# BloomCasting: Security in Bloom Filter Based Multicast

Mikko Särelä, Christian Esteve Rothenberg,
András Zahemszky, Pekka Nikander, and Jörg Ott

{mikko.sarela,andras.zahemszky,pekka.nikander}@ericsson.com,
chesteve@dca.fee.unicamp.br, jo@netlab.tkk.fi

**Abstract.** Traditional multicasting techniques give senders and receivers little control for who can receive or send to the group and enable end hosts to attack the multicast infrastructure by creating large amounts of group specific state. Bloom filter based multicast has been proposed as a solution to scaling multicast to large number of groups.

In this paper, we study the security of multicast built on Bloom filter based forwarding and propose a technique called BloomCasting, which enables controlled multicast packet forwarding. Bloomcasting group management is handled at the source, which gives control over the receivers to the source. Cryptographically computed edge-pair labels give receivers control over from whom to receive. We evaluate a series of data plane attack vectors based on exploiting the false positives in Bloom filters and show that the security issues can be averted by (i) locally varying the Bloom filter parameters, (ii) the use of keyed hash functions, and (iii) per hop bit permutations on the Bloom filter carried in the packet header.

## 1 Introduction

Recently, a number of routing and forwarding proposals [25,16,32] are re-thinking one of the most studied problems in computer networking – scalable multicast [12,23]. The unifying theme of these proposals is to use Bloom filters in packet headers for compact multicast source routing. This makes it possible for the multicast architecture to scale to the billions, or even trillions, of groups required, should the system need to support all one-to-many and many-to-many communications, such as tele and video conferencing, chats, multiplayer online games, and content distribution, etc.

While the Bloom filter is a space efficient data structure and amenable to hardware implementations, it is also prone to false positives. With in-packet Bloom filter based packet forwarding, a false positive results in a packet being erroneously multicasted to neighbors not part of the original delivery tree. Consequently, false positives lead to reduced transport network efficiency due to unnecessary packet duplications – a fair tradeoff given the potential benefits. However, false positives have also security implications, especially for network availability.

T. Aura, K. Järvinen, and K. Nyberg (Eds.): NordSec 2010, LNCS 7127, pp. 1–16, 2012.

Earlier work [26] has identified three forwarding anomalies (packet storms, forwarding loops, and flow duplication) and two solutions that provide fault tolerance for such anomalies, namely, varying the Bloom filter parameters and performing hop-specific bit permutations. Our contribution is to analyze the anomaly related problems and solutions from security perspective. It has also been shown [13] that Bloom filters can act simultaneously as capabilities, if the hash values used for the Bloom filter matching are cryptographically secure and depend on the packet flow.

In this paper, we concentrate on the security issues of Bloom filter based multicast forwarding plane. We analyze service and network infrastructure availability. The contributions of this paper are a characterization and evaluation of the security problems and solutions related to Bloom filter based forwarding. Other security issues for multicast, such as key management, policy, long term secrecy, ephemeral secrecy, forward secrecy, and non-repudiation are out of scope for this paper.

Additionally, we propose BloomCasting, a source specific multicasting technique that integrates the provided security solutions together. In BloomCasting, group membership protocol is carried from the receiver to the source. This pushes both the costs and the control of the multicast group management to the source. The Bloom filter used to forward the traffic is gathered hop-by-hop along the unicast path to the group source.

The rest of the paper is organized as follows. In Section 2, we review the principal aspects of Bloom filter based forwarding and scope the problem of secure multicast for the purposes of this paper. We present BloomCasting, a secure source-specific multicasting technique in Section 3 and in Section 4, we describe the security solutions in more detail. We evaluate our approach In Section 5, review the related work in Section 6, and conclude the paper in Section 7.

