Defending Against DDoS Attacks in Bloom Filter based Multicasting

XIAOHUA TIAN, Shanghai Jiao Tong University
WEI LIU, Shanghai Jiao Tong University
YU CHENG, Illinois Institute of Technology

Bloom filter (BF) based forwarding is an effective approach to implement scalable multicasting in distributed systems. The forwarding BF carried by each packet can encode either multicast tree or destination IP addresses, which are termed as tree oriented approach (TOA) and destination oriented approach (DOA), respectively. Recent studies have indicated that TOA based protocols have serious vulnerabilities under some distributed denial-of-service (DDoS) attacks, and raised doubt about deployability of BF based multicasting. However, security analysis for DOA based protocols is still unavailable. In this paper, we present a systematic analysis of security performance of BF based multicasting. Important DDoS attacks and the corresponding defending mechanisms are studied in the context of DOA. We have positive findings that DOA, with convenient enhancement, has a robust performance in resisting a variety of DDoS attacks that can deny service of TOA based protocols. Moreover, we reveal that TOA based protocols are prone to flow duplication attack when applied in the data center network (DCN). We propose a dynamic-sized BF mechanism to defend against flow duplication attack for TOA based protocols in the DCN. Simulation results are presented to validate our theoretical analysis.

CCS Concepts: • Networks → Routing protocols;
General Terms: Design, Performance, Security

Additional Key Words and Phrases: DDoS, multicast, Bloom filter, DCN

1. INTRODUCTION

Group communication is common in distributed systems such as the IP video delivery system and the data center network (DCN). For example, IPTV or video streaming service needs to disseminate shared video contents to multiple receivers over the Internet [Tian 2013], and web search or data backup operation needs to distribute query requests or file chunks to a group of servers in the DCN [Guo 2015]. Multicast mechanism could benefit the group communication in avoiding redundant traffic. With multicast, the same data packet is delivered over each link only once along a tree structure [Deering 1990]. As a network normally needs to accommodate a huge number of communication groups, scalability of multicast protocols becomes vitally important. Many schemes have been proposed towards a scalable multicasting [Costa 2006, Tao 2009, Ratnasamy 2006, Tian 2014], among which the Bloom filter (BF) based multicasting has attracted a lot of attention in recent years.

There are many variations of BF based multicast and all share the same design philosophy: encoding multicast routing information into a BF [Broder 2004] carried by each packet. Each forwarding node queries local forwarding states against the in-packet BF to determine the number of packet replications and the corresponding output interface for each replication. The implementation of BF based multicast protocols diverges into two classes, based on the content encoded into the BF. One class encodes multicast tree branches, such as FRM [Ratnasamy 2006], LIPSIN [Jokela 2009] and BloomCast [Särelä 2011]; this class is termed as the tree oriented approach (TOA) in this paper. The other class encodes the destination IP addresses in the BF such as AOM [Tian 2009] and DOM [Tian 2013, Tian 2014]; this class is termed as the destination oriented approach (DOA). The scalability of BF based multicast protocols is mainly credited to that each intermediate node only needs
to maintain limited forwarding states that are independent of the number of groups being supported [Ratnasamy 2006, Jokela 2009].

While BF based forwarding protocols were originally proposed for the inter-domain multicasting over the Internet, they can also be deployed in the DCN where large-scale and bandwidth-hungry applications prevail [Tian 2013, Li 2011]. Guo et al. design two scalable BF based forwarding schemes to implement in-network aggregation over massive concurrent shuffle transfers, where TOA and DOA are both considered [Tian 2013]. The DCN normally has a more controllable environment, where the network topologies are customized and the infrastructure is managed by a single authority. Such a controllable environment particularly facilitates the deployment of BF based multicasting. For example, the efficient and scalable data center multicast protocol (ESM) encodes the multicast tree into the in-packet BF with the help of a multicast manager, which exploits the multistage graph derived from the DCN topology [Li 2011].

Recent studies however show that BF based multicast protocols, particularly in the TOA class, have vulnerabilities to a variety of distributed denial-of-service (DDoS) attacks [Särelä 2012, Fahmy 2013, Antikainen 2011]. Tree branches encoded in the BF can be reversely engineered; a target receiver can be flooded through a packet injection attack; and subscribers can be prevented from leaving a multicast group. Moreover, it has been demonstrated that those security mechanisms proposed for TOA based protocols are overestimated and actually ineffective in the Internet environment [Antikainen 2011]. The negative findings then raise doubt about the deployability of BF based multicast mechanisms [Fahmy 2013, Antikainen 2011]. Nevertheless, the final answer is incomplete; the security performance of DOA based protocols are yet to be studied.

In this paper, we present a systematic study of the security performance of BF based multicasting. Some important DDoS attacks and corresponding defending mechanisms are studied in the context of DOA, which are the first-time contributions to the best of our knowledge. Whenever possible, security performance of the DOA class is compared to that of the TOA class, revealing how BF implementation designs are related to security performance. Moreover, we examine security performance of BF based multicasting protocols in DCNs, and reveal that TOA based protocol is prone to flow duplication attack when applied in DCNs. Specifically, this paper has the following contributions.

First, we provide a comprehensive study on packet injection attacks in the context of DOA, and reveal a new colluded packet injection attack pattern against both DOA and TOA. Specifically, we show how the packet injection attack can be launched to the basic DOA protocol, and then enhance DOA protocol with a bit permutation technique. We present a theoretical analysis on a sufficient condition that a packet can be falsely delivered to a target on purpose in the packet injection attack model. The derived results from our analysis provide a guideline for the BF design to achieve an objective security level. Moreover, we propose a scheme of source-driven tree construction with reverse bit permutation for DOA, which can defend against the colluded injection attack by tracing back to injecting bots in the Internet environment.

Second, we reveal security robustness of DOA compared with TOA. We show that DOA is inherently immune to the reverse engineering attack and the resubscription attack, which are convenient to launch in the context of TOA. We also reveal that the authentication technique can be more effective in DOA than in TOA or the IP multicast for defending against DDoS attacks. In particular, we analyze replay attack, colluded packet injection attack and silent subscription attack in the scenario where an authentication center (AC) exists.

Third, we conduct some novel studies of security performance of BF based forwarding in the DCN, and demonstrate flow duplication attack against TOA with both theoretical analysis and simulation studies. A dynamic-sized BF mechanism is proposed to defend against the flow duplication attack in TOA. Experimental results show that the dynamic-sized BF mechanism is able to notably reduce the redundant traffic incurred by the flow duplication attack in TOA based protocols.
2. PRELIMINARY

2.1. Bloom Filter based Multicast with TOA

A representative protocol in the TOA class is the free riding multicast protocol (FRM) [Ratnasamy 2006], which derives several variants [Jokela 2009, Särelä 2011]. We use Fig. 1(a) to illustrate working principles of TOA, where the left part provides a sample network topology and the right part elaborates the local states and processing details in node B of the network. Subscribers such as E and F send explicit registration or joining messages to the source node of a group. Path vectors from the data source to destinations are then calculated through scanning the border gateway protocol (BGP) routing information at the border router of the source domain [Rekhter 1995]. These path vectors are aggregated to form a multicasting tree, and the tree is then encoded as an in-packet BF, such as BF(l_2, l_5, l_6) in Fig. 1(a), where l_i denotes identifier of each link in the network.

The intermediate nodes are border routers of transit domains. Each intermediate node maintains identifiers of neighboring links as FRM forwarding states such as BF(l_1) and BF(l_2), which are also in the form a BF as shown in the figure. The intermediate node queries all the neighboring links against the in-packet BF during forwarding. A packet copy is generated and delivered over each matched interface, with the in-packet BF encoding the tree remains unchanged. As in Fig. 1(a), since BF(l_2) matches BF(l_2, l_5, l_6), a packet copy is delivered along the interface 2, still with in-packet BF BF(l_2, l_5, l_6).

2.2. Bloom Filter based Multicast with DOA

With DOA, subscribers also send explicit registration or joining messages to the source node of a group, where the difference is that such messages trigger each node along the joining path to install forwarding states to form a multicast tree within the network. Figure 1(b) illustrates working principles of DOA. The two subscribers send joining messages to the source and install along the joining path the forwarding states, which are the IP addresses of the subscribers encoded as BFs. In detail, DOA needs to deal with the asymmetric routing issue, since the path reverse to certain joining path may be invalid for the multicast forwarding for administrative reasons [Costa 2006]. In order to resolve the issue, a BGP-view based joining scheme was proposed [Tian 2014], where the basic idea is to make the receiver informed of the path vector perceived at the data source side. The joining message can then be delivered along the path reverse to the notified path using source routing.

