Computer Networks 105 (2016) 180–193

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Tumbler: Adaptable link access in the bots-infested Internet☆

Yao Zhang[a,b,*], Xiaoyou Wang[c], Adrian Perrig[b], Zhiming Zheng[a]

[a] LMIB and School of Mathematics and Systems Science, Beihang University, Beijing, China
[b] Institute of Information Security, Department of Computer Science, ETH Zurich, Switzerland
[c] Information Networking Institute, Carnegie Mellon University, Pittsburgh, USA

ARTICLE INFO

Article history:
Received 14 April 2015
Revised 2 June 2016
Accepted 5 June 2016
Available online 7 June 2016

Keywords:
DDoS attack
Capability scheme
Bandwidth allocation
Competition factor

ABSTRACT

Despite large-scale flooding attacks, capability-based defense schemes provide end hosts with guaranteed communication. However, facing the challenges of enabling scalable bandwidth fair sharing and adapting to attack strategies, none of the existing schemes adequately stand. In this paper we present Tumbler, a flooding attack defense mechanism that provides scalable competition-based bandwidth fairness at the Autonomous System (AS) granularity, and on-demand bandwidth allocation for end hosts in each AS. Tumbler enforces adaptability in the capability establishment via competition factors that are calculated upon leaf ASes' bandwidth utilization and reputation. Transit ASes independently manage each competition factor based on the corresponding feedback from dedicated bandwidth accounting and monitoring policies. Through Internet-scale simulations, we demonstrate the effectiveness of Tumbler against a variety of attack scenarios and illustrate the deployment benefits for ISPs.

1. Introduction

Individuals, industries, and governments are increasingly relying on Internet availability for dependable services. At the same time, Distributed Denial-of-Service (DDoS) attacks remain persistent threats in the current Internet. Emerging DDoS attacks are launched by millions of bots [1], flooding victim links with huge amounts of traffic [2–4]. Recent attacks have even been observed with startling 500 Gbps [5].

In response to these attacks, network capability-based mechanisms [6–14] have emerged as a promising class of DDoS defenses. In a capability-based scheme, a source initiates a capability request to a destination with all the routers on the transiting path adding authentication tokens to the packet header. Upon the request's arrival, the destination can explicitly authorize a desired flow with priority and return the packet to the source. Then generated tokens will act as a capability for the traffic of the authorized flow. By ensuring end-to-end privileged flows, such approaches isolate attack traffic. However, to achieve viable link-flooding defense, capability-based schemes face two critical challenges: *scalability* and *adaptability*.

Scalability refers to the fundamental problem of providing fair access, for the aspects of both capability bootstrapping and bandwidth allocation. Concerning capability bootstrapping, what if capability establishment is interfered by attackers? Given the fact that capability requests are forwarded by best effort, flooding on the requesting channel, namely Denial-of-Capability (DoC) attacks [15], could easily prevent benign sources from obtaining capabilities. Similarly, concerning bandwidth allocation, what if attackers leverage authorized capability packets to flood a link? For any per-source [10] or per-destination [8] fair-sharing mechanism, m end hosts only obtain $1/m$ of the capacity at a given link. The available bandwidth in per-flow schemes is limited to only $1/m^2$ [7]. In a nutshell, with millions of bots competing for the limited capacity of a link, the obtained link access for a legitimate end host becomes infinitesimal, too small to provide useful end-to-end guarantees.

Adaptability refers to another issue on how the ISPs can not only economically maximize their link utilization, but also dynamically provide each customer with deserved link access even under persistent DDoS attacks. The PSP scheme [16] leverages historical traffic to adjust bandwidth allocation for core network flows. But this model would hardly scale to end-to-end guarantees in the Internet as billions of flows have to be considered in the iterative computation. Moreover, previous defenses achieve mostly one-time resilience rather than adjustable protection. A sophisticated attack (e.g., replay flooding) would likely cause perpetual damage to the victim link. Intuitively, an approach to improve adaptability is to combine capability establishment with the observation of traffic.

☆ This project was conducted at ETH Zurich while the first author was a visiting student with the network security group.

* Corresponding author.
E-mail addresses: yaozhang@buaa.edu.cn (Y. Zhang), xiaoyouw@andrew.cmu.edu (X. Wang), adrian.perrig@inf.ethz.ch (A. Perrig), zzheng@pku.edu.cn (Z. Zheng).

http://dx.doi.org/10.1016/j.comnet.2016.06.005
1389-1286/

However, as ISP distance increases, both mutual trust and business incentives diminish. In such cases, cooperation between ISPs becomes an impractical requirement.

To overcome the above limitations, we propose Tumbler, a novel mechanism that provides scalable link access for end hosts and adaptable DDoS-resilience for ISPs in the Internet. Tumbler is designed with the following insights and solutions.

First, resource fair sharing at an Autonomous System (AS) granularity provides a scalable approach of solving the link access problem. Although adversarial botnets can be widely distributed in the Internet, the massive number of bots (on the order of millions) resides in ASes whose scale is always significantly smaller. By September 2015, the total AS number is about 50,000 [17]. Therefore, defending against DDoS attacks by throttling bandwidth among ASes will efficiently limit the damage over the contaminated ASes (i.e., ASes initiate the flooding attacks), regardless of their internal botnet size. Tumbler enforces a competition-based weighted fair sharing among leaf ASes. Namely, each transit AS records a periodically-updated *competition factor* for each leaf AS, based on which an active AS (i.e., ASes have valid allocation on the link) obtains its weighted-shared link access. Link access enables both capability requests and capability-enabled data packets. Subsequently, aggregated access will be further shared by the individual flows from the same AS.

Second, a simple per-AS fair sharing [11,12] is not an optimal strategy. Indeed, the *bandwidth utilization* from different ASes varies significantly due to its connection popularity. Even during different hours of a day, bandwidth utilization from an AS fluctuates [18]. In Tumbler, bandwidth consumption of a leaf AS at the given link is considered as the main factor in Tumbler's competition-based weighted fair sharing, which enables effective management of the link bandwidth. Historical bandwidth allocation statistics are kept at each router via local off-line accounting, and serve as a feedback via competition factor to influence the next round of bandwidth allocation.

Furthermore, despite the lack of inter-AS cooperation, local network analysis at one AS still enables adaptable capability establishment. In Tumbler, an *AS's reputation* is evaluated through regional traffic monitoring, and considered as the other factor in the competition-based weighted fair sharing. Such evaluation is performed independently at every traversing AS, thus eliminating the requirement of mutual trust between ISPs. Specifically, each AS monitors its inbound/outbound traffic. Based on defined traffic policies of bandwidth violation, timely feedback will be returned to adjust each AS's reputation and further regulate the subsequent capability establishments for each AS.

The rest of the paper is organized as follows. We specify the goals and assumptions in Section 2, and present the design of Tumbler in Sections 3 and 4. In Section 5, we analyze the security and overhead aspects of Tumbler. In Section 6, we compare Tumbler with related approaches through Internet-scale simulations, and confirm experimentally the properties of our scheme. More discussions are given in Section 7. Related work is given in Section 8, and we conclude in Section 9.

2. Design goals and assumptions

We first give the basic design goals. Then we specify the assumptions and the threat model that Tumbler aims to combat. For clarity, we denote the end host who sends as "source", and the end host who receives as "destination". We refer to the ASes who contain end hosts as leaf ASes, while other ASes on the routing paths as transit ASes. We call the leaf AS where the source (destination) resides the source (destination) AS.

2.1. Design goals

The following design goals enable a lightweight and deployable capability-based DDoS defense framework in today's Internet.

Scalability. We desire a defense mechanism that achieves scalable link access under the presence of botnets. In addition, an Internet-scale mechanism must be lightweight and incur minimal overhead on the deployed routers.

Adaptability. The mechanism should be able to optimize the utilization of link capacity in terms of economical benefits, and to dynamically adjust its defense strategy based on the attacks evolving over time.

Deployability. The mechanism is expected to be functional even if a few entities adopt. Moreover, the deployment plan should incentivize the early adopters.

2.2. Assumptions

First of all, we assume that when a source sends a capability request, it has the knowledge of network routing, namely a valid inter-domain path to the destination AS. Note that the requirement of path control is not a necessity here. Although selecting a specific routing mechanism remains outside the scope of this paper, multiple routing mechanisms can be leveraged to obtain AS-level paths: A straightforward approach is Border Gateway Protocol (BGP) [19] routing (further discussed in Section 7.1). Additionally, several solutions like NIRA [20], Pathlets [21], and SCION [22] present a fix. In these schemes, source could obtain and specify a valid routing path in the header of packets.

Additionally, we assume that all flows from an AS can be assigned with a unique and unforgeable source-AS identifier. This goal is achievable via some lightweight source authentication protocols [23,24]. In Tumbler, for simplicity, we set a source-AS identifier as the hash of the domain's public key.

2.3. Threat model

We assume that both end hosts and ASes may be malicious: It is possible for any end host to get compromised and support DDoS attacks. The ASes containing such botnets or even other transit ASes may tolerate the malicious traffic thus to assist the cooperative attacks. Yet, cases that ASes intentionally forge/distort packet information, or delay/drop packets are outside our paper scope. We require no restriction on the distribution of botnets, and within a contaminated AS, the botnet could have an arbitrary number of attackers. When evaluating a given end-to-end communication, both source and destination ASes are assumed to be non-contaminated.