## 2 Security Issues in Bloom Filter Based Multicast

As with unicast, securing multicast communications requires considerations in two orthogonal planes: the data plane (protecting multicast data forwarding) and the control plane (securing multicast routing protocol messages), although the problems are more difficult because of the large number of entities involved. While secure multicast data handling involves the security-related packet treatments (e.g., encryption, group/source authentication and data integrity) along the network paths between the sender and the receivers, control plane security aspects involve multicast security policies and group key management i.e., secure distribution and refreshment of keying material (see e.g. [22,11,23,18,24]). Ultimately, control plane security must be handled individually by each multicast routing protocol to provide authentication mechanisms that allow only trusted routers and users to join multicast trees (e.g., PIM-SM [3]).

Our focus in this paper, however, is elsewhere – on the *availability of the multicast infrastructure* in an open and general source specific multicast model [9]. A source specific multicast group is defined by the source and group address taken

together. We assume that multicast groups can contain receivers anywhere in the network. This means that hierarchical addressing [19] cannot be used to scale up the system with sub-linear growth in routing table size in relation to the number of groups. The number of potential source specific groups grows exponentially with the number of nodes in the network – compared to quadratic growth in the number of potential unicast connections and logarithmic growth in the size of routing table based on hierarchical addressing. State requirements create a potential for denial-of-service (DoS) attacks as described in 'stateless connections' [4].

Bloom filter based source routing has been proposed as a solution to scaling multicast into large networks and number of groups [25,16,32,13]. Such an approach places the state requirement at the source, instead of the routers alleviating the potential for DoS attacks against the network infrastructure.

### 2.1 Forwarding with in-Packet Bloom Filters

The Bloom filter [10] is a hash-based probabilistic data structure capable of representing a set of elements $S$ and answering set-membership questions of the type "is $x \in S$?". The insert operations consist of, given a bit array of size $m$, for each element $x$ in a set $S$ of size $n$, $k \ll m$ independent hash values are computed $H_1(x), ..., H_k(x)$, where $1 \leq H_i(x) \leq m, \forall x$ and the corresponding bit array locations are set to 1. Conversely, asking for the presence of an element $y$ in the approximate set represented by the Bloom filter involves applying the same $k$ hash functions and checking whether all bit positions are set to 1. In that case, the Bloom filter returns a 'true', claiming that $y$ is an element of $S$. The Bloom filter always returns the right answer for each inserted elements, i.e., there are no false negatives. However, due to hash collisions, there is some probability $p(m, n, k)$ for the Bloom filter returning a false positive response, claiming an element being part of $S$ even when it was not actually inserted.

In-packet Bloom filter based multicast [25,16,32,13] is based on the idea of turning the forwarding operations into a set-membership problem. The basic idea consists of encoding a multicast tree by inserting the appropriate link identifiers into a Bloom filter carried in the packet header. Forwarding nodes along the path process the packet and check whether neighboring link identifiers are present in the Bloom filter. Then, a copy of the packet is forwarded along the matching interface(s).

Inherited from Bloom filters, false positives cause packets to be unnecessarily duplicated over some extra links. When a router receives a falsely forwarded packet for which it does not find a matching forwarding directive, the packet is simply discarded. Hence, Bloom filter forwarding guarantees packet delivery to all intended destinations but introduces a degree of wasted resources due to unnecessary packet duplications – a tradeoff worth to consider given the benefits in terms of space efficiency (i.e., reduced state) and performance (i.e., fast forwarding decisions).

### 2.2 Threat Model and Existing Attacks

We restrict the scope of this paper to security issues of the Bloom filter based forwarding plane of one-to-many multicast, also referred to as source-specific multicast (SSM) architectures. We assume an attacker who may control large number of hosts (e.g. botnet) that wishes either to disrupt the network infrastructure, or deny service to target host or network links. We also evaluate available possibilities for *controlled multicast*, i.e. ensuring that only authorized senders and receivers are capable of sending to and receiving from a particular multicast group.

Our adversary model assumes malicious end hosts and benign routers. Consequently, *packet drop attack* or *blackhole attack* fall out of the scope. This assumption is coherent with the wired networking scenario under consideration where trust among routers and the management plane is provided by e.g. pair-wise shared secret techniques. Moreover, we assume an end-to-end security mechanism to provide payload confidentiality, authentication, and integrity (e.g., as discussed in [15]). Attacks related to these security mechanisms are not discussed further in this paper.

