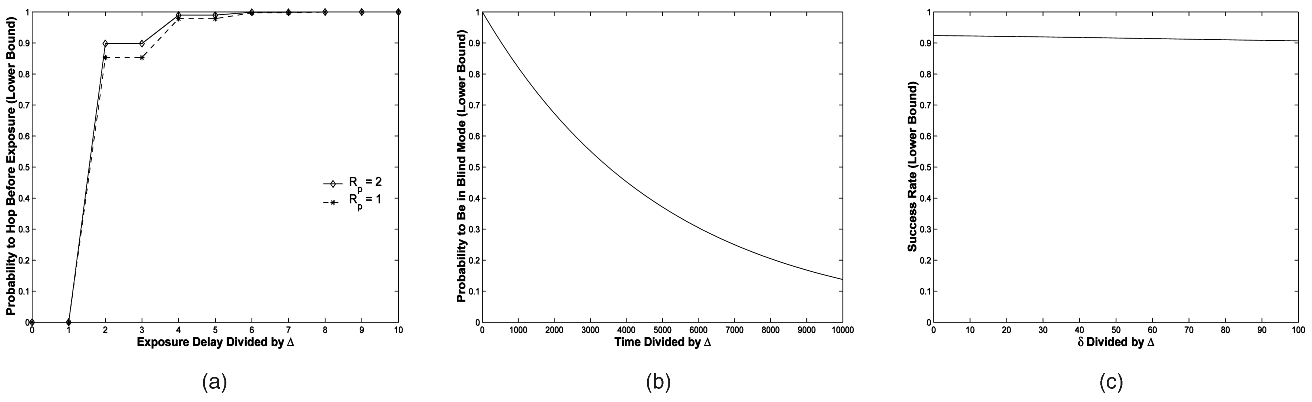
We now analyze the protocol's success rate under DoS attacks. We say that the adversary is in *blind mode* if it cannot distinguish the ports used by the protocol from random ports. We first give a lower bound on the success rate in blind mode and then give a lower bound on the probability to be in blind mode at a given time t. Finally, μ is bounded by the probability to be in blind mode throughout the execution of the protocol times the success rate in blind mode.

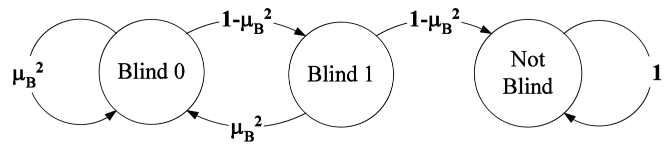
5. Namely, for every polynomial $g > 0$, there is some κ_g such that when $\kappa \geq \kappa_g$, then the probability $\geq 1 - g(\kappa)$.

Suppose that B opens port p with reception rate R_p and that $a \leq R_p$ messages from A are waiting in its channel, along with c_p messages from the adversary ($c_p \geq 0$). By Lemma 1, the success rate does not monotonically increase with a. Since the adversary can control a by varying the network delays, it can set a as high as possible for a delivery slot. Therefore, the worst case occurs when $a = T\Delta$. Using (1), we get that the success rate in blind mode is bounded from below by $f(T\Delta, C, R)$.

Note that the protocol begins in blind mode. We now analyze the probability that the protocol keeps the adversary in blind mode. The only way the adversary can learn of a port used by the protocol is using an *expose* event $\mathcal{E}$ time after a message is sent to that port. This information is only useful for an attack if the port is still in use. Let us trace the periodic sequence of events that causes the data port to change (once it changes, acks for the old port are useless). Assume that A continuously sends messages $m_1, m_2, \ldots$ to B starting at time 0 and consider an execution without an attack: 1) by time Δ, B receives a valid message from the channel and sends an ack to A, 2) by time 2Δ, A receives the ack and changes the sending port, and 3) B gets the last message destined for the old port at most at time 3Δ.

If $\mathcal{E} \geq 3\Delta$, the adversary remains in blind mode. Now, let us examine what happens under attack. In order to prevent the port from changing, the adversary must either prevent B from getting valid data messages or prevent A from receiving acks. By Lemma 2, the probability that all valid messages are dropped decreases when a increases. Thus, (as opposed to the previous analysis), in order to increase the probability that all valid messages are dropped, the





(d)

Fig. 5. The effect of $\mathcal{E}$ on the ack-based protocol, $\psi = 65{,}536$. (a) Bound on the probability to hop before exposure, $\frac{C}{R_p} = 10{,}000$. (b) Bound (g) on the probability to stay in blind mode, $R_p = 1$, $C = 10{,}000$, and $\mathcal{E} = 4\Delta$. (c) Bound on the protocol's success rate, $\Delta = T = R_p = 1$, $C = 10{,}000$, and $\mathcal{E} = 5\Delta$. (d) Markov chain for computing the lower bound in (b).

adversary would like to decrease a to its minimum. Obviously, the attacker would like to get out of blind mode, and for that, it needs A to send at least one message to B to expose the port being used, and so, $a = 1$. We get that the lower bound on the probability of a single message to be received on a single port, as given in Section 4.1, is $\mu_B = f(1, C, \frac{R}{2})$.