In the forwarding stage, each DOA packet generated at the source node carries a BF that encodes all the destination IP addresses associated with a multicast group, such as IP_1, IP_2 and IP_5 shown in Fig. 1(b). The in-packet BF will be compared with all forwarding states on each interface of the node. The matched states on each interface will be combined to form a new in-packet BF, which is to be carried by the packet duplicate downstreaming along the interface such as IP_2 and IP_5 in the figure. The original 1-bit positions denoting unmatched element such as IP_1 in the figure will
be reset to 0s. Recalculating the forwarding BF for each multicasting branch is the unique feature of DOA, which is an underpinning factor leading to the robust security performance of DOA, to be studied in detail in Section 3 and 4.

2.3. Bloom Filter based Multicast in the DCN

While BF based forwarding was originally proposed for the inter-domain multicast over the Internet, it can also be tailored to serve DCNs [Tian 2013]. As a DCN has a more controllable environment and customized topology, multicast forwarding states and in-packet BFs can be computed and installed through a centralized entity, such as the multicast manager employed in the ESM [Li 2012]. The centralized controller can help compute an efficient tree structure and prevent the loop formation. In order to reduce traffic leakage incurred by BF based multicast in DCNs, a multi-class BF (MBF) scheme is proposed [Li 2011], which determines the number of hash functions used to encode an element into the BF based on the probability that the element is to be queried in the future.

Rothenberg et al present a DCN networking approach termed as switching with in-packet Bloom filters (SiBF) and develop an OpenFlow based testbed implementation [Rothenberg 2010]. With SiBF, the DCN is governed by a Rack Manager, which provides facilitating functions such as address resolution, route computation and topology discovery. Communication among DCN nodes under SiBF is established in a source routing manner, where a route is represented by an in-packet BF carried in the Ethernet MAC fields. SiBF provides a cost-efficient DCN operation and realizes false-positive-free forwarding and load balancing.

2.4. Security Challenges

2.4.1. Attack Models. Several attack models in the context of TOA have been identified, with the assumption that nodes on the edge of the network under study can be compromised but the ones in the core are secured. The attack models are summarized in the below.

— **Packet injection attack.** If an attacker has compromised one sender as a bot, then the confederate bots in other places can also inject traffic to the target by intelligently constructing a forwarding BF [Rothenberg 2009,Antikainen 2011].

— **Reverse-engineering attack.** The forwarding BFs along paths that share common links are collected and analyzed to derive the link identifiers (in the BF format), and the derived link identifiers can then be used for hostile purposes such as forming loops or incurring redundant traffic [Rothenberg 2009,Antikainen 2011].

— **Replay attack.** A malicious data source can capture some legitimate BFs from the network and reuse them to send traffic to corresponding destinations [Rothenberg 2009].

— **Resubscription attack.** The attacker can replay a join message captured previously to prevent a node from leaving a multicast group [Antikainen 2011].

— **Forwarding anomalies attack.** An attacker can forge a BF to trigger anomalies during the packet forwarding, which may incur the explosion of the traffic load in the network and in some subscribers [Särelä 2012].

However, the security performance of DOA protocol is basically an unexplored area. It is of great importance to study whether those attacks listed above are still launchable in DOA. If positive, what are the jeopardy consequences and how can the attacks be prevented? Moreover, all the available analysis on those attacks is performed under the Internet environment, and the impact of these attacks on the DCN are not analyzed either.

2.4.2. Resistance Solutions. In order to resist the DDoS attacks, the following mechanisms have been proposed.

— **Fill factor.** A forwarding node needs to check the BF’s fill factor $\rho$, defined as the proportion of 1 bits in the bit vector, before any further packet processing. The fill factor of an ordinary BF should
not exceed $\rho_{\text{max}}$ which is usually set as 0.5. It resists those attacks requiring adding more bits in the BF such as the packet injection attack and the forwarding anomalies attack [Sürelä 2012].

— **Bit permutation.** With bit permutation, each intermediate node re-maps the BF to a different arrangement. The multicast tree is constructed with the assistance of joining messages, where each joining message from a subscribing node carries the BF that encodes not only the link IDs but also the accumulated bit permutation along the path from the subscribing node to the group source. A multicast tree can then be created at the data source node by ORing all the permuted BFs collected from joining messages. During the forwarding, a falsely delivered packet can not be correctly de-mapped through the reverse bit permutation at each hop, so the packet without matched output interfaces will be dropped. Bit permutation is very useful in resisting attacks involving manipulating BFs such as the packet injection attack and the forwarding anomalies attack [Antikainen 2011, Sürelä 2012].

— **Flow specific Bloom filter.** Flow specific information such as the flow ID from the transport layer, and flow-dependent cryptography keys are used to calculate BFs. The flow specific BF is useful in resisting attacks involving manipulating the BFs, such as the reverse-engineering attack, replay attack, and resubscription attack [Antikainen 2011, Sürelä 2012].

Although provisioning resistance to DDoS attacks to some extent, the existing mechanisms are not able to eradicate those threats in the context of TOA [Antikainen 2011]. This study will answer some important questions. How will those existing security mechanisms perform in the context of DOA? How should we design the effective security schemes exploiting the unique feature of DOA? Will the DCN environment alleviate the security predicament or make it worse, when BF based forwarding is applied?

**3. PACKET INJECTION ATTACKS IN DOA**

**3.1. Basic Packet Injection Attack Model**

**Definition 1.** The basic packet injection attack is to leverage BF bit mismatch at the intersection of a normal path for packet delivery and a path between two bots, in order to direct unrequested data traffic from bots to a target. It is assumed that bots on the edge of the network are controlled by the attacker with full knowledge of the network topology.

We use Fig. 2(a) to illustrate the attack model, where the target $T$ has subscribed to a data flow generated from a compromised node $A_a$. The goal of the attacker is to enable another bot $A_b$ to send traffic to $T$ as well. To this end, the attacker first makes another bot $A_c$ to subscribe to $A_b$, but in the name of $T$. This could be done because $A_a$ should be willingly to share the IP address of $T$ with its accomplice and $A_c$ just needs to apply the IP address spoofing to impersonate $T$ [Duan 2008]. After the forwarding states are constructed, which are BFs encoding the IP address of $T$ and denoted as $B_T$, $A_b$ sends traffic to $A_c$ with the in-packet BF $B_T$. The intersection node $I$ will also duplicate the packet and forward the duplicate to $T$, according to the forwarding rule shown in Fig. 1. It is worth mentioning that $A_b$ and other bots certainly can send to $T$ with the help of $A_a$ as a relay; however, this requires $A_a$ to have a large capacity, or $A_a$ will collapse before $T$. This attack strategy in fact is equivalent to the straightforward flooding attack, which is not related to BF based forwarding and thus out of the scope of this paper.

The spoofed address can be detected with the reverse path forwarding (RPF) check. When a packet arrives at an interface of a router, the router examines its source IP address and see whether the interface is a part of the shortest path from the router itself to the claimed source IP address. If positive, the source IP address is genuine with high probability, otherwise, the IP address could be a spoofed one. Suppose each networking node performs such a check, the joining message sent by the spoofing receivers can be blocked, and the attack model illustrated in Fig. 2(a) will not be a threat. The RPF check however can be applicable only in the symmetric routing scenario, where the shortest path from node $A$ to $B$ is the same one that is from $B$ to $A$. Unfortunately, BF based
multicast was proposed for the scalable inter-domain multicasting, where the inter-domain routing is usually asymmetric for administrative reasons [Costa 2006].

### 3.2. Resistance with Bit Permutation

**DOA with Bit Permutation.** Applying certain cryptographic authentication techniques can prevent the IP spoofing, which will be discussed in Section 4.3. We here are interested in studying the security technique that does not rely on the availability of cryptographic authentication. Note that the forwarding path from $A_b$ to $T$ may not be a reverse legitimate joining path (e.g., in the asymmetric routing case). In order to prevent $A_b$ from injecting packets to $T$, it is helpful if we can make the forwarding process path-dependent, where the bit permutation technique can be utilized.

**DEFINITION 2.** Performing a bit permutation on a BF is to reorder the bits in the BF.

With bit permutation, the BF in the joining message is permutated and the result obtained is installed as the forwarding state at the corresponding interface. Specifically, the bit permutation operation at each hop is denoted as $H_i(\cdot)$. Suppose that the bot $A_c$ also subscribes to $A_b$ in the name of $T$. The first-hop node installs the forwarding state $H_1(B_T)$ at the interface leading to $A_c$, when the node receives the joining message. The node then attaches BF $H_1(B_T)$ to the joining message and forwards it to the next hop, as shown in Fig. 2(b). The data source finally receives the joining message from the last-hop, say, $n$th-hop node, and the message will carry the BF as $H_n \circ H_{n-1} \circ \cdots \circ H_1(B_T)$, where we use $H_n \circ H_{n-1}$ to denote the expression $H_n(H_{n-1}(\cdot))$ for better readability. The source node ORs all such a cumulatively permutated BFs to create a forwarding in-packet BF.