In addition to these settings, we consider two classes of DDoS attacks: (1) Request packet flooding (DoC attacks [15]) and (2) capability packet flooding, where botnets may collude and present a coordinated behavior with different strength scenarios (pulsing attack [10]), or location scenarios (rolling attack [11]).

3. Tumbler overview

We present the high-level overview of Tumbler in this section. The Tumbler protocol includes the following two phases:

- Phase 1 (capability establishment): Source and destination set up a communication channel. A communication capability is generated hop-by-hop on the routing path according to the request configuration of the source and the link access admission policies of the transit routers.
- Phase 2 (data transmission and feedback regulation): Source sends data on the channel by adding its latest capability into the packets. Meanwhile, transit ASes perform accounting and

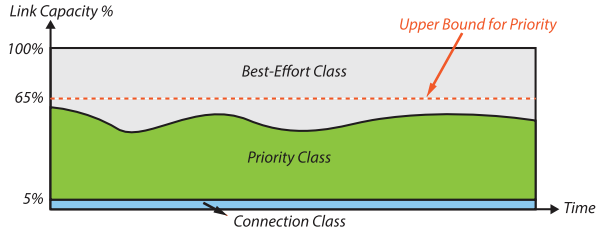

Fig. 1. Bandwidth categories in Tumbler. The connection class accounts for 5% of the link capacity, while the priority bandwidth takes a ratio up to 60%. Both 5% and 60% serve as a baseline for ISPs who can themselves determine the link configuration.

monitoring for data packets. Based on the feedback of the traffic accounting and monitoring, transit ASes regulate the subsequent capability establishments initialized from the source domain.

3.1. Establishing a capability

3.1.1. Traffic categories

We define three bandwidth categories on a link between two Tumbler-deployed routers: connection, priority, and best effort. As shown in Fig. 1, the connection class accounts for 5% of the link capacity, and is reserved solely for forwarding capability requests. Priority class refers to the privileged traffic labeled with capabilities and accounts for up to 60% of the link capacity. The remaining link capacity is used for best-effort class that serves the legacy traffic with no guarantees. The ratio between the three classes are chosen according to recent works [8,9,13,14]. Note that the above values serve as a baseline. Transit ASes can alter the allocated proportions of the classes themselves. Traffic class information is encoded in packet headers so that intermediate routers can interpret and forward different packets accordingly.

3.1.2. Request configuration

In Tumbler, end hosts can freely request capabilities with the desired bandwidth. An example is given in Fig. 2, where source k in source AS K is attempting to establish a capability with destination j in destination AS J. Source k generates a capability request packet that includes the following information: (1) its desired bandwidth number bw_k, (2) an expiration time t_{exp} for the capability, and (3) the source-AS identifier $AID_{(K)}$ into the capability request packet (see Section 4.1). Then k sends the packet via the routing path to the destination J.

3.1.3. Link access admission

At each AS on the path, the source's request needs to be approved by both the ingress and egress routers of the current AS. Each router independently keeps a corresponding *competition factor* (namely c-factor) for every leaf (i.e., source) AS. Competition factors reflect source domains' bandwidth utilization and reputation (see Section 3.2 for more details). Tumbler leverages competition factors to determine the maximum amount of link bandwidth that an AS can obtain.

Connection class fair sharing. Routers first determine how much connection bandwidth is available to each leaf AS. Then routers rate limits connection bandwidth for each active leaf AS based on the competition factors that each router keeps for the active ASes. That is, each router performs weighted fair sharing among the leaf ASes that are sending capability requests through the router using the competition factors associated with the leaf ASes. If a source AS sends more request packets than its share of connection bandwidth, the excessive requests over the limits will be dropped.

An example of bandwidth request is shown in Fig. 2. Source k's request is sent through transit AS 1 and AS 2. At ingress routers[1] $R_{(1,K)}$ and $R_{(1,J)}$, respectively 1 competition factor is involved in the weighted fair sharing since no flow from other leaf AS is currently forwarded by the router. Similarly, at egress routers $R_{(1,2)}$ and $R_{(2,J)}$, more competition factors (2 and 3, respectively) are involved since traffic converges.

Priority class fair sharing. Routers bound the amount of reservable priority bandwidth for each leaf AS proportional to its share of connection bandwidth. Furthermore, individual requests from the same AS will obtain an equal share to the unused part of the priority bandwidth obtained by their leaf AS, which eventually provides every source a real-time upper bound for priority-class allocation, denote as bw^{upper}. If the desired bandwidth of source k does not exceed bw^{upper}, the request will be approved by link access admission, otherwise the request will be denied.

3.1.4. Capability generation and bandwidth allocation

If a request is denied at intermediate routers, a denial packet will be sent back to the source, otherwise each egress router finalizes the access admission of each transit AS, and puts a cryptographic token (presented with a red rectangle in Fig. 2) into the request packet (details listed in Section 4.3). Hence, in the successful case, a request packet with a new generated capability (a chain of cryptographic tokens collected from all the transit ASes) will arrive at the destination. For request confirmation the destination j authorizes this capability by simply sending it in reverse back[2] to k. When the request has not complied with the destination's policy, the destination will inform the source by a denial message. All request failure packets (at intermediate routers or the destination) can be sent using best effort.

To eliminate the misused bandwidth by the denial requests, initially (after the access admission is approved) all the traversing routers temporarily reserve the desired bandwidth for source k for 1 s (covers most of the round-trip times in TCP connections).[3] If the confirmation packet with the generated capability arrives within 1 s, the allocation is confirmed; otherwise the temporarily reserved bandwidth will be released.

3.2. Data transmission and feedback regulation

After receiving the confirmation from the destination, the source can now send data tagged with the obtained capability. Allocated bandwidth indicated by the capability enables a guaranteed end-to-end communication. Tumbler capabilities are valid only during a short period of time; hence to retain their link access bandwidth, end hosts need to periodically renew their capabilities. Although inconvenient for the hosts, the short validity periods enable ASes to perform traffic engineering more flexibly. That is ASes can continuously adjust the bandwidth allocation among the end hosts based on the latest competition factor values.

As mentioned in Section 3.1, the competition factors of leaf ASes are used in the weighted fair sharing during the link access admission. In Tumbler, a competition factor is computed through

[1] The denotation of the routers in Fig. 2 uses the following way: $R_{(X,Y)}$ refers to the border router of AS X that connects to AS Y.

[2] For the ease of illustration, we simply assume the specified routing path is symmetric. We will discuss the capability establishment with asymmetric routes in Section 7.3.

[3] According to an experiment on TCP round-trip times [25], around 95% of the sampled connections have a RTT within 1 s. We thus suggest 1 s for the temporary allocation as a trade-off. Still, in a real situation the duration of the temporary allocation can be configured flexibly.

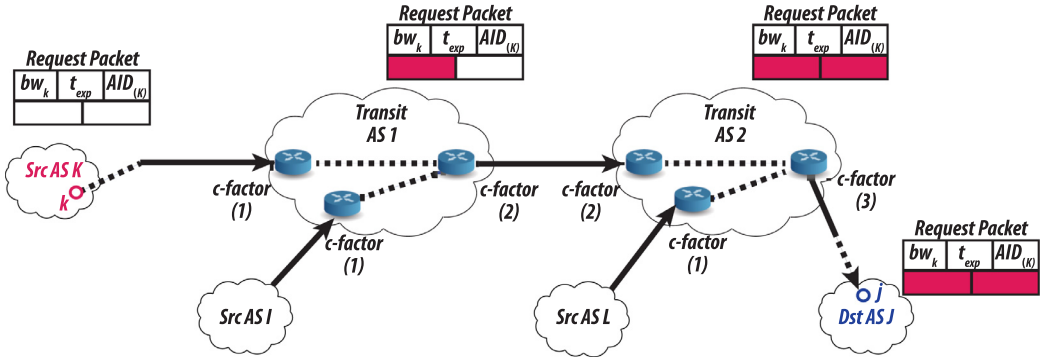

Fig. 2. A successful capability establishment between source k and destination j. Traversing routers perform link access admission independently based on the active competition factors, and further egress routers generate cryptographic tokens on the fly.

two sub-factors, namely *utilization factor* and *reputation factor*. Traversed routers independently maintain and update the two sub-factors of a leaf AS according to the leaf AS's bandwidth utilization and reputation.

To determine the bandwidth utilization of an AS, each router leverages an off-line accounting mechanism. Based on the obtained historical allocation statistics of the priority bandwidth, the utilization factor of an AS is periodically updated. Basically domains with more bandwidth consumptions will end up with higher utilization factors. The updating process of the utilization factor also take into account the length of updating intervals and the correlation with historical allocations. We give more details on the utilization factor update in Section 4.4.1.

Concerning bandwidth reputation, routers judge a reputation factor for a leaf AS via dedicated monitoring results. Traffic violations on either (1) request abuse (over-request case) or (2) data flooding (over-use case) will be detected. Then the reputation factor of a suspicious leaf AS will be degraded accordingly to throttle its link access. Tumbler selects the unused capabilities as the sign of over-request violation. Over-use violation is comprehensively measured both from the amount of over-sending traffic per-AS (coarse granularity) and the number of actual malicious flows (fine granularity); furthermore, recurrent AS violation is considered to avoid flooding replay.

With the feedback from the above traffic accounting and monitoring, transit ASes will be able to update the two sub-factors for each leaf AS. Thereby, the subsequent capability establishments can be regulated based on the updated competition factors. Consequently, leaf ASes with higher bandwidth consumption and reputation will have more advantages on bandwidth allocation. We will elaborate how competition factor works in Section 4.4.