Lemma 6. *If $\mathcal{E} = 2k\Delta$ for $k > 0$ and A sends messages to B at least every 2Δ time units, then the probability that the port changes while the attacker is still blind is at least $1 - (1 - \mu_B^2)^k$.*

Proof. The probability that the port does not change in a single round-trip is at most $1 - \mu_B^2$. Since A sends messages to B every 2Δ time units, at the conclusion of each maximal time round-trip, there is at least one new message on its own round-trip. In order for the port not to change while the adversary is still blind, every round-trip needs to fail. Since the attacker can react only after $2k\Delta$ time, there is time for k round-trips in which the attacker is blind, even if none of them succeed. The probability that all of them fail is less than $(1 - \mu_B^2)^k$. If one succeeds, the port changes. Therefore, the probability that the port changes is at least $1 - (1 - \mu_B^2)^k$. □

The lower bound above is illustrated in Fig. 5a.

We now bound the probability to be in blind mode at time t by assuming that once the attacker leaves the blind mode it never returns to it. The bound is computed using a Markov chain, where each state is the number of round-trips that have failed since the last port change. In the last state, all round-trips have failed before the exposure, and thus, the attacker is no longer blind. The Markov chain for $\mathcal{E} = 4\Delta$ is shown in Fig. 5d. We use the chain's transition matrix to compute the probability $g(t, \mathcal{E}, C, R)$ for remaining in blind mode at time t. Fig. 5b shows values of g for $\mathcal{E} = 4\Delta$. We can see that the protocol works well only for a limited time.

Finally, we note that the protocol's message complexity is 2, since it sends an ack for each message, and its bit complexity is constant: $\log_2(\psi)$ bits for the port plus κ bits for the authentication code.

5.2 Adding Proactive Reinitializations

We now introduce a proactive reinitialization mechanism that allows choosing new seeds for the ack-based protocol depending on time and not on the messages passed in the system. We denote by $t_A(t)$ and $t_B(t)$ the local clocks of A and B, respectively, where t is the real time. From Section 3.1, we get that $0 \leq |t_A(t) - t| \leq \Phi$, $0 \leq |t_B(t) - t| \leq \Phi$. We also assume that $t_A, t_B \geq 0$.

If A reinitializes the ack-based protocol and then sends a message to B at time $t_A(t_0)$, this message can reach B anywhere in the real-time interval $(t_0, t_0 + \Delta]$. Therefore, the port used by A at $t_A(t_0)$ must be opened by B at least throughout this interval. To handle the extreme case where A sends a message at the moment of reinitialization, B must use the appropriate port starting at time $t_B(t_0) - \Phi$. (We note that t_0 may also be Φ time units apart from $t_A(t_0)$.) We define δ as the number of time units between reinitializations of the protocol and assume for simplicity and effectiveness of resource consumption that $\delta > 4\Phi + \Delta$ (see Fig. 6 for more details).

Every δ time units, A feeds a new seed to the ack-based protocol, and B anticipates it by creating a new instance of the protocol, which waits on the new expected ports. Once communication is established using the new protocol instance or once it is clear that the old instance is not going to be used anymore, the old instance is terminated. The

```
PROTOCOL ADD-ON FOR SENDER $A$:

Whenever $t_A(t) \in \{0, \delta, 2\delta, \ldots\}$:
    $ack\_init(t_A(t)/\delta)$

PROTOCOL ADD-ON FOR RECIPIENT $B$:

When $t_B(t) = 0$:
    Create the first ack-based protocol instance
    For that instance, $ack\_init(0)$

PROTOCOL ADD-ON FOR RECIPIENT $B$ (CONTINUED):

Whenever $(t_B(t) + 2\Phi) \in \{\delta, 2\delta, 3\delta, \ldots\}$:
    Create a new ack-based protocol instance
    For that instance, $ack\_init((t_B(t) + 2\Phi)/\delta)$

$4\Phi + \Delta$ time after creating a new ack-based protocol instance
or $\Delta$ time after receiving the first msg for this new instance:
    Terminate all older protocol instances
```

Fig. 6. Proactive reinitialization of the ack-based protocol.

pseudocode for the proactive reinitialization mechanism can be found in Fig. 6. Due to space considerations, we do not detail the change in port rations at the recipient's side as protocol instances are created or terminated. We also note that there is a negligible probability that more than one ack-based protocol instance will share the same port. Even if this happens, differentiating between instances can be easily done by adding the instance number (that is, the total number of times a reinitialization was performed) to each message. The proof of the next theorem is given in Appendix C.