Nodes $T$ and $A_c$ both perform subscription in the name of $T$; however, the resulted BF at each of the source node is different from the other, since the initial BF $B_T$ experienced different permutations along different paths. In Fig. 2(b), BFs $B_a$ and $B_b$ are used to denote such two yielded BFs at source sides. It is worth noting that in the multicast forwarding stage, a forwarding node needs to apply the reverse permutation $H_i(\cdot)^{-1}$ to decode the accumulative permutation in a hop-by-hop manner.

In order to enable the joining process with bit permutation to handle the asymmetric routing environment, we could use a BGP-view based joining scheme. The source just notifies receivers the valid forwarding paths, and each receiver sends the joining message along the reverse path with source routing [Tian 2014]. Although designed for DOA, the BGP-view based scheme can also be applied to TOA bit permutation scheme, for which the technique for handling asymmetric routing has not been mentioned in the previous literature.

**Performance of the Bit Permutation and Insight.** We consider a general case for DOA with bit permutation as illustrated in Fig. 3(a). After DOA subscription process with bit permutation
Defending Against DDoS Attacks in Bloom Filter based Multicasting

Described above, packets with $B_a$ and $B_b$ are meant to be delivered from $A_a$ to $T$ and $A_b$ to $A_c$, respectively. According to the subscription procedure, the two packets with $B_a$ and $B_b$ can definitely reach $I$.

**Theorem 1.** If $B_a$ and $B_b$ are generated according to the random subscription from receivers, and if the number of 1-bit positions in $B_b$ is no less than that in $B_a$, a sufficient condition for the packet with $B_b$ at the intersection node $I$ to reach $T$ is that the positions set to 1 in $B_b$ are also set to 1 in $B_a$ after the permutation operations from the source node to $I$.

The theorem can be readily proved with contradiction: if the sufficient condition is not satisfied, some destinations encoded in the $B_a$ can not be reached according to DOA forwarding procedure (with permutation).

**Theorem 2.** The probability that the sufficient condition is satisfied is:

$$P_I = p_e \cdot \frac{z_b!}{(z_b - z_a)!} \cdot \frac{(m - z_a)!}{m!},$$

where $m$ denotes the size of the BF, $z_a$ and $z_b$ denote the number of 1-bit positions in $B_a$ and $B_b$ at $I$, respectively, and $p_e$ the probability that $z_b \geq z_a$.

**Proof:** For the first 1-bit position in $B_a$, the probability that the same position is also set to 1 in $B_b$ is $\frac{z_a}{m}$. After the first match, the probability that the second 1-bit position in $B_a$ is also set to 1 in $B_b$ is $\frac{z_a - 1}{m - 1}$. Along this vein, the probability for the last 1-bit position in $B_a$ to be set to 1 in $B_b$ is $\frac{z_a - (z_a - 1)}{m - (z_a - 1)}$. The probability that $B_b$ has more 1-bit positions than $B_a$ is $p_e$, which can be set as $\frac{1}{2}$ if the forwarding operations over the branches $A_a$ to $I$ and $A_b$ to $I$ are the same stochastically. The sufficient condition needs the positions being set to 1 in $B_a$ are also set to 1 in $B_b$, and $P_I$ shown in Equation (1) is then obtained as the product of the probabilities above.

**Corollary 1.** The probability $P_I$ achieves the upper bound when $z_b = m \cdot \rho_{max}$ and $z_a = k_1$, where $k_1$ is the least number of 1 bits for storing one element; $P_I$ achieves the lower bound when $z_a = m \cdot \rho_{max}$ and $z_b = z_a$.

The upper bound case occurs when $B_b$ has as many as possible positions set to 1 and $B_a$ has as few as possible positions set to 1, where the injecting packet is most likely to match $B_a$. Similarly, the lower bound case occurs when $B_a$ has as many as possible positions set to 1 and $B_b$ has as few as possible positions set to 1, where the injecting packet is most unlikely to match $B_a$.

Corollary 1 reveals that the ability of DOA to defend against the packet injection attack depends on configuration of some key parameters of the BF: the values of $m$, $\rho_{max}$ and $k_1$ determine how
likely the injection attack can succeed according to Corollary 1. In practice, \( k_1 \) can be set equal to the number of hash functions \( k \). Although it is possible that \( k_1 < k \) since different hash functions may map an element into the same bit position in practice; however, the probability of such event is very low when \( m \) is reasonably large. Equation (1) provides a guideline how to configure BF parameters with security considerations. Figure 4 gives some examples about how to determine BF parameters. Figure 4 (a) shows how to choose the number of hash functions \( k \) with different fill factor thresholds \( \rho \text{max} \), considering different levels of probability that the injection attack can occur \( P_I \). Figure 4 (b) shows how to set \( \rho \text{max} \) with different \( k \) in different levels of \( P_I \).

We could extend the scenario shown in Fig. 3 to more general cases. If there are multiple intersections, the probability \( P_I \) is still applicable to any one of them. In this case, the probability that a packet is falsely delivered to a target is the probability that the packet is mismatched at any of the intersections, i.e., \( 1 - (1 - P_I)^{C_I} \), where \( C_I \) is the number of intersections between the paths \( A_a - T \) and \( A_b - A_c \). Note that even without the on-purpose injection attack, it is still possible that a packet is forwarded to an unrelated receiver due to BF false positive matching over an interface, which has been well studied in the literature [Särälä 2011] [Antikainen 2011].

**Against Bit Permutation Derivation.** The study on TOA security performance [Antikainen 2014, Antikainen 2011] reveals that the injecting bot \( A_b \) in fact has more effective methods than the pure random mismatching to launch injection attack, referring to the following proposition.

**Proposition 1.** \( A_b \) only needs to produce an appropriate BF satisfying the following two conditions in order to reach \( T \):

\[
\begin{align*}
B_x \supseteq B_b, \\
H_b^{-1}(B_x) \supseteq H_a^{-1}(B_a),
\end{align*}
\]

where one BF is a subset of the other if all 1-bit positions in the first one are also set in the second, and \( H_a^{-1} \) and \( H_b^{-1} \) denote the reverse of the cumulative bit permutations \( H_a \) and \( H_b \), respectively.

We can understand this proposition using the example as shown in Fig. 3(b). The first sub-condition shown in condition (2) means that \( B_x \) must contain all 1-bits set in \( B_b \), thus the packet carrying \( B_x \) can reach the intersection \( I \). The second sub-condition shown in condition (2) means that after the reverse bit permutation from \( A_b \) to \( I \), \( B_x \) contains all the 1 bits that will be in \( B_a \) after the reverse cumulative permutation from \( A_a \) to \( I \), and thus can further reach \( T \) passing through the path the \( B_a \) would pass.

However, calculating such a \( B_x \) requires the information that how the permutation is performed along certain path segments. Specifically, the attacker needs to know \( H_b \bullet H_c \) and \( (H_a \bullet H_c)^{-1} \) as shown in the Fig. 3(b) before calculating \( B_x \) [Fahmy 2013, Antikainen 2011]. The attacker can use a trial-and-observe method in order to derive such permutation information in TOA. Specifically, the attacker can set an original 0-bit to 1 in the in-packet BF at the sending bot, and observe the newly
emerged 1 bit position in the BF at the receiving bot [Fahmy 2013, Antikainen 2011]. The method works in TOA because the 1-bits of the in-packet BF, though experiencing permutations, will never be eliminated during the forwarding. Furthermore, some security mechanism periodically updates link IDs in the network, and every bit position of an in-packet BF has some chance to be set to 0, which unexpectedly can be leveraged to measure the permutation pattern [Fahmy 2013, Antikainen 2011].

The method mentioned above can not be fully effective in DOA, as a BF is recalculated at each hop in the forwarding process, and the added 1 bits may be trimmed off. To fully derive the permutation information in DOA, the attacker has to ask the sending bot to subscribe to the receiving bot using BFs with 1 bits incrementally added in different positions. The receiving bot then let the sending bot know the final rearrangement. Take the situation in Fig. 5(b) for example, $H_a \bullet H_c$ and $(H_b \bullet H_c)^{-1}$ can be obtained after several rounds of $A_b$ to $A_b$ and $A_a$ to $A_c$ measurements.

For such a potential threat to DOA, the culprit of the information leakage is the abuse of the subscription. In order to prevent such abuse, we can use an address fixation scheme, which requires each subscriber to use only one IP address to do the subscription. The address fixation could be conveniently implemented by asking the first hop router to check if the subscriber is using the same BF (BF value of the IP address of the subscribing stub domain) when performing subscription. In this case, $A_b$ is unable to measure the permutation and thus can not construct $B_x$ according to proposition 1. With address fixation applied, packet injection in DOA can only be launched through random BF mismatching; therefore, the capability to defend against such an attack depends on how to design the BF as analyzed in Equation (1).