4. Tumbler protocol design

In this section, we first describe how an end host sets up a request with necessary parameters. Then we specify the process of access admission and the cryptographic operations to generate a capability. Lastly, we introduce our detailed design for competition factor.

4.1. Request configuration

To initialize a capability, the source configures the following parameters in a request.

- **Bandwidth number** (1 byte): chosen from a pre-defined bandwidth set $\mathscr{B}$. $\mathscr{B}$ regulates the range of possible requesting bandwidth volumes by setting the corresponding bandwidth numbers in a min-max order. In a successful capability establishment, the amount of bandwidth referring to the chosen bandwidth number will be allocated at each transit AS.
- **Expiration time** (4 bytes): describes the validity period of the requested capability. In Tumbler, expiration time is a timestamp computed by adding the current time with a fixed, short slot (e.g., 60 s).
- **Source-AS identifier** (32 bytes): classifies flows from different leaf ASes and allows efficient flow operations such as link access admission, accounting, and monitoring.

4.2. Access admission

A router performs two tasks when it receives a capability request: (1) ensures that the request rate of the source AS (where the request is originated) is below the maximum rate for the AS and (2) determines if the router's reservable bandwidth satisfies the bandwidth demand of the request.

For the first task, a Tumbler router keeps a competition factor for each leaf AS to enable weighted fair sharing of the connection bandwidth. The requests from different leaf ASes are put into separate queues whose length is proportional to each AS's competition factor. A router will process a request when the corresponding queue is not fully occupied. Namely, each active leaf AS K shares BW_{KC} from the connection bandwidth:

$$BW_{KC} = \frac{C_K}{\sum_{i=1}^{n} C_i} \cdot BW_{connection}, \tag{1}$$

where there are n active leaf ASes and C_K is the competition factor of leaf AS K. $BW_{connection}$ is the total connection bandwidth on the link.

If the queue is not full, the request will be processed. Now the router calculates the priority bandwidth upper bound BW_{KP} for AS K:

$$BW_{KP} = \frac{C_K}{\sum_{i=1}^{n} C_i} \cdot BW_{priority} \cdot (1 - r_b), \tag{2}$$

where $BW_{priority}$ is the total priority link bandwidth. Considering that the traffic may be asynchronous with burst, a buffer ratio $r_b \in (0,1)$ (e.g., 15%) is maintained at each link during the weighted fair sharing. Next, the router differentiates the bound BW_{KP} above for every requesting source. Specifically, each router maintains a real-time *accounting table* (*AT*) with the active capabilities and a request rate for each leaf AS, from which the router can update AS K's used priority bandwidth BW_{KU} and the number of requests R per second. Denote b_{max} as the maximum bandwidth according to the bandwidth set $\mathscr{B}$. Then during the current second, an upper bound bw^{upper} for every request from leaf AS K can be calculated as:

$$bw^{upper} = min\left\{b_{max}, \frac{(BW_{KP} - BW_{KU})}{R}\right\}, \tag{3}$$

Each router will judge if the requesting bandwidth from AS K exceeds the bound bw_{upper}. If the bandwidth on demand is within the bound, capability establishing will proceed.

4.3. Tumbler capabilities

Tumbler capabilities are built at each egress router when the access admission at the current AS is approved. According to the ongoing request packet, a capability $\mathscr{C}$ will be generated as:

$$\mathscr{C} = MAC(AS\ 1)||\cdots||MAC(AS\ i)||\cdots||MAC(AS\ M), \quad (4)$$

where there are M transit ASes on the path and each MAC is an authentication token of the transit AS. MAC stands for the cryptographic message authentication code (e.g., CBC-MAC [26]). In Tumbler, MAC function is calculated as:

$$MAC(AS\ i) = MAC_{s_i}(bw_k||t_{exp}||AID_{(K)}||flow_{(k)}), \quad (5)$$

where s_i is a secret MAC key known only to AS i. Inputs bw_k, t_{exp}, and $AID_{(K)}$ are the bandwidth number, expiration time, and AS identifier of a source k, respectively.

Additionally, to bind the capability with each end host, a *flow identifier* (or flow ID) $flow_{(k)}$ is set as one input of each MAC function. Flow ID can simply be derived from the 5-tuples of packet header and has a fixed size by using a hash function[4]:

$$hash(src\ IP||dst\ IP||src\ port||dst\ port||protocol\ num). \quad (6)$$

During data transmission, source k puts its capability into the packets, together with the corresponding MAC inputs (namely bw_k, t_{exp}, $AID_{(K)}$, $flow_{(k)}$). Data packets will be checked at every transit AS, and forwarded to the next hop only when the verification of the MAC is correct. A packet with corrupted or stale capability will be dropped immediately.

4.4. Competition factor

Competition factor is a 1-byte parameter that determines the bandwidth availability of a leaf AS. Each router updates its local competition factors based on each leaf AS's bandwidth utilization and reputation factors at the adjacent link. Conceptually, the bandwidth utilization factor is relatively static, while reputation factor changes more dynamically (e.g., by taking traffic violation into account). For a leaf AS K whose link utilization factor is U_K and reputation factor is R_K, its competition factor C_K is defined as:

$$C_K = U_K^{\gamma} \cdot R_K^{1-\gamma}, \quad (7)$$

where γ is a weighted coefficient that satisfies $0<\gamma<1$. When traversed router intends to enforce legitimate traffic, a small γ (closer to 0) will be chosen; and a bigger γ (closer to 1) will be chosen to promote bandwidth consumption. Note that we leverage weighted geometry mean (rather than arithmetic mean) in Eq. (7) to ensure that both utilization and reputation factors have a significant effect on the change of competition factor. Therefore, when either utilization or reputation factor becomes minimal, the corresponding competition factor will become minimal as well.

Then we describe how utilization and reputation factors are updated as follows:

4.4.1. Utilization factor update

Utilization factor (1 byte) is by default ranging between two bound values U_{min} and U_{max}. A new participating AS will be assigned with U_{mid} that equals to $(U_{min}+U_{max})/2$; while an existing AS's utilization factor is updated every utilization interval T. T is a pre-defined value which indicates how many minutes an interval has. The amount of the allocated bandwidth from AS K is denoted as $A_{(K,\,i)}$, where the subscript i refers to the ith previous slot from the current updating interval. The revised utilization of AS K (denoted as $B_{(K,\,i)}$) is calculated taking historical allocation (i.e., $B_{(K,i+1)}$) into account. Parameter α is a corrective constant that satisfies $\alpha < T$. Specifically, AS K's current utilization factor U_K is calculated as follows:

$$B_{(K,i)} = A_{(K,i)} + \frac{\alpha}{T} B_{(K,i+1)}, \quad (\alpha < T)$$

$$U_K = max\left\{U_{min},\ \frac{U_{max} B_{(K,0)}}{A_{(0)}}\right\}, \quad (8)$$

where $A_{(0)}$ refers to the entire allocated bandwidth at the router during the current interval. As shown in Eq. (8), U_K has a decreasing correlation to older bandwidth statistics.

In Tumbler, the update of the utilization factor leverages an off-line bandwidth accounting mechanism. Each router gathers the generated capability information with its source-AS identifier, bandwidth number and expiration time. Since the lifetime of a capability is fixed, the expiration time of an accounting entry implies the starting time of the allocation. Thus each router can quickly calculate for each source AS, the sum of the allocated bandwidth that is valid during the current interval. Consequently, $A_{(K,\,0)}$ and $A_{(0)}$ can be easily obtained. As in Eq. (8), in order to update U_K for AS K, the router only needs to locally maintain the revised utilization of the previous interval (i.e., $B_{(K,\,1)}$), and calculate U_K together with $A_{(K,\,0)}$ and $A_{(0)}$ that are returned from the off-line accounting.

Concerning the parameters in Eq. (8), T determines the update frequency. We calculate U_K with a T on the order of minutes or hours (but not seconds) to avoid expensive operations that tend to real-time per-flow accounting. Parameter α reflects the correlation to historical allocations. When choosing a larger rate of α/T, historical allocations will have a stronger influence on the current utilization factor. In Section 6.2, we further illustrate the effect of T and α on utilization factor using real-world network traffic.

4.4.2. Reputation factor update

Reputation factor (1 byte) is also by default set between two bounds R_{min} and R_{max}. Similar to utilization factor, a new participating AS will obtain a reputation value R_{mid} that equals to $(R_{min}+R_{max})/2$.

A leaf AS's reputation factor is updated more frequently with smaller time windows, with respect to two aspects of traffic violations. The first violation is over-request, when an AS sends intensive requests during the stage of capability bootstrapping. We depict over-request violation of an AS by the number of unused capabilities, meaning the established capabilities need to be used at least once as massive idle allocations are highly likely to indicate request flooding. With an over-request monitoring window t_r (which is equal to the setting of the expiration time slot, e.g., 60 s), each router checks if a given AS has a ratio of unused capabilities more than $r\%$. If so, each exceeding capability will cause a decrease of the reputation factor by an over-request penalty β:

$$R_K = max\{R_{min},\ R_K - \beta \cdot N_{r\%}\}, \quad (9)$$

where $N_{r\%}$ is the number of unused capabilities that exceeding $r\%$ of all the capabilities. The unused capabilities can be detected as follows. Each router updates a Bloom filter [28] that stores all the used capabilities for the past t_r seconds. The entries (bw_k, t_{exp}, $AID_{(K)}$, $flow_{(k)}$) set into the Bloom filter are the same in a router's accounting table AT. Therefore, by checking the Bloom filter with all the recorded capabilities in the AT, routers can efficiently verify the unused capabilities per leaf AS.