Theorem 2. *When using the ack-based protocol with proactive reinitializations, the probability that a data message that A sends to port p arrives when p is open is 1 up to a polynomially negligible factor.*

Proactive reinitialization every δ time units allows us to limit the expected degradation in the success rate for a single ack-based protocol instance. Choosing δ is therefore an important part of the combined protocol. A small δ allows us to maintain a high success rate in the ack-based protocol but increases the average number of ports that are open in every time unit (due to running several protocol instances in parallel). When several ports are used, the ration for each one of them is decreased and so might the success rate. On the other hand, choosing a high δ entails a lower success rate between reinitializations. We conclude the discussion above and the results presented in Section 5.1 with the following theorem:

Theorem 3. *Assume that if A sends a message to B in a single reinitialization period, then A keeps sending messages to B at least every 2Δ time units or until that period ends. Then, the success rate of the proactively reinitialized ack-based protocol with reinitialization periods of length δ is bounded from below by $g(\delta + \Delta, \mathcal{E}, C, R) \cdot f(T\Delta, C, R)$ up to a polynomially negligible factor.*

Fig. 5c shows the value of

$$g(\delta + 1, \mathcal{E}, 10{,}000, 1) \cdot f(1, 10{,}000, 1, 1).$$

We can see that the proactively reinitialized protocol's success rate stays over 90 percent even for $\delta = 100\Delta$, that is, even for relatively long periods between reinitializations.

5.3 Feasibility Discussion

A router/firewall that has IPsec support can be easily modified to support our hopping protocols. Such a router/firewall already has properties we can use: It is able to filter packets according to their SPI field, it has integrated authentication and hash functions (that can be used as PRFs), and it supports secret shared keys. The only thing that is left to do is to perform SPI hopping. Thus, combining our hopping protocols with IPsec allows for ease of implementation while providing IPsec's strong authentication capabilities for higher level protocols, along with our robustness to DoS attacks, since hopping ensures that only packets that are valid with high probability go through the expensive authentication stage. We therefore believe that an integration of our hopping protocols with IPsec is an attractive choice.

The two-party communication protocols we presented use a shared secret, known only to the two parties. Each pair of communicating parties shares a different secret. An integration of our protocols with IPsec in tunnel mode on a gateway means that the gateway might have to deal with several parties. The number of secrets that are stored on the gateway is thus linear in the number of parties. However, using a hash table, every SPI lookup takes $O(1)$ and, so, filtering is done at $O(1)$ per packet. All packets that do not contain the correct SPI are dropped at this filtering stage.

6 Conclusions and Directions for Future Work

We have presented a model for port-based rationing channels and a protocol robust to DoS attacks for communication over such channels. Our protocol is simple and efficient and hence can sustain high loads of traffic, as happens, for example, in high-speed networks. At the same time, our analysis shows that the protocol is highly effective in mitigating the effects of DoS attacks. Our formal framework and suggested protocol apply not only to port-based filtering but also to a much broader category of filtering based on any packet identifier. Thus, our work constitutes the first step in evaluating existing filtering and rate-limiting mechanisms.

As the important field of application-level DoS mitigation is relatively new, there is much research space to explore. Although our worst-case analysis is valuable, it can be followed by simulations, experiments, and common case analysis. Moreover, the system aspects of deploying such a protocol in today's Internet are yet to be dealt with. We now describe several exemplary future research directions.

Our model is realistic, as it only requires the underlying channel to provide port-based filtering; therefore, it can be efficiently implemented using existing mechanisms, typically at a gateway firewall or router. This raises an interesting question regarding the trade-off between the cost and the possible added value of implementing additional functionality by the channel (for example, at the firewall). We hope that future work will take further strides toward defining realistic yet tractable models of the channel and the adversary that will aid in answering this question.

This work has focused on two parties only. It would be interesting to extend it to multiparty scenarios such as client-sever and multicast. These scenarios may require a somewhat different approach and will obviously necessitate analyses of their own. Furthermore, we required the parties to share a secret key; we believe we can extend the solution to establish this key using additional parties, for example, a key distribution center, or using "proof of work" [7].

Our work has focused on resisting DoS attacks; however, it could impact the performance and reliability properties of the connection; in fact, it is interesting to explore combinations between our model and problem and the classical problems of reliable communication over unreliable channels and networks. Furthermore, since our work requires a shared secret key, it may be desirable to merge it with protocols using shared secret keys for confidentiality and authentication, such as Secure Sockets Layer/Transport Layer Security (SSL/TLS) and IPsec.

Appendix A

Channel Delivery Probability Analysis—Proofs of Lemmas

We now prove the lemmas from Section 4. Since a, C, and R are constants, denote $\mu_B(s_k) = \mu_B(a, C, R, s_k)$.