While false positives represent a well-known limitation of Bloom filters, the security implications of (random) false positives in packet forwarding are far reaching and less understood. Our main security goal is to guarantee *forwarding service availability* of Bloom filter based data planes under malicious attacks. Hence, we seek for data plane mechanisms that ensure that only packets from authorized users are forwarded, i.e., providing resistance to (potentially distributed) DoS attacks .

DoS can be divided into attacks on infrastructure availability and (end) service availability. These can be disrupted by bandwidth, state, or computation consumption attacks (cf.[7]). Any unauthorized sending of multicast data can be construed as a DoS attack. For instance, *flooding attacks* would cause an escalating of packets filling the network links to an extend that legitimate packets end up discarded due to massive link congestion. Such denial of service may affect a greater proportion of the network due to the "multiplier effect" of false-positive-prone multicast packet distribution.

**Chain Reaction Attacks.** False positives can cause forwarding anomalies that greatly increase the amount of network traffic. These include packet storms, forwarding loops, and flow duplication [26]. We review these anomalies that an attacker could do here. We highlight the fact that if Bloom filters are assigned per multicast tree or per flow, the anomalies will affect every packet in a given multicast tree or flow.

*Packets storms* are caused when, for sizable part of the network, the average number of false positives per router exceeds one. Should this be the case, then on average each false positive causes more than one additional false positive, creating an explosive chain reaction. The average number of false positives is $\rho^k \cdot (d - b - 1)$, where $\rho$ is the fill factor of the Bloom filter, $k$ is the number of hash functions used, $d$ is the number of neighbors, and $b$ is the number of actual

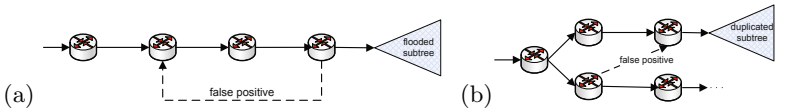



**Fig. 1.** (a) Forwarding loop and (b) flow duplication

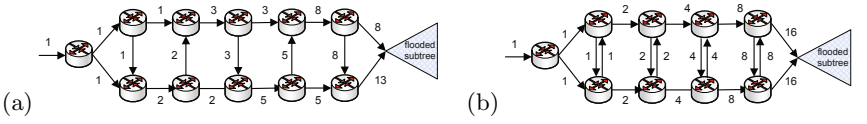



**Fig. 2.** (a) Flow duplication with Fibonacci number growth in the number of packet copies and (b) exponential growth in the number of packet copies

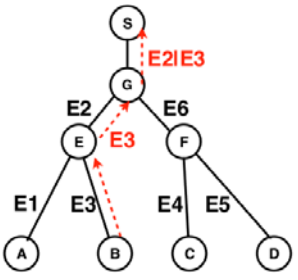
branches in the multicast tree at that node. After the first false positive $b = 0$. As an example, considering $k = 5$ and $\rho = 0.5$, a nodes with degree $d > 32$ would produce more than one false positive per node.

*Forwarding loop* happens, if a set of false positives cause the packet to return to a router it has already visited. The router will then forward the packet again to all the nodes downstream of it, including the false positive links that caused the packet to loop. As a result, not only will the packet loop, but every loop causes a copy of the packet to be sent to the full sub-tree below the router. A forwarding loop is shown in Figure 1.

*Flow duplication* is another possible anomaly as shown in Figure 2. Figures 2(b)-(c) show that even flow duplication can cause the number of packet to grow – according to Fibonacci sequence and as the powers of two.

The above attacks can also be combined. If link identifiers are common knowledge, the attacker can form a Bloom filter that corresponds to the Figure 2(c) which also includes one or more links back to the first router, causing the packet load to explode both in network and in all receiver hosts.