3.3. Colluded Packet Injection Attack Model

We now show that $B_x$ can be forged with legitimate subscriptions from multiple colluded bots, even if the address fixation scheme mentioned above or a trusted third party exists in the system.

**Definition 3.** The **colluded packet injection attack** is to create an appropriate $B_x$ at a bot using legitimate subscriptions from a set of colluded bots summoned by the attacker, so that the bot with $B_x$ can perform injection attack to a target.

The colluded packet injection attack can occur in both TOA and DOA, which was not identified before to the best of our knowledge. In TOA, as the attacker has known the objective $B_x$, and corresponding bit permutation can be derived, it can ask more bots to legitimately subscribe to $A_b$ to make $B_x$ available, as shown in Fig. 5(a). Even the trusted third party can not detect such an attack, since bots are performing subscription legitimately and the occurrence of $B_x$ becomes an artificial coincidence.
In DOA, the bit elimination process is actually playing the role of the trusted third party, since many illegitimate 1-bits forged by $A_b$ will be eliminated in the procedure of recalculating the forwarding BF. Furthermore, making $B_x$ to satisfy the condition (2) is not enough in DOA, the attacker must also install corresponding forwarding states along the path from $A_b$ to $I$, in order that the forwarding BF having survived the bit elimination can finally reach $I$.

To this end, the attacker must summon those bots whose shortest paths to $A_c$ share the path segment at least from $A_b$ to $I$. Although this takes efforts, it is still possible to find these bots. Moreover, the position of $A_b$ is critical in the attack described above. If $A_b$ is only one hop from the intersection, the bit elimination does not have any chance to remove redundant bits in $B_x$, which is still a potential threat. Figure 5(b) presents how many bots are available for waging such attack in two topologies with different extents of connectivity [BGP Analysis Reports]. Due to the randomized assignment of bots, the number of possible colluding bots may vary, but it is noted that the number is larger when the connectivity is smaller. This is because the connectivity can be regarded as profiling the extent of link sharing among nodes. The lower connectivity means that each link is shared by more nodes, thus more possible colluding nodes can be found.

3.4. Source-Driven Tree Construction with Reverse Bit Permutation

We propose a source-driven tree construction scheme with reverse bit permutation to resist the colluded packet injection attack. A subscriber sends a joining message towards data source when it subscribes to a flow. Different from the regular subscription process mentioned in Section 2.2, the intermediate networking nodes do not install forwarding states when the joining message passes by. When the joining message arrives at the data source, the source sends a tree construction message towards the subscriber and at the same time the forwarding states are installed at those nodes along the path. We use Fig. 6(a) to illustrate the tree construction process and the detailed process is shown in Algorithm 1.

![Fig. 6: Source-driven tree construction and reverse bit permutation.](image)

**Algorithm 1:** Tree construction

**Input:**
- Permutations of the node;
- Dest. IP addr. of the receiver $DstIP$;
- In-packet BF $iBF$ initially equals $B_T$ encoding $DstIP$;

Let $l$ = Output interface for $DstIP$;
Let $H = $ Permutations[$l$];
set $iBF$ in packet to $H(iBF)$;
if $H(iBF)$ at $l$ as a state NOT found then
  Install $H(iBF)$ at $l$ as a state;
end

Forward the packet via $l$;
The state installed at each interface is also the BF formatted IP address of the receiver. We incorporate the bit permutation in the process. The tree construction message will carry the BF formatted state, permute the state at each hop, and then install the state at corresponding interface. As shown in Fig. 6(a), we use $B_T$ to denote the BF formatted IP address of the receiver $T$. Note that the permutation for each output interface of the node is different and the permutations of the in-packet BF along the forwarding path are cumulative. For example, if we construct a tree branch from $A_n$ to $T$, the in-packet BF received by $T$ is $iBF_m = P_m(B_T)$, where $P_m = H_1 \cdot ... H_{m-2} \cdot H_{m-1} \cdot H_m(\cdot)$ is the cumulative permutations along the path. The state installed at the corresponding interface is just as shown in Fig. 6(b).

With the multicast tree obtained from the source-driven tree construction process, malicious injectors can be located through the reverse bit permutation, even if the colluded injection attack occurs. Specifically, if $A_b$ in Fig. 6(a) is able to manipulate an appropriate BF $B_x$ through collusion and inject traffic from $A_b$ towards $I$, and then towards $T$, node $T$ will be aware of the injection attack when it receives a BF (from $A_b$) other than the normal one $iBF_m = P_m(B_T)$. Suppose $T$ receives an injected packet from $A_b$ with the in-packet BF in the form of $iBF_n = P_n(B_T)$, where $P_n = H_1 \cdot ... H_{n-2} \cdot H_{n-1} \cdot H_n(\cdot)$, receiver $T$ can delete the payload and send the rest of the packet including the in-packet BF back to its father node, which performs the reverse bit permutation $H^{-1}(iBF_n)$ on each interface.

Algorithm 2 shows details of the bit permutation process at each node and Fig. 6(b) illustrates an example of reverse bit permutation at node $I$. In Fig. 6(b), node $I$ finds that $H^{-1}(iBF_n)$ matches the state at interface 2, i.e., the 1-bit positions in the forwarding state are also set in the $iBF_n$. Then the packet is forwarded along the horizontal path. Since the tree construction process actually labels the path, the reverse path forwarding can definitely locate the source nodes that owns the capability of delivering packets to $T$. The node is $A_b$ in the example shown in Fig. 6.

**Algorithm 2:** Reverse bit permutation

```
Input: Permutations of the node;
       Links of the node;
       In-packet BF iBF;

foreach outgoing link l do
    Let $H^{-1} = \text{ReversePermutations}[l]$;
    if $H^{-1}(iBF)$ matches a state at l then
        $iBF = H^{-1}(iBF)$;
        Forward the packet via interface l;
    end
end
```

When the upgoing message arrives at the first-hop router on the sender side, the first-hop router is able to identify the malicious injector by identifying the link with the matched state. As the probability of false forwarding due to random BF mismatch is extremely low under bit permutation, any traced source node other than the subscribed one is a malicious injector with high probability. It is worth noting that if bot $A_b$ attempts to inject packets to $T$, it simply constructs a tree branch from $A_b$ to $T$, which is equivalent to the straightforward flooding attack. The flooding attacker can also be located with the reverse bit permutation scheme.

The scheme above resolves an important issue in defending against the packet injection attack: the location of the bot is difficult to find. Once the malicious source is identified, it is comparatively easy to block the injected traffic through security configuration at the first-hop router. When the trace-back message arrives at $J$, it is possible that the message will be forwarded to the genuine data source node; nevertheless, tracing to the true source will not interfere localizing the injector.
4. SECURITY ROBUSTNESS WITH DOA

This section presents analysis to show that DOA is robust under some attacks that are severe threats to TOA. Without specific indication, it is assumed that bots on the edge of the network are controlled by the attacker with full knowledge of the network topology.

4.1. DOA: Inherent Immune to Reverse-Engineering Attack

**Definition 4.** The reverse engineering attack is to infer the content of the in-packet BF transmitted over the network, so that the attacker can create loops by producing a particular BF.

The reverse-engineering attack can be easily conducted in the context of TOA. The attacker simply obtains the BF formatted paths between all pairs of bots, and then conjectures the possible common path segments according to the topology. For each single link along the common path segments, the attacker bitwise “ANDs” all the path BFs containing that link, which yields the BF of the link ID with considerable accuracy [Antikainen 2011]. The derived BFs for certain links that form a loop can be bitwise “ORed” to create a new BF, with which the sender can induce a traffic loop within the network. The root cause of the reverse-engineering attack in TOA is that TOA encodes into a BF the entire tree information from source to destinations, and the amount of 1-bits in the BF never decreases. Experiments showed that an attacker with thousands of bots could derive the BF values of a significant portion of all the link IDs in a topology with tens of thousands of nodes [Antikainen 2011].

DOA is inherently immune to the reverse-engineering attack. The routing information encoded in BF is the destination IP addresses. Since each IP address is unique, it is impossible for the attacker to retrieve more information through ANDing over multiple BFs. The reverse-engineering attack is meaningless in DOA, since the multicast forwarding still needs the facilitation of forwarding states installed by joining messages.

4.2. Resisting Resubscription Attack

**Definition 5.** The resubscription attack is to prevent a subscriber from unsubscribing to a data source.

Each TOA packet carries the entire tree encoded into a BF. The number of 1-bits in the BF will decrease if some subscribers leave the group. Observing the decrement, the bot in the same group can subscribe to the source using the original BF, so that the sender side can continue sending packets to subscribers intending to leave [Antikainen 2011].