Lemma 3. *Fix k, a, C, R, and $s_k \in S_k$ and denote the ports attacked under s_k by $p_1, p_2, \ldots, p_k$ with attacking rates of $c_{p_1}, c_{p_2}, \ldots, c_{p_k}$, respectively. Then, under a blind attack with strategy s_k, the worst (that is, minimal) expected delivery probability of the system is achieved when $\forall i\ c_{p_i} = \frac{C}{k}$.*

Proof. By Proposition 1, $\mu_B(s_k) = \frac{\psi - k}{\psi} + \frac{1}{\psi}\sum_{i=1}^{k} \frac{R}{c_{p_i}+a}$. Calculating the partial derivatives of $\mu_B(s_k)$, we get that $\frac{\partial \mu_B(s_k)}{\partial c_{p_i}} = \frac{1}{\psi} \cdot \frac{-R}{(c_{p_i}+a)^2}$, that is, $\mu_B(s_k)$ is monotonically decreasing as we increase c_{p_i} and keep c_{p_j} the same for $j \neq i$. Thus, the attacker wants to increase c_{p_i} to decrease the delivery probability of the communication channel. However, we have the constraint $\sum_{i=1}^{k} c_{p_i} = C$. Integrating this constraint into our delivery probability function using a Lagrange coefficient denoted by β gives

$$\mu_{B'}(s_k) = \frac{\psi - k}{\psi} + \frac{1}{\psi}\sum_{i=1}^{k} \frac{R}{c_{p_i}+a} + \beta\left(\sum_{i=1}^{k} c_{p_i} - C\right).$$

We now look for an extremum point by comparing the partial derivatives of $\mu_{B'}(s_k)$ to zero:

$$\begin{aligned} \frac{\partial \mu_{B'}(s_k)}{\partial c_{p_i}} &= 0, \\ \frac{1}{\psi} \cdot \frac{-R}{(c_{p_i}+a)^2} + \beta &= 0, \\ c_{p_i} &= \sqrt{\frac{R}{\psi\beta}} - a. \end{aligned}$$

Putting the values of c_{p_i} into the constraint equation $C = \sum_{i=1}^{k} c_{p_i}$ gives

$$\begin{aligned} C &= \sum_{i=1}^{k}\left(\sqrt{\frac{R}{\psi\beta}} - a\right) \\ \beta &= \frac{R}{\psi\left(\frac{C}{k}+a\right)^2}. \end{aligned}$$

Going back to the equation for c_{p_i}, we get

$$c_{p_i} = \sqrt{\frac{R}{\psi \cdot \frac{R}{\psi\left(\frac{C}{k}+a\right)^2}}} - a = \sqrt{\left(\frac{C}{k}+a\right)^2} - a = \frac{C}{k}.$$

This result also fits our constraint $c_{p_i} > 0$, and we have an extremum point for $\mu_B(s_k)$ at $c_{p_i} = \frac{C}{k}$. (We note that $\frac{C}{k}$ might not be an integer, but since we want a lower bound, this does not make a difference.) We denote this extremum point by s_k^*. Now, we need to show that s_k^* is a minimum point. If we show that $\mu_B(s_k)$ is convex, then from the Kuhn-Tucker Theorem, we get that s_k^* is a global minimum point. We proceed by showing that $\mu_B(s_k)$ is convex.

We have already shown that $\frac{\partial \mu_B(s_k)}{\partial c_{p_i}} = \frac{R}{\psi} \cdot \frac{-1}{(c_{p_i}+a)^2}$. We get that $\mu_B(s_k)$ is twice continuously differentiable, and the second derivative is

$$\frac{\partial^2 \mu_B(s_k)}{\partial c_{p_i} \partial c_{p_j}} = \begin{cases} 0 & i \neq j, \\ \frac{R}{\psi} \cdot \frac{2(c_{p_i}+a)}{(c_{p_i}+a)^4} & i = j. \end{cases}$$

We get that the Hessian of $\mu_B(s_k)$ is a positive diagonal matrix. Thus, $\mu_B(s_k)$ is convex, and from the Kuhn-Tucker Theorem, $\mu_B(s_k^*)$ is a global minimum of the delivery probability function $\mu_B(s_k)$. □

Lemma 4. *Let $R = a$. Then, an adversary with $C \geq \psi$ cannot decrease the expected delivery probability lower than $\frac{\psi a}{C + \psi a}$, and an adversary with $C \leq \psi$ cannot decrease the expected delivery probability lower than $1 - \frac{C}{\psi(1+a)}$.*

Proof. Let $R = a$. We get that

$$\mu'_B(k) = \frac{R-1}{\psi} - \frac{R}{\psi} \cdot \frac{C^2 + (2kC + k^2R)(R-1)}{(C+kR)^2}.$$

We now show that $\mu'_B(k) < 0$:

$$\begin{aligned} \frac{R-1}{\psi} - \frac{R}{\psi} \cdot \frac{C^2 + (2kC + k^2R)(R-1)}{(C+kR)^2} &\overset{?}{<} 0, \\ 0 &\overset{?}{<} C^2. \end{aligned}$$