Since each capability is built with associated bandwidth allocation, the second aspect of flow violation happens when the source

[4] To further prevent flow ID spoofing or hijacking, Host Identity Protocol [27] can be leveraged to ensure the origin of a source.

sends more traffic than its allocation (namely over-use or out-of-profile case). Specifically, during an over-use monitoring window t_u (e.g., 10 s), each ingress/egress router monitors its inbound and outbound traffic according to the following three aspects:

- **Out-of-profile domain traffic** refers to the amount of bandwidth overused entirely by a leaf AS. During every monitoring window t_u, routers compare the actual utilization in aggregation $u_{(K,t_u)}$ with the allocated bandwidth $ab_{(K,t_u)}$ indicated by the bandwidth numbers of flows from a leaf AS K. Then the out-of-profile traffic O_K is calculated as $max(u_{(K,t_u)} - ab_{(K,t_u)}, 0)$.
- **Overusing flows** are additionally counted on the basis of each leaf AS. The routers detect the number of over-use flows during the monitoring window. *Detected capability will be blocked for a capability slot (e.g., 60 s) until it expires.* However, counting every over-use flow could cause excessive sensitivity to the monitoring system. Instead, we set two thresholds for the violation detection, a low threshold TH_{low} equals to the allocated bandwidth of a flow, and a corresponding high threshold TH_{high} determined by its TH_{low}. We desire that flows sending over TH_{high} will be detected with probability 1 while flows sending less than TH_{low} will never be counted. This goal is easily achievable with a large flow detection mechanism [29] where each flow identifier is associated with a bandwidth counter. The routers can update an over-use monitoring table $\mathscr{T}$ with all ASes' over-use flow numbers in a time window.
- **Recurrence detection** is aimed to prevent against flooding flows sent from certain leaf ASes periodically. Namely, a recurrence monitoring is employed to detect the appearance of an attack origin that was historically malicious. If any of the above two monitoring processes show a violation from a leaf AS, the AS will be added into a router's recurrence blacklist $\mathscr{L}$ for a long period of time. Routers will impose stern sanctions to these leaf ASes that are in the blacklist.

Based on the feedback of the over-use monitoring, routers update the reputation factors of the ASes. For instance, each router first checks if AS K has valid allocation during the current time window. If so, AS K is considered as active. The router will furthermore check from the over-use monitoring table $\mathscr{T}$ if there is any over-use flow from an AS K. If not, a bonus value δ is added to its reputation factor:

$$R_K = min\{R_{max},\ R_K + \delta\}. \tag{10}$$

Otherwise, R_K will decrease according to both the out-of-profile domain traffic and the number of overusing flows. R_K is set as in Eq. (11) when AS K is not in the recurrence blacklist $\mathscr{L}$:

$$R_K = max\left\{R_{min}, R_K\left(1 - \frac{O_K}{u_{(t_u)}}\right) - \psi N_K\right\}; \tag{11}$$

Otherwise, R_K is set to:

$$R_K = max\left\{R_{min}, R_K\left(1 - \frac{O_K}{u_{(t_u)}}\right) - \omega N_K\right\}, \tag{12}$$

where $u_{(t_u)}$ refers to the total traffic during time t_u, ψ, ω are denoted as decreasing coefficients, and N_K is the returned over-use flow number from the monitoring table $\mathscr{T}$. When choosing of the above parameters, we have:

$$\delta < \psi \ll \omega \tag{13}$$

As a punishment, ψ is set larger than δ. Thus one AS has to keep consistently legitimate behavior for a sufficiently large reputation factor. We further set ω much larger than ψ as recurrence traffic will have a higher penalty.

When AS K is not active during the current monitoring window, routers decrease R_K by ϵ when R_K is larger than R_{mid}. Hence an inactive AS will eventually be adjusted to the same reputation factor as a new participating AS. Reputation factors lower than R_{mid} will remain unchanged during inactive intervals.

If the update of reputation factor does not mitigate the link congestion, the router enforces stricter monitoring by shortening window sizes. Thus, traffic violation will be rapidly detected with a more accurate monitoring. Detailed update procedures of the reputation factor is summarized in Algorithm 1.

Algorithm 1: The update of reputation factor for AS K

$AID_{(K)}$ - valid source AS K identifier;
`IsOverReq`$(AID_{(K)})$ – returns *True* if AS K is over-requesting during the current monitoring window t_r, *False* otherwise;
`IsActive`$(AID_{(K)})$ – returns *True* if AS K is active during the current monitoring window t_u, *False* otherwise;
`IsBlacklist`$(\mathscr{L}, AID_{(K)})$ – returns *True* if AS K is in the blacklist $\mathscr{L}$, *False* otherwise;
`GetCountFromTable`$(\mathscr{T}, AID_{(K)})$ – returns the number of overused flows from monitoring table $\mathscr{T}$.

Every t_r seconds, do
if `IsOverReq`$(AID_{(K)})$ **then**
  $R_K = max\{R_{min},\ R_K - \beta \cdot N_{r\%}\}$;
end

Every t_u seconds, do
if `IsActive`$(AID_{(K)})$ **then**
  $N_K =$ `GetCountFromTable`$(\mathscr{T}, AID_{(K)})$;
  if $N_K == 0$ **then**
    $R_K = min\{R_{max},\ R_K + \delta\}$;
  end
  else if `IsBlacklist`$(\mathscr{L}, AID_{(K)})$ **then**
    $R_K = max\{R_{min},\ R_K(1 - \frac{O_K}{u_{(t_u)}}) - \omega N_K\}$;
    else
      $R_K = max\{R_{min},\ R_K(1 - \frac{O_K}{u_{(t_u)}}) - \psi N_K\}$;
    end
  end
end
else if $R_K > R_{mid}$ **then**
  $R_K = max\{R_{mid},\ R_K - \epsilon\}$;
end

5. Analysis

In this section, we formally analyze the main security aspects and the deploying overhead of Tumbler.

5.1. Security analysis

Link access analysis. During the capability establishment, Tumbler guarantees the scalability of the requests via competition factors on the basis of each leaf AS, regardless of the actual location of the ASes (close to the ongoing link or not). Basically, assuming all the participating ASes have a similar utilization level on the given link, then for each of them, the connection-class link access is $O(1/S)$ where S is the total number of the ASes. Priority-class link access keeps proportional to the access of connection class, with a ratio r_p that approximately equals to $\frac{BW_{priority}}{BW_{connection}}$ (where $BW_{priority}$ and $BW_{connnection}$ refer to the link capacity of each bandwidth category). Then, for a given AS with n end hosts, each end host will share a link access of $O(1/nS)$ and $O(r_p/nS)$ at connection and priority class, respectively.

Concerning contaminated ASes, their link accesses will shrink from the initial access $O(1/S)$ since the link router will

accordingly reduce their competition factor as a punishment. On the other hand, benign ASes will share a better link access that approximately equals to $O(\frac{1}{S-S_c})$ where S_c is the number of contaminated ASes. Hence, Tumbler provides scalable and adaptable link access for leaf ASes and their residing end hosts. We further show how this property enable a resilient defense to DoC attack [15] in Section 6.1.

Botnet behavior. In this analysis, we consider different behavior of botnets. Implied by Tumbler enforcement, if the bots from an AS keeps sending massive traffic aimed at congesting a link. Link router will quickly detect the mis-behavior and lower the corresponding competitor factor. Malicious traffic will also be blocked and the source-AS identifier will be added to the blacklist. Botnet may try to bypass the detection by occasionally flooding the bottleneck. However, for each time the bots launching the attack, the competition factor of the source AS will be reduced. Since bonus value δ is configured smaller than both decreasing coefficients (ψ and ω), it takes a long time for a pre-malicious AS to recover its reputation factor, and even a longer time to get eliminated from the blacklist. Once the botnet conducts a recurrent attack, it will be severely punished. As a result, such sophisticated attack requires even more effort than passive flooding. We further evaluate our defense against these attacks in Section 6.3.

Capability replay. We employ MAC functions to create Tumbler's capabilities. Thus it is critical to consider replay attack as a potential threat. Instead of including nonce, we leverage solely a 32-bit timestamp (namely, the UNIX time) as one of the inputs of the MAC calculation to provide freshness for a certain capability. As introduced in Section 4.1, the timestamp refers to the expiration time of a reservation and will not be rotated as the validation of a capability is on the order of seconds (while we use a modulo 2^{32} clock). Hence, when the adversary intentionally replays a previous capability with a stale expiration time, the tagged packets will be dropped at a router. Consequently, replay attack can be prevented.

Malicious AS. Besides botnets in the leaf ASes, transit ASes may be contaminated and collude with each other to assist the attacks. These ASes could ignore the malicious traffic ongoing through them and remain (or even increase) the competition factors belonging to the malicious leaf ASes. Denote the set of legitimate and malicious transit ASes as P and U, respectively. When the out-of-profile traffic goes beyond the boundary of U, the legitimate ASes will handle the traffic strictly with the policy of competition factor adjustment. Flooding traffic will be throttled effectively in set P. In other words, adversary can fool only the ASes colluding in U, thus have no gain from such attacks.