In the context of DOA, the bot is unable to subscribe for others even without the authentication. BF recalculation in the forwarding process makes a receiver only accessible to the BF representing the receiver’s own IP address. Moreover, even if the bot can get the BF representing the target that intends to leave, the BF encoding the destination IP addresses itself is insufficient to enable delivering traffic to the target. With DOA, the forwarding states (associated with a destination) installed at related routers are also required for routing traffic. When a subscriber decides to leave the group, it will send un-subscribing messages to remove those forwarding states. With the address fixation scheme applied, it is impossible for a bot to reinstall the states for a resubscription attack.

As of now, we find that DOA provides a higher security level compared to TOA under the same network setting:

- BF bit permutation can be measured and thus exploited for traffic injection attack under TOA [23]. Under DOA, a simple yet effective address fixation mechanism can be applied to disable such permutation measurement and thus defend against the injection attack.
- DOA is immune to the reverse-engineering attack while TOA is not able to prevent the information leakage even with the strongly favored assumption that the network topology is unknown [Antikainen 2011].
- DOA is immune to the resubscription attack while TOA is not.
4.3. Effectiveness of Authentication Center

In order to provide a secured multicast service, there could be a trusted third party such as the authentication center (AC), which is responsible for billing and other operations such as BF computation [Xylomenos 2013]. A natural question to ask is: to what degree could the AC bring security to multicast protocols?

4.3.1. Defending Against Co-Residence Replay Attack

**Definition 6.** The co-residence replay attack enables a bot to send unrequested data traffic to certain target, which leverages legitimate BFs generated by other bots located in the same edge domain.

The co-residence replay attack can be identified if we look into the implementation details of BF based multicast protocols, which are designed for inter-domain multicasting in the first place. The networking node shown in previous figures in fact represents the border router of a domain. Take $A_2$ for example, it acts as the border router of an autonomous system (AS) domain as shown in Fig. 7. If the data source host is responsible for constructing the in-packet BFs, the attacker has no need to compromise the router. The bot which has been legitimately subscribed by the target can just send the legitimate BF to other bots within the same domain, as shown in Fig. 7. The other bots could just send the spam packets with the “replayed” BF to the border router, and these spam packets are then delivered along the same tree structure to the subscriber’s domain. Within the receiver domain, the border router discriminates the subscribing hosts by their IDs, which can also have been “replayed” at the source domain; therefore, the spam packets will finally reach the target host.

The flow specific BF computation is insufficient to resolve the issue, since the content in the legitimate packet can be replayed by confederate bots, including the flow ID from the transport layer [Antikainen 2011]. The bit permutation is insufficient to resolve the issue either: it can not defend against the replay attack from the same domain. In order to defend against the co-residence replay attack, the AC can be utilized as the trusted third party to separate the data source from BF computation and insertion, which helps both DOA and TOA based protocols.

4.3.2. Extra Benefits of AC with DOA. The AC could be equally effective in defending the replay attack for both DOA and TOA based protocols; however, the AC can provide extra benefits with DOA in defending against certain other attacks, compared to TOA and the traditional IP multicast.

Recall the colluded packet injection attack studied in Section 3.3. A separate AC can effectively defend against the colluded subscription attack. This is because the AC is able to prevent subscribers from using multiple IDs or different attack-facilitating IDs to subscribe to a bot source node such as $A_6$, each time. No matter the subscribing ID is a genuine IP address of the subscriber or not, the ID
can be authenticated only once. Thus the colluded bots need more efforts and luck to help the bot source create an appropriate BF $B_x$ in DOA.

In contrast, the AC is unable to prevent the bot source to forge a $B_x$ in TOA. This is because the AC is unable to determine if the BF sent from $A_b$ will incur injection or not. The authentication functionality can only verify the one is as claimed or not. DOA needs subscribers’ IP addresses and on-path states to construct in-packet BFs and perform in-network forwarding, that is, the forwarding is closely related to the ID of subscribers, thus the AC can improve the security level of DOA. However, as the bot source can make in-packet BF on its own in TOA based protocols, the AC can be of little help.

Moreover, we find that AC is more effective in defending against a silent subscription attack, which also happens in a traditional IP layer multicast protocol, the source-specific multicast (SSM) protocol [Rekhter 2006]. The silent subscription attack is to install as many forwarding states as possible in the network. Consider a situation of source-driven construction, where bots just subscribe to other bots and there is no AC. The bots to be subscribed just construct the forwarding states along forwarding paths, but do not inject any packets. We could not locate the malicious bots, as we are able to do that only if injected packets are available; however, the computation and memory resources in the intermediate routers will be exhausted, because there is no AC and bots could crazily subscribe to other bots in different names.

In the SSM, subscribers can also install many group-specific forwarding states in the network, where the defending strategy is to limit the number or the rate of tree construction messages can be initiated from certain interface or source node [Rekhter 2006]. The reason SSM proposes such a design is that although a node is authenticated as it claims to be, it also can crazily subscribe to the data source with group IDs that are independent with its own ID. In contrast to SSM, the subscriber’s ID and the states are the same in DOA based multicast; therefore, the AC is more effective in DOA compared to SSM.

5. SECURITY ANALYSIS ON TOA UNDER DCN

The forwarding anomalies attack triggers forwarding anomalies such as loop and flow duplication on purpose to consume network resources, where the success ratio of the attack is highly dependent on the topology that the multicast tree is constructed on. In contrast to the Internet environment where the topology is unpredictable, the DCN has a managed topology. The special architecture of the DCN, unfortunately, provides convenience for the attacker to launch the forwarding anomalies attack against BF based multicast protocols designed for the DCN itself such as ESM [Li 2012], which is in the TOA class.

5.1. Managed Topology under DCN

There are basically two kinds of topologies for the DCN: the switch-centric such as Fat-Tree and the server-centric such as BCube [Li 2012, Chen 2004]. A well-known characteristic of the two architectures is that there must be multiple paths between any two servers. This is to avoid the bandwidth bottleneck and point of failure. While DCNs have different interconnection structures, they utilize several levels of switches for server interconnection, and switches within the same level are not directly connected, thus can be modeled as multistage graphs [Li 2012]. Due to the limitation of the space, we will analyze the Fat-Tree architecture as an example of the DCN.

The multi-path configuration of the DCN especially assists the loop formation, such as loops $w_8 \rightarrow w_{16} \rightarrow w_{10} \rightarrow w_{17} \rightarrow w_8$ and $w_9 \rightarrow w_{18} \rightarrow w_{11} \rightarrow w_{19} \rightarrow w_9$ shown in Fig. 8, which is the multistage graph expression of the paths from server $v_0$ to $v_5$ in a Fat-Tree DCN constructed with 4-port switches [Li 2012]. A bot knows BFs for multiple paths from itself to its confederate bots, since the routing scheme in the DCN exploits the path diversity for traffic engineering purposes [20]; therefore, it only needs to OR different path BFs in order to create loops in the switched network. In order to defend against such an attack, any node $j$ is instructed to forward the packet to a neighboring node only if the neighbor has a longer distance to the packet source than $j$ itself, where it is assumed that each switch knows the entire topology information [Li 2012].
5.2. Forwarding Anomalies Attack in the DCN

**Definition** 7. The forwarding anomalies attack triggers forwarding anomalies such as loop and redundant traffic to consume networking resources, where the legitimate in-packet BF could be tampered.

The distance comparison scheme mentioned above is unable to defend against one form of the forwarding anomalies attack, the flow duplication attack [Sürelä 2012]. We still consider the example illustrated in [Li 2012]. We here examine the paths from v₀ to v₅ for the convenience of demonstration. Note that only one path is needed from v₀ to v₅, but the network provides 4 different paths as highlighted. If the malicious host v₀ tampers with a BF from the centralized controller by encoding more tree branches, v₅ could suffer the packet flood. For example, if the BF is configured to deliver the packet along both paths w₈ → w₁₀ and w₈ → w₁₀ → w₁₀₀, then there will be 2 packets delivered along the path w₁₀ → w₂ → v₅. Similarly, if the other two paths are also encoded, there will be other 2 packets along path w₁₁ → w₂ → v₅. Consequently, there will be 4 times the traffic as required delivered along the path segment w₂ → v₅. These duplicated traffic can not be blocked with the distance comparison rule. For example, w₁₆ and w₁₇ are all farther to v₀ then w₈ is from v₀, thus w₈ will definitely forward the packets to w₁₆ and w₁₇, as long as the in-packet BF has encoded corresponding branches.

The routing scheme in the DCN also facilitates the flow duplication attack. The DCN uses different paths for load balancing and congestion avoidance, which leaks the information about BF formatted paths. All the malicious source needs to do is to combine these BFs. Note that the path pairs in the upper and lower part of the Fig. 8 the difference between each path pair is slight. It is very possible the attacker adds a number of 1 bits to BF, and meanwhile keeps the fill factor below ρmax. We now examine the maximum elements could be added to BF while remaining the fill factor below ρmax.