Clearly, the last inequality holds, and we get that $\mu_B(k)$ monotonically decreases as k increases. Thus, the adversary wants to choose k as large as possible. Ideally, $k = \psi$, $C \geq \psi(R - a + 1) = \psi$, and we get

$$\mu_B(a, R, C) \geq \frac{a}{\psi} \cdot \frac{\psi^2}{C + \psi a} = \frac{\psi a}{C + \psi a}.$$

However, this attack requires substantial strength from the adversary, that is, the adversary needs to be more than ψ times stronger than B. If $C \leq \psi(R - a + 1) = \psi$, we get that $k = \frac{C}{R-a+1} = C$. The resulting degraded delivery probability is

$$\begin{aligned}\mu_B(a, R, C) &\geq \frac{\psi - C}{\psi} + \frac{a}{\psi} \cdot \frac{C^2}{C(1 + a)} \\ &= \frac{\psi(1 + a) - C(1 + a) + aC}{\psi(1 + a)} \\ &= 1 - \frac{C}{\psi(1 + a)} \geq 1 - \frac{\psi}{\psi(1 + a)} = 1 - \frac{1}{1 + a}.\end{aligned}$$

□

Lemma 5. *Let $a < R$. Then, an adversary with $C \geq \frac{\psi a}{\sqrt{\frac{R}{R-a}}} - 1$ cannot decrease the expected delivery probability lower than $\frac{\psi R}{C+\psi a}$, and an adversary with $C \leq \frac{\psi a}{\sqrt{\frac{R}{R-a}}} - 1$ cannot decrease the expected delivery probability lower than*

$$\frac{\psi a - C\left(\sqrt{\frac{R}{R-a}} - 1\right)}{\psi a} + \frac{R}{\psi} \cdot \frac{C\left(\sqrt{\frac{R}{R-a}} - 1\right)^2}{a^2\sqrt{\frac{R}{R-a}}}.$$

Proof. Since $a < R$, we get $R \geq 2$. Let us find the value of k that minimizes the delivery probability:

$$\begin{aligned}\mu'_B(k) &= 0, \\ \frac{R - 1}{\psi} - \frac{R}{\psi} \cdot \frac{C^2 + (2kC + k^2a)(a - 1)}{(C + ka)^2} &= 0, \\ ak^2 + 2Ck - \frac{C^2}{R - a} &= 0.\end{aligned}$$

Since $k > 0$, we get that the solution is

$$k = \frac{-2C + \sqrt{4C^2 + \frac{4C^2a}{R-a}}}{2a} = \frac{-2C + \sqrt{\frac{4C^2R}{R-a}}}{2a} = \frac{C\left(\sqrt{\frac{R}{R-a}} - 1\right)}{a}.$$

Obviously, this value of k is not an integer. However, we use it to bound the minimum delivery probability under a blind DoS attack. First, we need to show that this value of k is indeed a minimum point. We do this by showing that the second derivative of $\mu_B(k)$ is always positive:

$$\mu''_B(k) = \frac{R}{\psi} \cdot \frac{2x(C + kx)[C^2 + (2kC + k^2a)(a - 1)] - (2C + 2ka)(a - 1)}{(C + k)^4}.$$

It suffices to show that the numerator is always positive. That is, we need to show that

$$a(2C + 2ka)\left[C^2 + (2kC + k^2a)(a - 1)\right] > (2C + 2ka)(a - 1).$$

This is clearly true, since $a \geq 1$, $k \geq 1$, and $C \geq 1$, and we get $a[C^2 + (2kC + k^2a)(a - 1)] > a - 1$. Thus, $\mu''_B(k)$ is always positive, and we have found a minimum point.

We also need the found k to be in range. Clearly, $k > 0$. We still need to show that $k \leq \frac{C}{R-a+1}$:

$$\begin{aligned}k &\overset{?}{\leq} \frac{C}{R - a + 1}, \\ \frac{C\left(\sqrt{\frac{R}{R-a}} - 1\right)}{a} &\overset{?}{\leq} \frac{C}{R - a + 1}, \\ a &\overset{?}{\leq} R - \frac{1}{R}.\end{aligned}$$

The last inequality holds since $a < R$, a is an integer, and $R \geq 2$. Thus, $k \leq \frac{C}{R-a+1}$.