5.2. Overhead analysis

Computational overhead. We analyze the scalability property of Tumbler in terms of overhead. To establish a capability, every request is processed with respect to the current competition factor; thus the update of the competition factors will not cause latency to the packet forwarding. For capability generation and verification, Tumbler relies on basic operations and lightweight cryptographic functions such as MAC and hash that can be fast processed [8]. Specifically, we deploy an AES-based CBC-MAC [26] in which AES New Instructions (AES-NI) [30,31] is employed to enhance the performance of MAC calculation.[5] It deserves to mention that the ratio of AES-NI enabled processors among all new Intel CPU products is also increasing. During the past 6 years, this ratio has increased from 30% (in 2010) to over 90% (in 2015) according to Intel's Processor Feature Filter [32].

Table 1
Storage overhead estimation at a router.

Stage	Operation	Maximum storage (MB)
Capability establishing	Competition factors	0.15
	Request-rate counter	0.1
	Accounting table	41.4
Data transmission	Utilization factor update	0.1
	Over-request detection	1.03
	Large flow detection	0.002
	Flow blocking	19.2
	Over-use domain traffic	0.1
	Over-use flow number	0.1
	Recurrence blacklist	0.16

Bandwidth overhead. We then analyze the bandwidth overhead introduced by Tumbler. To run the protocol, only the request parameters and the generated capability are put into each packet. For the entire configuration information, 69 bytes (1 byte for bandwidth number, 4 bytes for expiration time, both 32 bytes for AS identifier and flow identifier[6]) are required. Similar to other capability-based schemes (e.g., SIFF [7]), relatively short MAC (e.g., first 6 bytes truncated from the original CBC-MAC output) can be leveraged with sufficient security margin achieved in the meantime. Moreover, the average AS-level path length in the Internet is 3.9 for IPv4 and 3.5 for IPv6 (without AS prepending) [34], which also indicates that the size of most capabilities will be limited. Therefore, the bandwidth overhead of Tumbler is insignificant.

Storage overhead. We further analyze the storage overhead of our protocol. A summary of the storage overhead at a router is shown in Table 1.

On the stage of capability-establishing, Tumbler keeps per leaf AS a competition factor, as well as the corresponding utilization and reputation factors. Since each factor is encoded in 1 byte, a memory of 150 KB is required at a router (We consider the number of ASes as 50,000 [17], the same as follows). Additionally, when assigning 2 bytes per AS for the request-rate counter, the memory cost is 100 KB.

An on-line accounting table is maintained for access admission, but only active capabilities are recorded. As the size of each entry S_{entry} is 69 bytes, the total bytes in storage will be bounded by $S_{entry} \cdot BW_{priority}/b_{min}$, where $BW_{priority}$ is the total priority capacity and b_{min} refers to the minimal bandwidth allocation according to bandwidth set $\mathscr{B}$. Considering a 10 Gbps link capacity and a 10 Kbps b_{min}, the above storage bound is just 41.4 MB (considering the ratio of priority class is 60%, same as below).

During data transmission, the utilization factor updating leverages off-line accounting, which costs no overhead to router's memory except the per-AS revised utilization for the previous interval. It therefore costs 100 KB when using 2 byte to store this value for each AS.

Additionally, each router stores a Bloom filter [28] for used capabilities in over-request detection. Given the false positive rate p and the expected number of inserting items n, the size of the Bloom filter m (in bits) is

$$m = -\frac{n \ln p}{(\ln 2)^2}. \tag{14}$$

When the link capacity is 10 Gbps and b_{min} is 10 Kbps, the maximum number of insert items is 6×10^5, then the storage of the Bloom filter is at most 1.03 MB with a 0.1% false positive.

[5] AES-NI is a cryptographic microprocessor instruction set proposed by Intel in 2010. With the support of AES-NI, CBC-mode AES encryption only requires 4.15 CPU cycles per byte for a 1 KB buffer [31].

[6] For hash function, we choose SHA-3 [33] with an output of 256 bits.

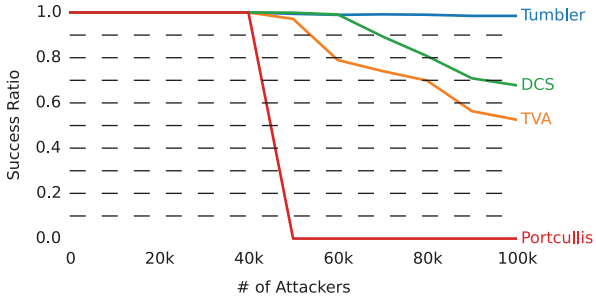

Fig. 3. The average successful ratio of the legitimate end hosts when requesting capabilities through the victim link with a fully deployment of each defense mechanism. For Tumbler, we set β to 1 and r% to 75%.

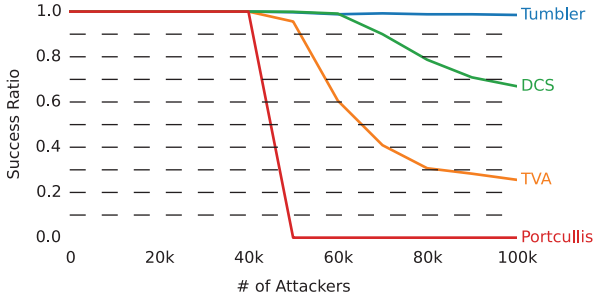

Fig. 4. The average successful ratio of the legitimate end hosts when requesting capabilities through the victim link with only partial deployment of each defense mechanism. For Tumbler, we set β to 1 and r% to 75%.

Large flow detection algorithm costs extremely small memory [29]. For flow blocking, when the link capacity is 10 Gbps and b_{min} is 10 Kbps, it costs a memory of 19.2 MB as each blacklisted entry is the violation flow identifier (32 bytes). Besides, Tumbler records the over-use synopsis of each original AS (out-of-profile domain traffic and over-use flow number) and a blacklist of recurrence domains. Using per-AS 2 bytes each for out-of-profile domain traffic and over-use flow counter, thus even when the blacklist contains 10% of the ASes, the entire overhead of over-use synopsis is less than 360 KB.

Note that the above analysis considers a worst case that the link is fully-occupied by all ASes. If taking only the number of leaf ASes into account, the cost will be even lower. Overall, Tumblers overhead is practical for today's routers with a 4 to 8 GB flash memory.

6. System evaluation

We experimentally evaluate the properties of Tumbler using ns3 [35] in this section. We first compare Tumbler with closely related proposals on capability bootstrapping, and further illustrate how competition factor can enforce adaptability in terms of bandwidth utilization and reputation.

6.1. Denial of capability attack

First, we evaluate Tumbler's resilience against DoC attacks in comparison with TVA [8], Portcullis [9], and DCS [11]. For simulation setup, we construct a realistic topology using CAIDA AS-relationship dataset [36]. We select a link between two high-degree transit ASes as "victim", and further define the scope of the simulation to those leaf ASes that have respectively 1-hop, 2-hop, and 3-hop distance to the victim. From the 28,685 involved leaf ASes, we randomly choose 100 leaf ASes as legitimate, and up to 500 leaf ASes as malicious. Every leaf AS has 200 end hosts.

In the DoC attack, end hosts from a legitimate AS overall send 200 requests per minute while each end host in a malicious AS has a 10 packets/s sending rate. Each request packet has a fixed size of 1000 bits. The capacity of the victim is set to 9.6 Gbps, 5% of which is used for capability bootstrapping. Figs. 3 and 4 present the successful ratio (average in 10 min) of the legitimate end hosts during capabilities requesting. In both fully deployment case (Fig. 3) and partial deployment case (Fig. 4) that only the routers of the victim link deploy defenses, Tumbler-enabled routers can quickly detect the over-requesting ASes. Tumbler eliminates the sources of the flooding at the bottleneck and guarantees the most effective protection even against 10^5 bots.

With DCS, per-AS fair sharing and path aggregation allow a differentiation link access for legitimate ASes. But when the number of malicious AS keeps increasing, new added ASes might not be dispersed closely to each other, thereby path aggregation becomes less effective in the Internet-scale simulation.

TVA performs hierarchical per-interface fair queueing to the requests. In our simulation, all leaf ASes have a distance to victim within 3 hops, but still the successful rate of TVA drops when more malicious ASes join. In the partial deployment case, TVA's ratio decreases significantly, as fewer attack packets are filtered at earlier hops.

Portcullis leverages computation puzzles tagged in each request packet. The requests with higher computation level will be forwarded first. However, as more bots join the DoC attack, more computation time is required to generate a high-level puzzle. In both deployment cases, the successful ratios of Portcullis start with nearly 100%, but sharply drop to zero (when setting a 4s request timeout based on Mirkovic et al.'s DDoS defenses testing report [37]).

6.2. Utilization adaptability

Real-world traffic experiment. To evaluate the adaptability of the utilization in Tumbler, we first conduct an experiment with the real traffic trace from San Diego Network Access Point (SDNAP [18]). According to a 24-h passive SDNAP monitoring snapshot, traffic from 13 different source ASes makes up the entire capacity, while 4 ASes hold nearly 90% of the link traffic.

Fig. 5 depicts the status of the competition factors in 1 day with different parameter settings. We choose 10, 20, 30, 60 for interval T during the utilization factor updating and the rate of α/T is set to be 1/3, 1/2, 2/3, respectively. As shown from all the sub-figures, even with different settings, the changes of the competition factors have no dramatic fluctuation. Each source AS's competition factor has a stable range (for instance, approximately between 100 and 120 for AS 1). Hence, utilization adaptability is effectively guaranteed.