**Lemma 1.** Given an m-bit, k-hash-function BF with y elements encoded, the probability that a bit position is set to be 1 in the BF is

\[ p_y = 1 - \left(1 - \frac{1}{m}\right)^{y \cdot k}, \quad y \cdot k \leq m \cdot \rho_{max}. \]  

(3)

This is because the probability a bit position is set to be 1 in the BF is \( \frac{1}{m} \); after encoding y element with each element hashed k times, the probability that a bit position is still not set to be 1 is \( (1 - \frac{1}{m})^{y \cdot k} \). Considering different hash functions may map different elements into the same bit position, \( y \cdot k \leq m \cdot \rho_{max} \). Consequently, if we encode another \( y_1 \) elements into the BF, noting that the original 1-bit positions are still possible to be set to 1 again, the probability that a bit position is
LEMMA 2. Given an m-bit, k-hash-function BF with y elements encoded, the probability that there are totally $m_1$ bit positions set to be 1 is

$$P(m_1 = i) = \binom{m}{i} \cdot p_y^i \cdot (1 - p_y)^{m-i}, \quad 1 \leq i \leq y \cdot k. \quad (5)$$

THEOREM 3. Given an m-bit, k-hash-function BF with y elements encoded, we use a random variable X to denote the number of new 1-bit positions can be obtained if encoding another $y_1$ elements into the BF, then

$$P(X = x) = \sum_{i=1}^{y \cdot k} P(X = x | m_1 = i) \cdot P(m_1 = i), \quad (6)$$

where

$$P(X = x | m_1 = i) = \binom{m-m_1}{x} \cdot p_{y_1}^x \cdot (1 - p_{y_1})^{m_1-x}, \quad 0 \leq x \leq y_1 \cdot k.$$

We have the expression of $P(X = x | m_1 = i)$ because the newly set 1 bit positions must be obtained by setting the 0 bit positions before new elements are encoded.

Numerical analysis on Equation (6) is illustrated in Fig. 9. For each curve, we fix the values of m and k, and encode as many elements as possible to make $y \cdot k = m \cdot \rho_{\text{max}}$. As some hash functions will map the same element into the same bit position, the fill factor in fact cannot achieve $\rho_{\text{max}} = 0.5$. Figure 9 illustrates the cumulative distribution function (CDF) of $X$. For example, when $m = 800$, $y = 80$ and $k = 5$, we find that if adding $y_1 = 36$ more elements, the newly emerged 1 bit positions will be up to 120, and the entire fill factor after adding the new elements is still below $\rho_{\text{max}}$. This means that the attacker has a high probability to encode a number of new paths into BF and meanwhile keep the fill factor below $\rho_{\text{max}}$, which facilitates the flow duplication attack.

5.3. Resisting Flow Anomalies with Dynamic-Sized BFs

The loophole in BF based multicast for the forwarding anomalies attacker is that the fill factor of the in-packet BF could be far below $\rho_{\text{max}}$. That is, even if extra 1-bits are inserted into BF, the proportion of 1-bits could be still below $\rho_{\text{max}}$. The root cause of the phenomenon is that the size of BF is fixed, but the number of elements to be encoded into the BF may not perfectly fit the size all the time.
To fully understand this, we need to review the BF encoding process. In TOA service model, data source knows subscribers and thus can compute corresponding tree branches. The number of tree branches could be more than the capacity of the BF; therefore, multiple BFs may need to be created, with each encoding a subset of tree branches that could reach a subset of destinations. However, it is difficult to guarantee that each of such BFs contains the number of elements, which just load BF with fill factors close to $\rho_{max}$. For example, if a BF can contain 4 tree branches and the source now has 10 branches to encode, the source will need to create 3 BFs at least, and there must be a BF that contains only 2 branches, which makes it very possible for the BF to be inserted into more 1-bits to incur flow duplication because the original fill factor is very low.

We propose a dynamic-sized BF to encode the elements such as tree branches, where the size of BF suits the number of elements to be encoded. If we have a very sparse BF, we could shrink its size to improve the fill factor and reduce the possibility the malicious one could insert other 1-bits. Although the previous study on BF mentioned halving BF [Broder 2004], the design details and corresponding analysis on its properties are not provided. Guo et al proposed a dynamic BF for database system [Guo 2012], but the complexity and memory cost are not suitable for BF based multicast. Rizvi et al propose a hierarchical tree splitting mechanism, which splits a large multicast tree covering a large number of receivers into multiple smaller multicast trees [Rizvi 2011, Rizvi 2012]. The scheme strikes a balance between the number of receivers and the bandwidth efficiency. However, the purpose of the tree splitting is to make the common links among those smaller trees as few as possible with the size of the BF is static, while the dynamic-sized BF scheme is to shrink the sparse BF and reduce the possibility that the malicious one could insert extra 1 bits.

The basic idea of the proposed dynamic-sized BF is to compute a smaller-sized BF based on the full-sized BF. The BF is initially an empty bit vector with all positions set to 0. To denote a set with a BF, each element in the set is hashed $k$ times and each hash will map the element into a position in the bit vector and set the original 0 to 1. A careful examination of the hash operation reveals that the element is actually mapped into an integer first, and the integer is then mapped into a bit position through modulo operation [30]. Given that an element is mapped to $L$, and the size of the BF is $m$, the position yielded from the entire hash operation is $L \mod m$. If we need to insert the element into a BF with size $m/2$, we should find the corresponding position through $(L \mod m) \mod \frac{m}{2}$.

**Claim 1.** If $L' = L \mod m$, then $L' \mod \frac{m}{n'} = L' \mod n'$, where $m = r \cdot n'$ with $r = 1, 2, 3...m$.

**Proof:** By the definition of modulo operation,

$$
L' = L \mod m = L - (L \div m) \cdot m,
$$

where $(L \div m)$ denotes the quotient of the Euclidean division of $L$ by $m$. Thus,

$$
L' \mod \frac{m}{n'} = (L' \div \frac{m}{n'}) \cdot \frac{m}{n'} = (L \div m) \cdot m - (L \div m) \cdot n' = L - (L \div m) \cdot m - (L \div m) \cdot n' = L - (L \div m) \cdot \frac{m}{n'}.
$$

\[\blacksquare\]
5.4. Determining Parameters of the in-packet BF

Although Claim 1 provides the theoretical foundation of the dynamic-sized BF, it is non-trivial to determine the size of the in-packet BF. The size of the BF \( m \) could influence the false positive rate of the BF, which is another important factor for designing the BF significantly impacting the probability of forwarding anomalies and the performance of BF based forwarding [Säreliä 2011, Säreliä 2012, Tian 2012]. Unfortunately, the desire of a comparatively high fill factor and a low false positive rate are naturally in dilemma. Figure 10 gives an illustration that the false positive rate of BF could increase with the corresponding fill factor by several orders of magnitude. This is because a higher fill factor incurs that more elements are mapped into the same position of the bit vector, which is the root cause of false positive in BF. Consequently, increasing the fill factor by squeezing BF is able to reduce the possibility of packet duplication attack on one hand, but increase the possibility of forwarding anomalies on the other.

We here investigate the possible solution to design parameters of BF, in order to not only make the fill factor close to \( \rho_{\text{max}} \) but also make the false positive rate of BF below certain threshold.

**Theorem 4.** In order to not only make the fill factor close to \( \rho_{\text{max}} \) but also make the false positive rate of a BF below \( p_{\text{max}} \), we could roughly set

\[
k \approx \left\lceil \frac{1}{\log_{p_{\text{max}}} \rho_{\text{max}}} \right\rceil,
\]

(7)

\[
m \approx \frac{1}{1 - (1 - \rho_{\text{max}})^{\frac{1}{kn}}}.
\]

(8)

**Proof:** Given an \( m \)-bit BF containing \( n \) elements, the probability any bit position is set to 1 is

\[
1 - (1 - \frac{1}{m})^{nk},
\]

where \( k \) is the number of hash functions used to generate the BF. The number of 1-bit positions in the BF can be roughly calculated as

\[
n_{1} \approx m \cdot (1 - (1 - \frac{1}{m})^{nk}),
\]

(9)

thus the expected value of the fill factor is

\[
\rho \approx \frac{n_{1}}{m} = 1 - (1 - \frac{1}{m})^{nk}.
\]

(10)

Since the false positive rate of BF is known as

\[
p = (1 - (1 - \frac{1}{m})^{nk})^{k},
\]

(11)

With approximate equations (8) and (9), we have

\[
\rho \approx p^{\frac{1}{k}}.
\]

(12)
Based on $\rho_{\text{max}}$ and $p_{\text{max}}$, which are known as design objectives. According to approximate equation (8), we can choose a suitable value for $k$ as shown in approximate equation (8) and then determine $m$ as shown in approximate equation (9).