We can now bound the expected delivery probability $\mu(a, R, C)$ from below. For the case where

$$k = \frac{C\left(\sqrt{\frac{R}{R-a}} - 1\right)}{a} \leq \psi,$$

we get

$$\begin{aligned}\mu_B(a, R, C) &\geq \frac{\psi - \frac{C\left(\sqrt{\frac{R}{R-a}}-1\right)}{a}}{\psi} + \frac{R}{\psi} \cdot \frac{\frac{C^2\left(\sqrt{\frac{R}{R-a}}-1\right)^2}{a^2}}{C + \frac{C\left(\sqrt{\frac{R}{R-a}}-1\right)}{a}a} \\ &= \frac{\psi a - C\left(\sqrt{\frac{R}{R-a}} - 1\right)}{\psi a} + \frac{R}{\psi} \cdot \frac{C\left(\sqrt{\frac{R}{R-a}} - 1\right)^2}{a^2\sqrt{\frac{R}{R-a}}}.\end{aligned}$$

For the case where $\frac{C\left(\sqrt{\frac{R}{R-a}}-1\right)}{a} > \psi$, since $\mu_B(k)$ has just one extremum point and it is a minimum point with $k > \psi$, we get that the attacker's best strategy is to choose $k = \psi$, and we get

$$\mu_B(a, R, C) \geq \frac{\psi - \psi}{\psi} + \frac{\psi^2 R}{\psi(C + \psi a)} = \frac{\psi R}{C + \psi a}.$$

Note that we got the same result for $R = a$ and $k = \psi$. However, the conditions for choosing $k = \psi$ are different. For $R = a$, we choose $k = w$ if $C \geq w$. For $R > a$, we choose $k = \psi$ if $\frac{C\left(\sqrt{\frac{R}{R-a}}-1\right)}{a} > \psi$. □

Appendix B

Ack-Based Protocol—Proof of Correctness

Invariant 1. *Let P_A and P_B be the P counters that A and B hold in the ack-based protocol, respectively. The probability that $P_B - P_A \in \{0, 1\}$ is 1 up to a polynomially negligible factor.*

Proof. After the initialization stage, $P_A = P_B$, and the property $P_B - P_A \in \{0, 1\}$ holds.

When the counters are equal, the part of the protocol that may update them proceeds as follows:

1. A sends a message to B on port

$$PRF_{S_{AB}}(P_A|''data'').$$

2. If the message reaches B in a valid state, B adds 1 to P_B and sends an ack back to A on port $PRF_{S_{BA}}(P_B|\,''ack'')$.
3. If the ack reaches A in a valid state, A adds 1 to P_A.

If steps 2 and 3 complete successfully, both counters advance by 1 and remain equal to each other. If step 2 fails (message dropped or modified in transit), both counters remain unchanged. If step 2 succeeds but step 3 fails (ack lost or changed in transit), P_B is incremented by step 1, but P_A remains the same. Thus, if $P_A = P_B$, the next change of counters will still maintain the property $P_B - P_A \in \{0, 1\}$.

Now, suppose that we have reached the state where $P_B = P_A + 1$. The portion of the protocol that may update the counters proceeds as follows:

1. A sends a message to B on port
$$PRF_{S_{AB}}(P_A|\,''data'').$$
2. If the message reaches B in a valid state, B sends an ack back to A on port $PRF_{S_{BA}}(P_B - 1|''ack'')$.
3. If the ack reaches A in a valid state, A adds 1 to P_A.

If steps 2 and 3 complete successfully, P_A advances by 1 and the counters become equal to each other. If steps 2 or 3 fail (messages dropped or are not valid), both counters remain unchanged. Thus, if $P_B = P_A + 1$, the next change of counters will still maintain the property $P_B - P_A \in \{0, 1\}$.

The only way to break this invariant is if the attacker makes just one party advance its counter. This means that the adversary has to fabricate a message so that one party will think it is valid. Thus, the attacker needs to guess both the port number and the authentication information attached to each message. The probability that the attacker succeeds in doing so is a polynomially negligible factor. □

Theorem 1. *When using the ack-based protocol, the probability that a data message that A sends to port p arrives when p is open is 1 up to a polynomially negligible factor.*

Proof. According to Invariant 1, when A sends a data message to B, either $P_A = P_B$ or $P_B = P_A + 1$, with probability 1 up to a polynomially negligible factor.

For the first case, let M be a message A sends to B when $P_A = P_B$. Since B always opens two ports for data, we need to show that P_B does not increase by more than one until M actually reaches B. Since the link maintains the FIFO semantics, messages sent after M was sent cannot change the value of P_B before M reaches B. The only messages that can change P_B are messages that preceded M but reached B only after M was sent.

According to the protocol, P_B increases by one if and only if B receives a data message from A that was sent using the counter $P_A = P_B$. Furthermore, all messages preceding M were sent using a counter that is less than or equal to P_A. It follows that P_B can only increase by one from the time M leaves A until it reaches B.

Consider now the second case where M was sent when $P_B = P_A + 1$. Since B only opens two ports for data, we need to show that P_B does not change at all. Again, since the link has FIFO semantics, P_B can only change by messages preceding M that reach B after M was sent but before it reaches B. However, such messages have counters that are less than or equal to P_A and, thus, strictly less than P_B. According to the protocol, messages sent with such counters do not affect the value of P_B. □

Appendix C

Ack-Based Protocol with Reinitializations—Proof of Correctness

Theorem 2. *When using the ack-based protocol with proactive reinitializations, the probability that a data message that A sends to port p arrives when p is open is 1 up to a polynomially negligible factor.*

Proof. From Theorem 1, we get that if A and B both use the ack-based protocol initialized with $seed$, then messages sent by A arrive to open ports at B. To complete the proof, we need to show the following:

1. When A reinitializes the protocol with a new seed, B has already started running an ack-based protocol instance using the same seed.
2. B does not terminate a protocol instance while it may still receive messages corresponding to that instance.