Furthermore, taking Fig. 5(a), (c), and (e) in comparison, each AS's (competition factor) curve becomes more smooth when α/T increases as a stronger correlation to historical allocations limits oscillation. As can be seen from Fig. 5(c), (b), (d) and (f), a shorter time interval contributes to higher accuracy due to the timely update. The above observations confirm the design principle of T and α for utilization factor.

Realistic scenario experiment. Our second experiment shows how Tumbler will adapt to the bandwidth utilization of the leaf ASes in a real-life scenario. In this experiment we construct a topology (as in Fig. 6) that 10 children ASes are connected to a common parent AS whose upstream link capacity is 2.4 Gbps. Then, we select half of child ASes as company ASes while the other as resident ASes. Company ASes is configured to have a much higher bandwidth consumption during the working hours, comparing to their bandwidth utilization in the night; while resident ASes is on the

Fig. 5. The competition factors of 4 major source ASes in SDNAP traffic monitoring. Different parameter settings are configured for utilization factor update. In the left column, T (minute) remains the same while the rate of α/T changes. In the right column, T is set differently while α/T equals to 1/2. Reputation factor is the same for all ASes.

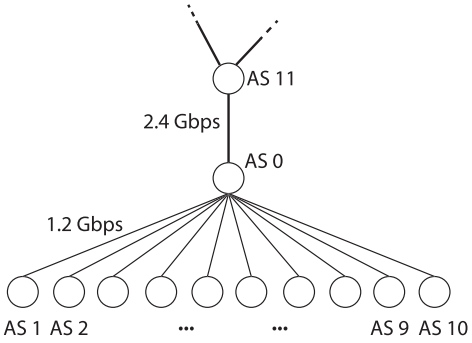

Fig. 6. The AS topology using in Sections 6.2 and 6.3. AS 1 to AS 10 are leaf ASes whose connections to AS 0 have a capacity of 1.2 Gbps. AS 0 and AS 11 are transit ASes and the link capacity in between is 2.4 Gbps.

opposite situation as resident Internet service is used mostly during the night.

When setting the utilization interval T to 10 min, the priority class ratio to 60%, and the corrective constant α to 1, Fig. 7 and Fig. 8 respectively present the change of competition factors for both AS categories in a 24-h simulation. As exhibited in Fig. 7, the bandwidth allocation of a company AS increases rapidly at around 8 a.m. and remains stable until a sharp drop at 18 p.m. The competition factor changes in step with the bandwidth consumption, leveling off during the working hours. Similarly, in Fig. 8 the bandwidth allocation from resident ASes surges during the leisure time (20 p.m. to midnight). Meanwhile, the competition factor ratio of a resident AS goes up accordingly. Specifically, during the period with peak bandwidth consumption, both kinds of ASes have the predominant competition factors whose ratios are close to 15%.

Through the feedback regulation of utilization factors, Tumbler provides synchronous adaptability to the allocation status of ASes, thus enabling dynamic bandwidth coordination among leaf ASes. Thereby, the priority bandwidth of the upstream AS can be assigned effectively.

6.3. Authorized packet flooding

Next, we test Tumbler's resilience against authorized packet flooding. The topology configuration is the same as in Fig. 6. We consider the most severe attack scenario where the attackers can coordinate both their attack strength (pulsing attack [10]) and location (rolling attack [11]), aiming to congest the victim link without exhausting their reputation.

In the experiment setting, we label children AS 1 as legitimate and the other 9 ASes as malicious. Those malicious ASes are further divided into 3 attacking groups (AS 2, AS 3, AS 4), (AS 5, AS 6, AS 7), (AS 8, AS 9, AS 10). To launch packet flooding attack in the priority class, each group will in turn (every 1 min) overuse their bandwidth (10 times larger than their allocation), while legitimate AS only sends in-profile traffic. For detection, we fix the over-use monitoring slot to 10 s, the well-behaving bonus value δ to 1, and the decreasing coefficients ψ, ω to 5, 50, respectively.

As shown in Fig. 9, the accepted packet ratio of AS 1 stays around 100% except a slight fluctuation at the beginning of every minute. After the first monitoring slot, the attacking flows will be detected and blocked, as presented in Fig. 10. Meanwhile, as can be seen in Fig. 11, the competition factor of AS 1 linearly increases for well behaving. In contrast, malicious ASes' competition factors drop steeply during each punishment. More importantly, the influence of the attack gradually shrinks because new capabilities from malicious ASes have a declining sharing to the link capacity. The figures lead to the conclusion that Tumbler provides effective DDoS protection and high motivation for leaf ASes to build long-term reputation.

7. Discussion

7.1. Integrating with existing protocols

Tumbler is a practical approach that can be deployed with today's Internet. Hereby we briefly illustrate how our proposal can coordinate with existing protocols.

Tumbler is workable without separate capability packets. Capability information can be piggybacked into a legacy packet, extending one shim layer above IP. Alternatively, Tumbler can also be implemented with the IPv6 format as an optional extension header, triggered to provide guaranteed bandwidth services. Furthermore, it is feasible to integrate the process of capability establishment into TCP handshakes [8,12]. For instance, initially when a source sends a request, the request can be combined with a TCP SYN packet. After the destination authorizes the capability, the response will be associated with a TCP SYN/ACK packet. Consequently, no extra circuit is required as data transmission will start ahead of bandwidth reservation.

The BGP speaker maintains the BGP routing table of a domain. Assisted by the speaker, an AS-level path to a certain destination can be obtained by end hosts of each leaf AS. Transit ASes can also contribute by announcing their compatibility of Tumbler via BGP updates. Hence the support of Tumbler on a given path will be known by a source before sending any packets. Note that the instability of BGP is not a concern as the vast majority of BGP routes are considerably stable [38]. Additionally overall long-lived routing paths are still used 96% of the time [39].

For the end hosts, we envision the support of Tumbler can be reflected via enhanced protocols such as DNS or ICMP. Specifically, based on the query of DNS service, a source is able to verify if an intended destination has upgraded Tumbler. Also, the source can send a modified ICMP packet to a destination, checking the compatibility of Tumbler. If the returned packet indicates that Tumbler is supported, the source can send a capability request; otherwise legacy traffic will be used in the subsequent communication.

7.2. Deployment and limitations

Providing adaptable DDoS protection for inter-domain traffic, Tumbler can be directly developed by the current network architecture and efficiently deployed with existing Internet protocols. Moreover, such lightweight protocol can be incrementally upgraded. Essentially, each Tumbler-enabled router independently maintains its competition factor and link configuration. No cooperation is necessarily required during the deployment.

We also envision that communication destinations (such as website servers) and vulnerable transit ASes are more likely to have higher incentive of deploying Tumbler as they are typically targeted or act as the bottleneck in the adversarial scenario. As shown in Section 6.1, even if only the bottleneck routers deploy Tumbler, the system provides high resilience against Denial of Capability attacks.

Nevertheless, in case that not all ASes (on the capability-established path) deploy Tumbler, an attacker will be able to flood a legacy router and thus to influence the priority-class traffic passing through. One possible solution is to add legacy routers a simple differentiating policy, giving higher preference to capability-enabled packets. Therefore, instead of implementing the entire Tumbler protocol, the damage of the flooding attack can be sufficiently eliminated.

7.3. Functional use cases

A crucial characteristic of our approach is the short validation of capabilities. End hosts on demand retain their priority-class bandwidth by renewing their capabilities. Thus traversing routers will be able to reclaim and rearrange the unused bandwidth flexibly. Although Tumbler makes bandwidth reservations, deploying Tumbler remains no negative effects on techniques such as inter-domain traffic engineering. Moreover, when armed with recent inter-domain multipath routing protocols,[7] we expect Tumbler to achieve even better performance in terms of reliability and functionality.

Specifically, end hosts can obtain multiple capabilities on several inter-domain paths for *load balancing* purpose so that the entire priority bandwidth can be maintained with high availability. In Tumbler, capabilities can be associated with different volumes of allocated bandwidth. Thus by default an end host can send most of the priority traffic on a primary path (e.g., the optimal path returned from BGP), with the remaining traffic sent on suboptimal paths. Therefore, even a failure happens on the main path, the communication can still work among backup paths so that *fault resilience* is achieved.

Asymmetric bandwidth can be supported by Tumbler through uni-directional capability requests. One possible solution is to add a request-type flag in the capability request indicating if the ongoing request is bidirectional or not. In the uni-directional case, for capability establishment the destination can specify a path (can also be the reversed one) and also launch a capability request back to the source. Note that in this case bandwidth will be respectively allocated and both directions of capabilities will be forwarded back to the source eventually. The source (and destination) will use these capabilities together in the transmitted packets afterwards.

7.4. Adoption incentive

Tumbler provides strong adoption incentive for both transit and leaf ASes. Since Internet transit prices shrink irreversibly every year[8] [44], traffic delivery beyond best effort becomes essential, functional services such as priority bandwidth make up the main revenue for most ISPs. As shown in Section 6.2, through the adjustment of the competition factor, Tumbler enables transit ASes to

[7] Although designing and evaluating a certain multipath protocol is out of scope of this paper, fruitful schemes have been presented and discussed in the literature. Such approaches include BGP-based solutions proposed by Cisco and Juniper Networks [40,41], MIRO [42], YAMR [43], and also novel protocols based on new Internet routing architectures [20–22].