**Target Path Attack.** An attacker controlling a large number of hosts can try to coordinate as many packet flows as possible to a single link or a particular path. If link identifiers are common knowledge (1), then this is simple. Each host just computes a forwarding tree that goes through chosen link. If however, the link identifiers are secret and static (2), then the attacker has a few potential attacks available: *injection attack* – where she tries Bloom filters that get traffic forwarded along a certain delivery tree, *correlation attack* – where she attempts to infer workable link identifiers from a collection of legitimate Bloom filters, and *replay attack* – where a valid Bloom filter is misused to send unauthorized traffic (i.e., with different content or flow identifiers). [13]




**Source S**

| Receiver | BF |
|---|---|
| A | E1\|E2 |
| B | E2\|E3 |
| C | E4\|E6 |
| D | E5\|E6 |
| ABCD | E1\|E2\|E3\| E4\|E5\|E6 |

**Fig. 3.** The left side shows multicast Join message using iBFs. The right side shows a simplified Membership Table MT(S) that contains the Bloom filters for A, B, C, and D. The separated bottom row shows how to combine the Bloom filters in to an iBF.

## 3 BloomCasting

BloomCasting is a secure source specific multicast technique, which transfers the membership control and per group forwarding state from the multicast routers to the source. Similar to [25,16,32], it uses in-packet Bloom filter (iBF) to encode the forwarding tree. BloomCasting separates *multicast group management* and *multicast forwarding*.

To join, a host sends a join request (`BC_JOIN`) towards the source S. Intermediate routers record forwarding information into the packet, thus when the packet reaches S, it will contain a *collecting* iBF for the source-receiver path. By combining together the iBFs for all the receivers, the source will have an iBF encoding for the whole multicast tree. When a host does not wish to receive packets for the group anymore, it sends an authenticated leave message to S. Upon processing this packet, the source will reconstruct the Bloom filter for the group leaving out the recently pruned path. The operation is illustrated on Figure 3.

Data packets are routed using the *forwarding* iBF placed in the `BC_FW` header. Each intermediate router takes its forwarding decision by querying it with the question: which of my outgoing links are present in the iBF? It then forwards the packet to the corresponding peers. Eventually, the packet reaches all the receivers, following the sequence of routers the `BC_JOIN` packets traversed, in reverse order.

### 3.1 Group Membership Management

Group membership management includes the joining, leaving, and maintenance of multicast groups, and this is the main task of the control plane. Along this discussion, we show how multicast trees are encoded into iBFs.

**Joining a Group:** When a host joins a multicast group, it sends a (`BC_JOIN`) message towards the source. The packet contains the following information: (S,G) specifying the multicast group and a *collecting* iBF. The latter is used for

**Algorithm 1.** Adding edge-pair labels (E) and permuting collect and forward iBFs at transit routers.

```
Collect_iBF (C):
  E ← Z_K(S,G,R_p, R_c, R_n);
  C ← C ∨ E;
  C ← Permute_c(C);
Forward_iBF (F):
  foreach outgoing link i do
      F ← Permute_c^{-1}(F);
      E ← Z_K(S,G,R_n, R_c, R_p);
      if E ∧ F = E then
          Send F → i;
      end
  end
```

collecting the forwarding information between the source and the receiver. Finally, it also contains a hash chain anchor for future signaling security.

In each router, the next hop for the `BC_JOIN` message towards S is found from routing information base.[1] As the message travels upstream towards the source, each router records forwarding information into the packet by inserting the edge pair label E into the *collector* iBF. After this, for loop prevention and increased security, it performs a bit permutation on the *collector* iBF. Finally, it selects the next hop usptream towards S. The operation is shown on Algorithm 1. Unlike traditional IP multicast approaches, where the forwarding information is installed in routers on the delivery tree, transit routers do not keep any group-specific state.

Once the `BC_JOIN` message reaches the source, it contains sufficient information so that the source can send source-routing style packets to the recently joined host. The source stores this information in the Membership Table (MT), as shown in Figure 3. The source can now send packets to the multicast tree by combining iBFs for the group, by bitwise ORing them together.

**Leaving a Group:** When a receiver wishes to leave the group, it sends a `BC_LEAVE` towards S, including the next element from the hash chain it used when joining the group. On-path routers forward the packet to S. As no further processing is needed in intermediate routers, unlike pruning packets in IP multicast, `BC_LEAVE` packets always routed to the source.