For the first property, let us look at some real time t_n^A when A reinitializes the protocol, where $t_A(t_n^A) = n\delta$, $n \in \mathbb{N}$. From the bounded drift assumption, we get the bound $t_n^A \geq n\delta - \Phi$. The seed corresponding to the initialization at t_n^A is $\frac{t_A(t_n^A)}{\delta} = n$. Now, let us look at the real time t_n^B in which B starts a new ack-based protocol instance with the seed n. This happens when $t_B(t_n^B) + 2\Phi = n\delta$, that is, when $t_B(t_n^B) = n\delta - 2\Phi$. Using the bounded drift assumption, we get the bound $t_n^B \leq n\delta - 2\Phi + \Phi = n\delta - \Phi \leq t_n^A$.

For the second property, let us look at seed n again. A terminates the instance with seed n at real time t_{n+1}^A. The last packet sent using the ack-based protocol initialized with seed n inevitably reaches B before real time $t_{n+1}^A + \Delta$. B terminates the ack-based protocol instance in either one of the following two cases:

1. at time $t_B(t_{n+1}^B) + 4\Phi + \Delta$ and
2. Δ time units after receiving the first message for a newer ack-based protocol instance.

For the first case, we get $t_{n+1}^B \geq (n+1)\delta - 2\Phi - \Phi + 4\Phi + \Delta = (n+1)\delta + \Phi + \Delta \geq t_{n+1}^A + \Delta$. For the second case, we observe that if a message for a newer instance of the ack-based protocol has arrived, then A is no longer sending messages with instances initialized with older seeds. However, the varying message propagation delay means that messages from older protocol instances can take up to Δ time units to arrive, whereas the new message might have taken negligible time to arrive. □

ACKNOWLEDGMENTS

Gal Badishi is supported by the Israeli Ministry of Science. A preliminary version of this paper appeared in the *Proceedings of the International Symposium on Distributed Computing (DISC '05)*.

REFERENCES

[1] D.G. Andersen, "Mayday: Distributed Filtering for Internet Services," *Proc. Fourth Usenix Symp. Internet Technologies and Systems (USITS '03)*, 2003.

[2] K. Argyraki and D.R. Cheriton, "Active Internet Traffic Filtering: Real-Time Response to Denial-of-Service Attacks," *Proc. Usenix Ann. Technical Conf.*, Apr. 2005.

[3] R. Atkinson, *Security Architecture for the Internet Protocol*, IETF RFC 2401, 1998.

[4] G. Badishi, I. Keidar, and A. Sasson, "Exposing and Eliminating Vulnerabilities to Denial of Service Attacks in Secure Gossip-Based Multicast," *Proc. 37th Ann. IEEE/IFIP Int'l Conf. Dependable Systems and Networks (DSN '04)*, pp. 223-232, June-July 2004.

[5] M. Collins and M.K. Reiter, "An Empirical Analysis of Target-Resident DoS Filters," *Proc. IEEE Symp. Security and Privacy*, pp. 103-114, May 2004.

[6] Computer Crime and Security Survey, Computer Security Inst./ Federal Bureau of Investigation (CSI/FBI), 2003.

[7] V.D. Gligor, "Guaranteeing Access in Spite of Service-Flooding Attacks," *Proc. 11th Int'l Workshop Security Protocols*, 2003.

[8] O. Goldreich, S. Goldwasser, and S. Micali, "How to Construct Random Functions," *J. Assoc. for Computing Machinery*, vol. 33, no. 4, pp. 792-807, 1986.

[9] C. Jin, H. Wang, and K.G. Shin, "Hop-Count Filtering: An Effective Defense against Spoofed DDoS Traffic," *Proc. 10th ACM Conf. Computer and Comm. Security (CCS '03)*, V. Atluri and P. Liu, eds., pp. 30-41, Oct. 2003.

[10] J. Jung, B. Krishnamurthy, and M. Rabinovich, "Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites," *Proc. 11th Int'l World Wide Web Conf. (WWW '02)*, pp. 252-262, May 2002.

[11] "The Need for Pervasive Application-Level Attack Protection," white paper, Juniper Networks, 2004.

[12] A.D. Keromytis, V. Misra, and D. Rubenstein, "SOS: An Architecture for Mitigating DDoS Attacks," *J. Selected Areas in Comm.*, vol. 21, no. 1, pp. 176-188, 2004.