[8] In 2015, the Internet transit price in U.S. region has a 33% decline, to only $0.63 per Mbps.

dynamically synchronize their priority traffic assignment with the actual demand of their customers. Thus, transit ASes can maximize their local bandwidth allocation in order to gain more economical benefits. Driven by the competition-based DDoS protection, transit ASes can enforce their traffic to be more well-behaved, which in turn reduces the extra management costs. Concerning leaf ASes, as long as they keep high reputation (e.g., not contaminated with botnets), they will be protected by Tumbler and hold advantages against those mis-behaving domains.

7.5. Protocol parameters

When updating the key factors in Tumbler, we introduce several parameters such as the over-request penalty β, the over-using coefficients ψ and ω, etc. We have not specified in the paper how to choose these values (with suggested numbers in Section 6), although for which plenty of tests and tweaking are involved. It is worth to mention that, adjusting the value for a parameter will result in different levels of defense sensitiveness as well as system dynamics. For instance, a bigger β reflects a strong reaction to request flooding; while a small ψ leads to a looser traffic enforcement. In this paper, we focus on the demonstration of our defense mechanism and we leave the selection of optimal parameter values for future work, or as one self-alter feature for ISPs deploying Tumbler.

8. Related work

A recent survey on DDoS defense is presented by Zargar et al. [45]. Here we focus mainly on the capability-based mechanisms that inspire the design of Tumbler.

In capability schemes [6–14], the destination explicitly authorizes token that indicates the desired traffic. This idea, first proposed by Anderson et al. [6], assumes a separate overlay for capability request packets, yet incurs expensive setup overhead. Later work improves the approach by removing the overlay request channel. **SIFF** [7] suffers from DoC attacks due to the same priority setting for both request and legacy traffic. **TVA** [8] leverages the ingress interfaces of the transit ASes to enforce a hierarchical fair queueing on connection access: Queues for directly connected links are assigned equally, and each queue is split recursively for further connected links (towards the customer ASes). Due to the disadvantage of the remote ASes and the large combination of potential paths, TVA is still susceptible to link-flooding attacks.

Portcullis [9] employs proof of work to achieve per-computation fair sharing for capability establishment. This puzzle-based scheme may be expensive to protect every packet. Moreover, the effectiveness of Portcullis decreases linearly with the rise of bots scale, and becomes insignificant as the botnet contains over 1 million bots [1].

A novel capability mechanism based on client reputation is proposed by Natu et al. [10]. The destination assigns each source a credit and a penalty value, indicating the degrees of trust. However, this per-source evaluation does not scale to the Internet. Intermediate routers also have low incentive to trust the judgment from the destination, which may misbehave and provide false information. Our approach is inspired by this reputation scheme. Instead, the reputation is established on the basis of leaf ASes. At

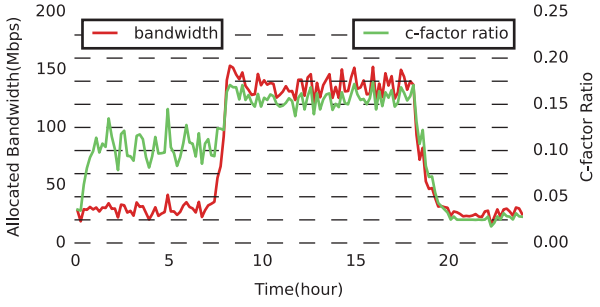

Fig. 7. The changes of bandwidth allocation and the competition factor ratio of a company AS in a day.

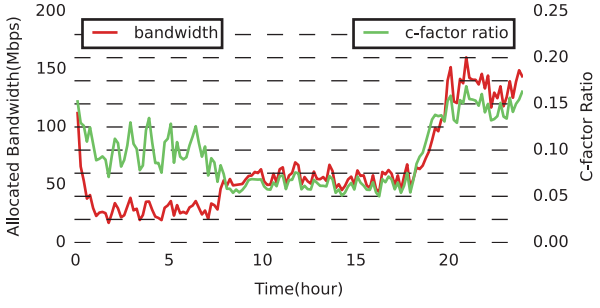

Fig. 8. The changes of bandwidth allocation and the competition factor ratio of a resident AS in a day.

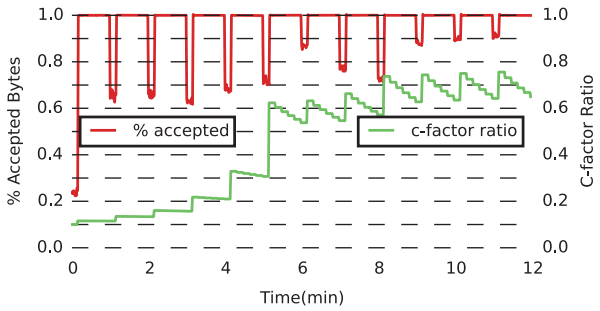

Fig. 9. The ratios of accepted packets and the competition factor of the legitimate AS.

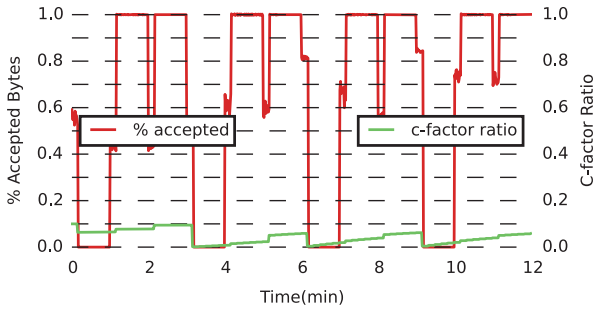

Fig. 10. The ratios of accepted packets and the competition factor from the malicious ASes (attacking group 1).

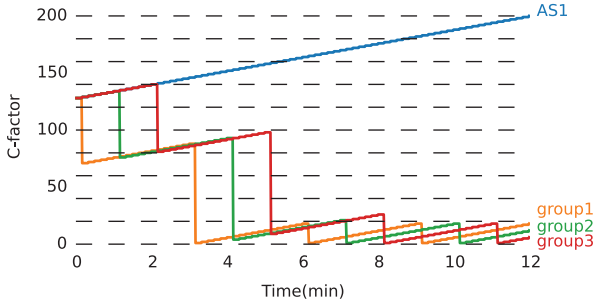

Fig. 11. The changes of competition factors with time. AS 1 is the legitimate AS and groups 1,2,3 refer to malicious ASes.

each hop, reputation factors are maintained independently, thus solving the issue of trust.

Recent capability approaches involve the concept of autonomous systems to provide scalable fair access for either the requesting channel [11] or the link bandwidth [12], or both [13,14]. In **DCS** [11] and **FLoc** [12], link access is shared equally among ASes and path aggregation is further implemented to provide differentiated accesses between contaminated and legitimate ASes. However, aggregation-based approaches are effective against non-uniform distributed bots, but still inefficient against wide-dispersed botnets. **STRIDE** [13] introduces various traffic paths–best effort, static, and dynamic. Through static and dynamic paths, STRIDE achieves guarantees for request channel and link bandwidth respectively, but only within a special scope of the Internet. **SIBRA** [14] improves STRIDE on how to build scalable protection in the Internet with effective traffic processing and enforcement. However, both STRIDE and SIBRA prevent flooding attacks under SCION architecture [22], while in contrast our protection is built on the top of the current Internet.

Besides, it is also crucial to compare Tumbler with existing Quality of Service (QoS) mechanisms. Such QoS schemes as **IntServ** [46] and **DiffServ** [47], fail to work efficiently in the Internet scale. Specifically, IntServ requires the intermediate routers to keep per-flow state as **RSVP** [48] is leveraged as the underlying mechanism. Thus, IntServ cannot guarantee end-to-end connectivity when a quadratic number of flows passing through the core routers. DiffServ deploys differentiated service with a dedicated traffic-classification mechanism. As essentially no bandwidth reservation happens, DiffServ fails to provide consistent link access across multiple domains.

9. Conclusion

This paper has presented Tumbler, an adaptable DDoS resilient mechanism in the Internet scale. Tumbler provides bandwidth guarantees for end hosts through competition-based fairness. Comprehensive simulations under sophisticated flooding strategies show that Tumbler achieves more effective protection than other capability schemes. Moreover, Tumbler introduces minimal overhead on traversing routers. We believe that Tumbler brings substantial adoption incentive to ISPs. To confirm the property of our approach, a full implementation of Tumbler is suggested for future work.

Acknowledgments

We would like to thank Taeho Lee, Hsu-Chun Hsiao, David Barrera, Christos Pappas, and Raphael Reischuk for their valuable feedback. We also gratefully acknowledge support from ETH Zurich, and the China Scholarship Council (CSC) program.