As we can only resize the BF with the factor of $\frac{1}{n}$, we could choose the actual size of the BF closest to the one computed according to approximate equation (9). Although the analysis only gives an approximative solution, it provides a guidance on how the in-packet BF could be designed. The question is, if we must maintain the false positive rate threshold, to what extent the scheme can help making the in-packet BF approaching $\rho_{\text{max}}$, and to what extent the flow duplication attack can be resisted. We will provide simulation results in Section 6.3, which indicates that the dynamic-sized BF scheme could help resisting, but is unable to completely prevent the attack. The fundamental reason is that TOA based protocols have complete routing information in BF, which easily invites attacks.

5.5. Discussions

We note that it is assumed that the DCN adopts the centralized controller for multicasting, such as the multicast manager in the ESM [Li 2012]. The switches in the DCN are usually low-end, thus it requires further study if it is practical to assume the switch can learn entire topology information to avoid the forwarding that can form loops. Switches can certainly ask the controller for the topology information, but a careful design to ensure the efficient query deserves investigation. An implicit advantage of the low-end switch devices is that they are unlikely to be installed some hostile software thus compromised; therefore, it could be safer if the controller encodes the computed tree and sends the packet to the first-hop switch directly in the Fat-Tree DCN.

6. SIMULATION RESULTS

6.1. Packet Injection Attack under the Internet

We validate our analysis through simulating with the Internet topology containing about 4000 nodes, which is created using the practical Internet statistics [BGP Analysis Reports 2013]. We evaluate performance of bit permutation in defending the packet injection attack in the context of DOA. In the simulation, we randomly pick up a path from $A_a$ to $T$, as the example shown in Fig. 3 (a). For the given path, we search all the bots whose shortest paths to $A_a$ share the path segments at least from $A_a$ to the intersection $I$. This is to enable $B_b$ to encode as many as possible elements to increase the probability that $B_b$ can match $B_a$ at the intersection $I$. For every such setup, we first only ask $T$ to subscribe to $A_a$ so that $B_a$ encodes only one element. We then change the order of bit positions in $B_b$ and try to match $B_a$. The matching probability can be obtained by observing how many trials needed before the two BF match for the first time. This probability is $P_I$ yielded from the simulation, which is compared with the theoretical result under the same setting computed from Equation (1). After examining the situation $B_a$ encodes only one element $T$, we then ask two subscribers to subscribe to $A_A$, so that we can evaluate what if $B_a$ contains 2 elements.

We perform such experiments multiple times in two network environments, where there are 15% and 26% of nodes acting as bots, respectively. The simulation results are illustrated in Table 1, where $\text{Ite.}$ in the table indexes the round of the experiments, and $B_b$ column means the number of elements encoded in $B_b$, $\text{Theor. } P_I$ means the theoretical probability of a successful injection and $\text{Sim. } P_I$ means the successful injection probability obtained from the simulation. The results show that the the simulation and the theoretical results are in the same order of magnitude, which indicates that our theoretical analysis is reasonable. For the scenario where there are 15% of nodes acting as bots, it is more difficult to successfully inject the packet than in the 26% scenario. This is because $B_b$ has less 1 bit positions in the 15% case. We also observe that if $B_a$ contains two elements, the success ratio of injection attack is lower than when it contains only one. This is because it demands more for $B_b$ to match it. The simulation results comply with our theoretical analysis.

The results indicate that the bit permutation is highly effective in defending the packet injection attack under DOA. BF $B_a$ encodes 1 more element, the success ratio of the injection attack could...
decrease 2 to 3 in the order of magnitude. BF $B_a$ will encode many elements rather than 1 or 2 in practice, the success ratio of injection could be ignorable small.

Table I: Performance of Bit Permutation in Resisting Injection

| Iteration | $B_a$ with 1 ele. ($\times 10^{-4}$) | $B_a$ with 2 ele. ($\times 10^{-4}$) |
|-----------|-----------------------------------|-----------------------------------|
|           | Theor. $P_I$ Sim. $P_I$           | Theor. $P_I$ Sim. $P_I$           |
| 1         | 39                                | 3.73                             |
| 2         | 37                                | 2.61                             |
| 3         | 37                                | 2.76                             |
| 4         | 41                                | 5.20                             |
| 5         | 35                                | 2.39                             |

| Percentage of bots: 26% |
|------------------------|
| Iteration | $B_a$ with 1 ele. ($\times 10^{-3}$) | $B_a$ with 2 ele. ($\times 10^{-3}$) |
| 1         | 64                                | 2.06                             |
| 2         | 66                                | 2.39                             |
| 3         | 65                                | 2.18                             |
| 4         | 68                                | 2.85                             |
| 5         | 68                                | 2.85                             |

6.2. Flow Duplication Attack in the DCN

We now examine how likely the flow duplication attack can happen in the DCN. First, we want to find out the total number of paths that allow the flow duplication to occur in the Fat-Tree DCN. Recall the scenario illustrated in Fig. 8 where the duplication can happen if there exists multi-paths forming the shape of a rhombus. Since a typical Fat-Tree has only 3 levels, the maximum distance between two servers is 6 hops. Consequently, two kinds of rhombus can be observed as highlighted in Fig. 8 where one kind contains 4 edges and the other contains 8 edges. Figure 11 (a) shows the total number of rhombuses can be found under different amounts of servers in the DCN. We can see there are huge number of paths can be leveraged to launch the flow duplication attack in the DCN.
In order to launch the flow duplication attack, attacker needs to encode more links into the BF. However, the proportion of 1 bits in the BF is constrained by $\rho_{max}$. Our theoretical analysis in Section 5.2 shows that the attacker can encode many more elements while keeping the fill factor below $\rho_{max}$. In the simulation, we want to find out how many duplicated packets can be generated if the sender sends out only one packet, when we randomly choose a set of servers as receiving destinations. The simulation evaluates the Fat-Tree DCN with 3456 servers.

We find that the attacker has more opportunities to generate duplicated flows in practice. First, the number of receivers varies. If the number of receivers is small, the attacker can have much space in the BF to encode more links to form the rhombuses. We can see from Fig. 11(b) that if there are only two receivers, the attacker can generate up to 140 redundant packets when $\rho_{max} = 0.5$. It is worth noting that the number of redundant packets increase to a peak and then fall down, especially for the curves in the upper side of the figure. This is because more receivers bring more opportunities to find the path vectors that can form rhombuses, when the space of the BF is sufficient. But after certain points, the BF has less space to encode the redundant path segments, since there are more paths need to be encoded to cover all receivers. The second opportunity comes from the fact that if the number of branches exceeds the capacity of the BF, the sender needs to produce multiple BFs, each of which covers branches leading to a subset of receivers. It is possible that some of such BFs encode a small number of branches, which can be exploited by the attacker. This is due to the fact that it may not be possible to partition the branches evenly.

6.3. Performance of Dynamic-Sized BF

In this subsection, we evaluate to what extent the dynamic-sized BF scheme can help making the in-packet BF approaching $\rho_{max}$, and to what extent the flow duplication attack can be resisted, if we must maintain a BF false positive rate threshold. We set the DCN network environment as a 48-port-switch DCN with Fat-Tree structure, and set the size of the BF $m = 1000$, the fill factor threshold $\rho_{max} = 0.5$, the BF false positive threshold $p_{max}$ in the order of $10^{-6}$. We increase the number of elements to be encoded into the BF from 2 to 40, and the step-size is 2 in our experiment, so that the imbalanced encoding situation described in Section 5.3 can appear.

We first examine the effectiveness of the dynamic-sized BF scheme in making the fill factor of every BF approaching $\rho_{max}$, where the results are shown in Fig. 12(a). In the experiments, if the number of elements in the BF is small, we choose appropriate BF parameters as described in Section 5.3 to resize the BF. Figure 12(a) shows that the dynamic-sized BF scheme can make the fill factor of every packet closer to $\rho_{max}$, which means that the attacker has lower probability to insert more 1-bits for the purpose of flow duplication attack. Without the dynamic-sized BF scheme, the BF can be sparsely populated. This will present significant vulnerability to the flow duplication attacker, as indicated in Fig. 12(a). It is notable in Fig. 12(a) that sometimes the two curves representing the scenarios with and without the dynamic-sized BF scheme are overlapping. This is because the dynamic-sized BF can only shrink the BF by a factor of $n'$, as proved in Theory 1. It happens that the resulted fill factor will exceed $\rho_{max}$ if the BF is shrunk. In this case, we have to keep the BF in its original form. We can also see in Fig. 12(a) that the fill factor for the case without dynamic-sized BF scheme can be smaller than otherwise, as shown in the right hand part of Fig. 12(a). In this case, the number of elements may exceed the capacity of a single BF, so multiple BFs need to be created. There could be only a small number of elements left for some of the BFs to encode. Without the dynamic-sized BF, the fill factor could be very low.