[13] B. Krishnamurthy and J. Wang, "On Network-Aware Clustering of Web Clients," *Proc. ACM Conf. Applications, Technologies, Architectures, and Protocols for Computer Comm. (SIGCOMM '00)*, Aug. 2000.

[14] H.C.J. Lee and V.L.L. Thing, "Port Hopping for Resilient Networks," *Proc. 60th IEEE Vehicular Technology Conf.*, Sept. 2004.

[15] P. Mahajan, S.M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling High Bandwidth Aggregates in the Network," *Computer Comm. Rev.*, vol. 32, no. 3, pp. 62-73, July 2002.

[16] D. Moore, G. Voelker, and S. Savage, "Inferring Internet Denial-of-Service Activity," *Proc. 10th Usenix Security Symp.*, pp. 9-22, Aug. 2001.

[17] W.G. Morein, A. Stavrou, D.L. Cook, A.D. Keromytis, V. Misra, and D. Rubenstein, "Using Graphic Turing Tests to Counter Automated DDoS Attacks against Web Servers," *Proc. 10th ACM Conf. Computer and Comm. Security (CCS '03)*, pp. 8-19, 2003.

[18] "Web Application Firewall: How NetContinuum Stops the 21 Classes of Web Application Threats," white paper, NetContinuum, 2004.

[19] "DoS Protection," white paper, P-Cube, 2004.

[20] "Minimizing the Effects of DoS Attacks," white paper, P-Cube, 2004.

[21] "Defeating DDoS Attacks," white paper, Riverhead Networks, 2004.

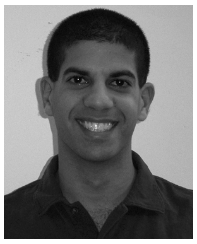
[22] S.M. Schwartz, "Frequency Hopping Spread Spectrum (FHSS) vs. Direct Sequence Spread Spectrum (DSSS) in the Broadband Wireless Access and WLAN Arenas," white paper, 2001.

[23] A. Stavrou and A.D. Keromytis, "Countering DoS Attacks with Stateless Multipath Overlays," *Proc. 12th ACM Conf. Computer and Comm. Security (CCS '05)*, Nov. 2005.

[24] J. Wang, X. Liu, and A.A. Chien, "Empirical Study of Tolerating Denial-of-Service Attacks with a Proxy Network," *Proc. 14th Usenix Security Symp.*, 2005.

[25] A. Yaar, A. Perrig, and D. Song, "Pi: A Path Identification Mechanism to Defend against DDoS Attacks," *Proc. IEEE Symp. Security and Privacy*, May 2003.

[26] A. Yaar, A. Perrig, and D. Song, "SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks," *Proc. IEEE Symp. Security and Privacy*, May 2004.



Gal Badishi received the BSc degree in computer science from the Hebrew University of Jerusalem in 2000 and the MSc degree from Technion, Israel Institute of Technology, in 2004. He is currently a PhD candidate in the Electrical Engineering Department, Technion, Israel Institute of Technology, on a direct PhD track. He has worked as a software engineer at Educational Robotics Ltd. and has been invited twice for an internship at Sun Microsystems Research Laboratories. He holds the Eshkol fellowship from the Israeli Ministry of Science and Technology.

Amir Herzberg received the BSc degree in computer engineering, the MSc degree electrical engineering, and the DSc degree in computer science from Technion, Israel Institute of Technology, Israel, in 1982, 1987, and 1991, respectively. Since 1982, he has worked in software and systems R&D, mostly in security and networking, as a developer, manager, or Chief Technical Officer (CTO), in a few companies. During 1991-2000, he filled research and management positions at IBM Research (New York and Israel). Since 2002, he has been an associate professor in the Computer Science Department, Bar Ilan University. His current research interests include applied cryptography, secure communication, and secure e-commerce.

Idit Keidar received the BSc, MSc, and PhD (*summa cum laude*) degrees from the Hebrew University of Jerusalem. She is a faculty member in the Department of Electrical Engineering, Technion, Israel Institute of Technology. She was a postdoctoral research associate in the Laboratory for Computer Science, Massachusetts Institute of Technology (MIT), where she held postdoctoral fellowships from Rothschild Yad-Hanadiv and US National Science Foundation (NSF) Computer and Information Science and Engineering (CISE). She has consulted for BBN Technologies (a Verizon Company) in the area of fault tolerance and intrusion tolerance and for Microsoft Research in the area of fault-tolerant storage systems. She is the academic head of the Software Systems Laboratory, Technion. His research focuses on distributed algorithms and system, as well as fault-tolerant network-based computing. He served as a member of the steering committee of the ACM Symposium on Principles of Distributed Computing (PODC), has served on numerous program committees of leading conferences in the area of distributed and parallel computing, has twice served as a vice chair for the IEEE International Conference Distributed Computing Systems (ICDCS), and once served as a vice chair for Euro-Par. He is a recipient of the National Alon Fellowship for new faculty members.