References

[1] Botnets remain a leading threat, 2013 (blogs.mcafee.com/business/security-connected/tackling-the-botnet-threat).
[2] Verisign Q1 2016 DDoS trends, 2016 (https://www.verisign.com/assets/report-ddos-trends-Q12016.pdf).
[3] Akamai detected over 400 reflection DDoS attacks leveraging DNSSEC protocol, 2016 (http://news.softpedia.com/news/akamai-detected-over-400-reflection-ddos-attacks-leveraging-dnssec-protocol-500497.shtml).
[4] Technical details behind a 400 Gbps NTP amplification DDoS attack, 2014 (https://blog.cloudflare.com/technical-details-behind-a-400gbps-ntp-amplification-ddos-attack/).
[5] Arbor networks 11th annual worldwide infrastructure security report, 2016 (https://www.arbornetworks.com/images/documents/WISR2016_EN_Web.pdf).
[6] T. Anderson, T. Roscoe, D. Wetherall, Preventing internet denial-of-service with capabilities, ACM SIGCOMM Comput. Commun. Rev. 34 (1) (2004) 39–44.
[7] A. Yaar, A. Perrig, D. Song, SIFF: a stateless internet flow filter to mitigate DDoS flooding attacks, in: Proceedings of the IEEE Symposium on Security and Privacy (S&P), IEEE, 2004, pp. 130–143.
[8] X. Yang, D. Wetherall, T. Anderson, A DoS-limiting network architecture, ACM SIGCOMM Comput. Commun. Rev. 35 (4) (2005) 241–252.
[9] B. Parno, D. Wendlandt, E. Shi, A. Perrig, B. Maggs, Y.-C. Hu, Portcullis: protecting connection setup from denial-of-capability attacks, ACM SIGCOMM Comput. Commun. Rev. 37 (4) (2007) 289–300.
[10] M. Natu, J. Mirkovic, Fine-grained capabilities for flooding DDoS defense using client reputations, in: Proceedings of the Workshop on Large Scale Attack Defense (LSAD), ACM, 2007, pp. 105–112.
[11] S.B. Lee, V.D. Gligor, A. Perrig, Dependable connection setup for network capabilities, in: Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), IEEE, 2010, pp. 301–310.
[12] S.B. Lee, V.D. Gligor, FLoc: dependable link access for legitimate traffic in flooding attacks, in: Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS), IEEE, 2010, pp. 327–338.
[13] H.-C. Hsiao, T.H.-J. Kim, S. Yoo, X. Zhang, S.B. Lee, V. Gligor, A. Perrig, STRIDE: sanctuary trail–refuge from internet DDoS entrapment, in: Proceedings of ACM Symposium on Information, Computer and Communications Security (ASIACCS), ACM, 2013, pp. 415–426.
[14] C. Basescu, R.M. Reischuk, P. Szalachowski, A. Perrig, Y. Zhang, H.-C. Hsiao, A. Kubota, J. Urakawa, SIBRA: scalable internet bandwidth reservation architecture, in: Proceedings of Symposium on Network and Distributed System Security (NDSS), 2016.
[15] K. Argyraki, D.R. Cheriton, Network capabilities: the good, the bad and the ugly, in: Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets), 2005.
[16] J.C.-Y. Chou, B. Lin, S. Sen, O. Spatscheck, Proactive surge protection: a defense mechanism for bandwidth-based attacks, IEEE/ACM Trans. Netw. (ToN) 17 (6) (2009) 1711–1723.
[17] CIDR report, 2015 (http://www.cidr-report.org/as2.0/).
[18] SD-NAP passive monitor, 2013 (http://www.caida.org/projects/sdnap/).
[19] Y. Rekhter, T. Li, A border gateway protocol 4 (BGP-4), 1995(RFC 1771).
[20] X. Yang, D. Clark, A.W. Berger, NIRA: a new inter-domain routing architecture, IEEE/ACM Trans. Netw. (ToN) 15 (4) (2007) 775–788.
[21] P. Godfrey, I. Ganichev, S. Shenker, I. Stoica, Pathlet routing, ACM SIGCOMM Comput. Commun. Rev. 39 (4) (2009) 111–122.
[22] X. Zhang, H.-C. Hsiao, G. Hasker, H. Chan, A. Perrig, D.G. Andersen, SCION: scalability, control, and isolation on next-generation networks, in: Proceeding of the IEEE Symposium on Security and Privacy (S&P), IEEE, 2011, pp. 212–227.
[23] X. Liu, A. Li, X. Yang, D. Wetherall, Passport: secure and adoptable source authentication, in: Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI), USENIX Association, 2008, pp. 365–378.
[24] T.H.-J. Kim, C. Basescu, L. Jia, S.B. Lee, Y.-C. Hu, A. Perrig, Lightweight source authentication and path validation, in: Proceedings of ACM SIGCOMM, ACM, 2014, pp. 271–282.
[25] J. Aikat, J. Kaur, F.D. Smith, K. Jeffay, Variability in TCP round-trip times, in: Proceedings of the ACM Internet Measurement Conference (IMC), ACM, 2003, pp. 279–284.
[26] CBC-MAC, 2015 (http://en.wikipedia.org/wiki/CBC-MAC).
[27] R. Moskowitz, P. Nikander, P. Jokela, T. Henderson, Host identity protocol, 2008(RFC 5201).
[28] B.H. Bloom, Space/time trade-offs in hash coding with allowable errors, Commun. ACM 13 (7) (1970) 422–426.
[29] H. Wu, H.-C. Hsiao, Y.-C. Hu, Efficient large flow detection over arbitrary windows: an algorithm exact outside an ambiguity region, in: Proceedings of the ACM Internet Measurement Conference (IMC), ACM, 2014, pp. 209–222.
[30] S. Gueron, Intel's new AES instructions for enhanced performance and security, in: Proceedings of the International Workshop on Fast Software Encryption (FSE), Springer, 2009, pp. 51–66.
[31] S. Gueron, Intel advanced encryption standard (AES) instructions set, 2010 (Intel Corporation).
[32] ARK: advanced search, 2015 (http://ark.intel.com/search/advanced/?s=t).
[33] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, Keccak sponge function family main document, Submission to NIST (Round 2) 3 (2009) 30.
[34] V. Asturiano, Update on AS path lengths over time, 2015 (https://labs.ripe.net/Members/mirjam/update-on-as-path-lengths-over-time).
[35] NS3 network simulator, 2013 (http://code.nsnam.org/ns-3-allinone).
[36] AS relationship dataset, 2015 (http://data.caida.org/datasets/as-relationships/).
[37] J. Mirkovic, S. Fahmy, P. Reiher, R.K. Thomas, How to test DoS defenses, in: Proceedings of the Cybersecurity Applications & Technology Conference for Homeland Security, 2009, pp. 103–117.
[38] J. Rexford, J. Wang, Z. Xiao, Y. Zhang, BGP routing stability of popular destinations, in: Proceedings of the ACM SIGCOMM Workshop on Internet Measurement, ACM, 2002, pp. 197–202.
[39] Í. Cunha, R. Teixeira, C. Diot, Measuring and characterizing end-to-end route dynamics in the presence of load balancing, in: Proceedings of Passive and Active Measurement Conference, Springer, 2011, pp. 235–244.
[40] BGP best path selection algorithm, 2015 (http://www.cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/13753-25.html).
[41] Configuring BGP to select multiple BGP paths, 2015 (http://www.juniper.net/techpubs/software/junos/junos90/swconfig-routing/configuring-bgp-to-select-multiple-bgp-paths.html#id-13280349).
[42] W. Xu, J. Rexford, MIRO: multi-path interdomain routing, in: Proceedings of ACM SIGCOMM, ACM, 2006, pp. 171–182.
[43] I. Ganichev, B. Dai, P. Godfrey, S. Shenker, YAMR: Yet another multipath routing protocol, ACM SIGCOMM Comp. Commun. Rev. 40 (5) (2010) 13–19.

[44] Internet transit prices–historical and projected, 2010 (http://drpeering.net/white-papers/Internet-Transit-Pricing-Historical-And-Projected.php).
[45] S.T. Zargar, J. Joshi, D. Tipper, A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks, IEEE Commun. Surveys Tuts. 15 (4) (2013) 2046–2069.
[46] J. Wroclawski, The use of RSVP with IETF integrated services, 1997 (RFC 2210).
[47] K.H. Chan, J. Babiarz, F. Baker, Configuration guidelines for DiffServ service classes, 2006 (RFC 4594).
[48] L. Zhang, S. Berson, S. Herzog, S. Jamin, Resource reservation protocol (RSVP)–version 1 functional specification, 1997 (RFC 2205).

Yao Zhang received the B.S. degree with distinction from Beihang University, Beijing, China, in 2011. He is currently a Ph.D. candidate at Key Laboratory of Mathematics, Informatics and Behavioral Semantics and School of Mathematics and Systems Science, Beihang University. From 2013 to 2015, he is with the Network Security Group, ETH Zürich as an academic guest. His research interests include network security and applied cryptography.

Xiaoyou Wang received the B.S. degree from Tsinghua University, Beijing, China, in 2015. She is currently a Master student at the Information Networking Institute at Carnegie Mellon University, Pittsburgh, United States. Her research interest is network security.

Adrian Perrig is a Professor at the Department of Computer Science at the Swiss Federal Institute of Technology (ETH) in Zürich, Switzerland, where he leads the network security group. He is also a Distinguished Fellow at CyLab, and an Adjunct Professor of Electrical and Computer Engineering at Carnegie Mellon University. From 2002 to 2012, he was a Professor of Electrical and Computer Engineering, Engineering and Public Policy, and Computer Science (courtesy) at Carnegie Mellon University; From 2007 to 2012, he also served as the technical director for Carnegie Mellon's Cybersecurity Laboratory (CyLab).

Zhiming Zheng received the Ph.D. degree from Peking University, Beijing, China, in 1987. He is currently the Professor of Mathematics at Beihang University and the Director of Key Laboratory of Mathematics, Informatics and Behavioral Semantics, Ministry of Education. His research interests include information security, complex information system, and dynamic system. He is the Editor in Chief of the journal *Mathematical Biosciences and Engineering* published by SPRINGER, and the journal *Mathematics in Computer Science* published by BIRKHAUSER.