S verifies the hash and removes (or de-activates) the entry in the Membership Table. Single message hash authentication, vulnerable to man-in-the-middle attacks, is sufficient, since the hash is only used to verify that the host wishes to leave the group. As a final step, it recomputes the forwarding iBF of the delivery tree. An example of a forwarding iBF is shown in Figure 3 at the separated bottom row of the table.

[1] Just like in standardized IP multicast protocols, this forwarding decision can be taken according to the RIB created by BGP or according to the Multicast RIB created by MBGP [8].

**Refreshing Membership State:** The iBFs in the MT may become stale, either because of changing the key to compute the edge-pair labels or due to route failures. Keys are expected to change periodically (e.g., every few hours) to increase security by excluding brute force attacks [13]. This means that the iBF needs to be recomputed with a new `BC_JOIN` packet. When making the forwarding decision, during a transition period routers need to compute edge-pair labels for both the old and the new key. If they find that an edge-pair label computed with the old key is present in the iBF, they set a flag in the `BC_FW` header indicating that the receiver should send a `BC_JOIN` again, as the iBF will soon become invalid. When a packet is to be forwarded on a failed link, the router sends an error message back to the source.

### 3.2 Multicast Forwarding

So far, we have discussed how hosts join and leave multicast groups. We now show how data packets are forwarded between the source and the receiver.

As we saw previously, iBFs for each receiver border router are stored separately in the Membership Table. We also saw the basic concept of deriving the forwarding iBF from the MT information; now we extend that with new details.

For each group, the source stores one or more iBF for each next hop router in its BloomCasting Forwarding Table (BFT).[2] In practice, the capacity of a packet-size iBF is limited in order to guarantee a certain false positive performance (practical values suggest around 25 destinations in 800-bit iBFs [25]). In case of large multicast groups, several iBFs are created, one for each partial multicast tree, and duplicate packets are sent to each next hop.

The source creates one copy of the packet for each next hop for (S,G) in the BFT. It creates a `BC_FW` header, fills it with the corresponding iBF, and sends it to the next hop router.

Each router makes a forwarding decision based on the iBF, as shown in Algorithm 1. First, it applies the reverse permutation function to the iBF, replacing the iBF with the result. Then, it checks for the presence of peer routers by computing one edge-pair label for each potential next hop router $R_n$, based on the previous and the current router on path $R_p$ and $R_c$ respectively,[3] and on group identity (S,G) found in the IP header as shown in Algorithm 1. In the final step, the router checks whether the iBF contains the edge-pair label, by simple bitwise AND and comparison operations.

The remaining problem is how to compute the dynamic edge-pair labels at core routers at line speed. This can be done by taking the values $(S, G, K, Rp, R_c, R_n)$ and running them through a fast, spreading hash function (cf. e.g. [20,31]). The spreading hash function yields the bit locations for the edge-pair labels. The method can be applied locally at each router, having no impact on the protocol.

[2] This improves forwarding performance, as the false positive probability increases with the number of iBF inserted elements.

[3] The router uses the same inputs as in the `BC_JOIN`. hence the $R_p$ and $R_n$ switch places due to direction change.

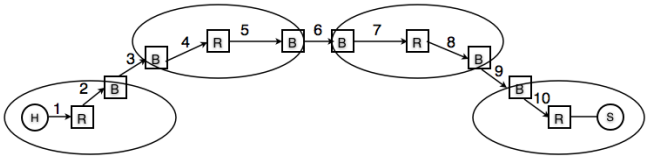



**Fig. 4.** Protocol messages when joining group (S,G): 1 - IGMP Membership Report or MLD Multicast Listener Report; 2,10 - PIM-SSM JOIN(S,G); 3-9 - BC_JOIN(S,G)

### 3.3 Connecting Intra-domain Multicast with BloomCasting

BloomCasting can be used to specify the operations between source and receiver ASes.[4] This section discusses how multicast forwarding state is set up inside the domains containing the sender and/or receivers using IP multicast (PIM-SSM deployments) Figure 4 illustrates the protocol messages when a multicast receiver joins a multicast group (S,G).