Fig. 12(b) illustrates the corresponding resulted false positive rate when we perform the experiment for Fig. 12(a). Besides the curve of false positive threshold, the rest of the two curves stand for the worst-case BF false positive rate among yielded BFs in each case. It can be easily seen that the false positive rate is much lower without the dynamic-sized BF scheme. This is a straightforward result because the scheme is to make the BF more densely populated with 1-bits, which can dramatically increase the corresponding false positive rate as discussed in Section 5.4. The overlapping part occurs for the same reason as that for Fig. 12(a).
Now we investigate the effectiveness of the dynamic-sized BF scheme in the perspective of the attacker. Specifically, we want to know how many elements the attacker is able to insert into the in-packet BF and keeping $\rho \leq \rho_{\text{max}}$ in the meanwhile, with and without the dynamic-sized BF scheme. The results are shown in Fig. 12 (c). It is obvious that the number of elements can be inserted into the BF is smaller with the dynamic-sized BF scheme than otherwise. The number could be up to 15 even with the dynamic-sized BF scheme, which may also incur redundant traffics. Figure 12 (d) shows the amount of redundant packets can be generated with the corresponding 1-bits can be inserted. The maximum possible number of redundant packets generated also depends on the topology of the network and positions of the receivers. The dynamic-sized BF scheme can remarkably mitigate the risk of flow duplication attack, but can not totally eliminate it.

7. CONCLUSIONS

In this study, we have presented a systematical study on security of the Bloom filter (BF) based multicast forwarding scheme. Fundamental principles of important DDoS attack models in the context of DOA have been investigated. We also have demonstrated a colluded packet injection attack model and provided the resisting mechanism of source-driven tree construction with reverse bit permutation. We have found that the particular features of the protocols with the destination oriented approach (DOA) provides a better deterrence of DDoS attacks than TOA in the Internet environment. Moreover, we have conducted some novel studies of security performance of BF based forwarding in the data center network (DCN), and demonstrated the flow duplication attack against TOA with both theoretical analysis and simulation studies. A dynamic-sized BF mechanism has been developed to prevent such an attack, where the size of BF can be reduced to lower the success ratio of the flow duplication attack while keeping BF false positive rate below the threshold. Our studies of BF based forwarding in both Internet and DCN context have shed light on how BF design parameters can influence security performance of BF based multicast forwarding. Simulation results have been presented to validate our analysis.
REFERENCES

Xiaohua Tian, Yu Cheng and Xuemin Shen. 2013. Fast channel zapping with destination-oriented multicast for IP video delivery. IEEE Trans. Parallel and Distributed Systems, 24, 2.

Deke Guo, Junjie Xie, Xiaolei Zhou, Xiaomin Zhu, Wei Wei and Xueshan Luo. 2015. Exploiting efficient and scalable shuffle transfers in future data center networks. IEEE Trans. Parallel and Distributed Systems, 26, 4.

Stephen E. Deering and David R. Cheriton. 1990. The PIM architecture for wide area multicasting. ACM Trans. Computer Systems, 8, 2.

Kevin C. Almeroth. 2000. The evolution of multicast: from the MBone to interdomain multicast to Internet2 deployment. IEEE Network, 14, 1.

Luis Henrique M.K. Costa, SergeFdida and Otto Carlos M. B. Duarte. 2007. Incremental service deployment using the hop-by-hop multicast routing protocol. IEEE/ACM Trans. Networking, 14, 3.

Tae Won Cho, Michael Rabinovich, K.K. Ramakrishnan, Divesh Srivastava, Yin Zhang. 2009. Enabling content dissemination using efficient and scalable multicast. in Proceedings of IEEE International Conference on Computer Communications (INFOCOM).

Sylvia Ratnasamy, Andrey Ermolinskiy and P. Scott Shenker. 2006. Revisiting IP multicast. in Proceedings of the ACM SIGCOMM Conference.

Petri Jokela, András Zahemszky, Christian Esteve Rothenberg, Somaya Arianfar, and Pekka Nikander. 2009 LIPSIN: Line speed publish/subscribe inter-networking. in Proceedings of the ACM SIGCOMM Conference.

George Xylomenos, Christopher N.Ververidis, Vasilios A. Siris, Nikos Fotiou, Christos Tsilopoulos, Xenofon Vasilakos, Konstantinos V. Katsaros and George C. Polyzos. 2013. A survey of information-centric networking research. IEEE Communications Surveys & Tutorials.

Hugh W. Holbrook. 2006. Source-specific multicast for IP. IETF RFC 4607.

Mikko Särelä, Christian Esteve Rothenberg, Tuomas Aura, András Zahemszky, Pekka Nikander and Jörg Ott. 2011. Forwarding anomalies in Bloom filter-based multicast. in Proceedings of IEEE International Conference on Computer Communications (INFOCOM).

Deke Guo, Jie Wu, Honghui Chen, Ye Yuan and Xueshan Luo. 2012. The dynamic Bloom filters. IEEE Transactions on Knowledge and Data Engineering, 20, 1.

Andrei Broder and Michael Mitzenmacher. 2004. Network applications of Bloom filters: A survey. Internet Mathematics, 1, 4.

Bin Xiao and Yu Hua. 2010. Using Parallel Bloom Filters for Multiattribute Representation on Network Services. IEEE Trans. Parallel and Distributed Systems, 21, 1.

Sajjad Rizvi. 2011. Performance analysis of Bloom filter-based multicast. M.S. thesis, Computer Science and Engineering Dept., Aalto Univ., Espoo, Finland.

Sajjad Rizvi, András Zahemszky and Tuomas Aura. 2012. Scaling Bloom filter based multicast with hierarchical tree splitting. in Proceedings of IEEE International Conference on Communications (ICC).

Dan Li, Yuanjie Li, Jianping Wu, Sen Su and Jiangwei Yu. 2012. ESM: Efficient and scalable data center multicast routing. IEEE/ACM Trans. Networking, 20, 3.

Dan Li, Henggang Cui, Yan Hu, Yang Xia and Xin Wang. 2008. Scalable data center multicast using multi-class Bloom filter. in Proceedings of IEEE International Conference on Network Protocols (ICNP).

Christian E. Rothenberg, Carlos A. B. Macapuna, Fábio L. Verdi, Maurício F. Magalhães, and András Zahemszky. 2010, Data center networking with in-packet Bloom filters. in Proc. Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuidos.

Kai Chen, Chengchen Hu, Xin Zhang, Kai Zheng, Yan Chen and Athanasios V. Vasilakos. 2004. Survey on routing in data centers: Insights and future directions. IEEE Network, 23, 4.

Christian E. Rothenberg, Petri Jokela, Pekka Nikander, Mikko Särelä and Jukka Ylitalo. 2009. Self-routing denial-of-service resistant capabilities using in-packet Bloom filters. in Proceedings of IEEE European Conference on Computer Network Defense.

Mikko Särelä, Christian E. Rothenberg, András Zahemszky, Pekka Nikander and Jörg Ott. 2012, Bloomcasting: Security in Bloom filter based multicast. in Information Security Technology for Applications Lecture Notes in Computer Science, 7127, 1–16.

S. Fahmy and M. Kwon, Markku Antikainen, Tuomas Aura and Mikko Särelä. 2014. “Denial-of-service attacks in Bloom-filter-based forwarding,” IEEE/ACM Trans. Networking, 22, 5.

Markku Antikainen. 2011. On the security of in-packet Bloom-filter forwarding. M.S. thesis, Computer Science and Engineering Dept., Aalto Univ., Espoo, Finland.

Xiaohua Tian, Yu Cheng and Bin Liu. 2009. Design of a Scalable Multicast Scheme with an Application-Network Cross-Layer Approach. IEEE Trans. Multimedia, 11, 6.
Xiaohua Tian and Yu Cheng. 2014. A scalable destination-oriented multicast protocol with incremental deployability. *IEEE Trans. Computers*, 63, 4.

Yakov Rekhter et al. 1995. A border gateway protocol 4 (BGP-4). IETF RFC 1771.

Zhenhai Duan, Xin Yuan and Jaideep Chandrashekar. 2008. Controlling IP spoofing through interdomain packet filters. *IEEE Trans. Parallel and Distributed Systems*, 5, 1.

Xiaohua Tian and Yu Cheng. 2012. Loop mitigation in Bloom filter based multicast: a destination-oriented approach. in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*.

Arash Partow, General Purpose Hash Function Algorithms, [http://www.partow.net/programming/hashfunctions/index.html](http://www.partow.net/programming/hashfunctions/index.html)

Geoff Huston, BGP Analysis Reports, Jan. 2013. [http://bgp.potaroo.net/as6447/](http://bgp.potaroo.net/as6447/)