When a receiver joins (S,G), it signals (1) its interest in its LAN with IGMPv3 or MLDv2 protocols. The Designated Router then sends a PIM-SSM `JOIN(S,G)` message upstream (2), by looking up the reverse path to S. The message is propagated upstream until a router is found that holds forwarding state for the group or until a border router of the domain is reached (standard PIM-SSM operations). The border routers implement both PIM-SSM and BloomCasting. PIM signaling now terminates. If the border router was not yet a receiver for the group, it creates a `BC_JOIN` packet and sends it towards S (3-9).

The iBF collection process is otherwise as described in Section 3.1 except each AS is considered to be a singe logical router.

At the other end, the source AS border router receives a `BC_JOIN` for a group that resides in the local domain and processes it as specified in Section 3.1. If it is not yet a receiver for the group locally, it sends a join packet using PIM-SSM standardized operations (10). The `JOIN(S,G)` is propagated further upstream towards the source, with standard PIM operations. Eventually, as far as PIM concerned, a domain-local multicast tree will be built with routers representing local receivers and border routers representing subscribers in remote domains.

The data packets are *forwarded* using the domain-local multicast protocol to the border routers in the source AS. The border router creates a single copy for each entry in the BFT, adds the BloomCasting header, and forwards the packets. When an ingress border router receives packet with the `BC_FW` header, it checks whether it has domain-local receivers and forwards a decapsulated copy using the domain-local multicast protocol. The router also checks whether neighboring domains are included in the iBF and forwards the packet to those domains (using e.g. IP-in-IP, GRE encapsulation, or MPLS paths or trees).

[4] An AS is an autonomous system, a network participating in the inter-domain routing protocol (BGP). The source and receiver could also be an area consisting multiple ASes that deploys a shared multicast architecture.

## 4 Security Techniques in Bloom Filter Based Forwarding

In Section 2, we introduced a threat model for in-packet Bloom filter based forwarding by showing several attacks taking advantage of some forwarding anomalies inherent to Bloom filter based forwarding. Now, we present techniques for solving these forwarding anomalies; then, in Section 5, we evaluate them from security perspective.

Basically, the solutions presented here include pre-processing verification of Bloom filters, and some rules to be followed in the packet forwarding process and during the Bloom filter creation phase.

*Limiting the fill factor ($\rho_{max}$)* ensures that the attacker cannot set, e.g., all bits in the Bloom filter to 1, which would equal to every link in the network being included. Before any packet handling operation, routers need to verify the Bloom filter [30], i.e. they need to check for $\rho_{max}$ compliance before attempting to forward the packet. Typically, $\rho_{max}$ is set to $\approx 0.5$, which corresponds to the most efficient usage in terms of bits per element per false positive rate.

*Cryptographic Bloom filters:* Bloom filters for forwarding can be differentiated based on the nature of the link identifiers: (1) link identifiers are common knowledge [25], (2) link identifiers are secret, but static [16], and (3) link identifiers are secret per flow and change periodically with key are computed per incoming/outgoing edge pair instead of per link [13].

Bloom filters gain capabilities [2], when the edge pair label is computed using cryptographically secure hash functions, secret key, and flow information from the packet (e.g., IP 5-tuple, $(S, G)$). Each link identifier of size $m$ and with $k$ bits set to one (i.e., a one element Bloom filter) can be computed as the output of a function $zF(In, Out, K(t_i), I, p)$. The resulting identifiers become *dynamic* and bound to the $In$ and $Out$ interfaces of each hop $i$, the time period of a secret key $K(t_i)$, and additionally dependent of packet information like the Flow ID $I$ (e.g., source and group addresses) and an optimization parameter $p$ (cf.Z-formation [13]).

*Varying the number of hash bits ($k_{var}$):* This technique deals with the number of ones ($k$) in the link identifiers set by different routers, and aims to decrease the false positive rate. Assuming that there is a fixed maximum fill factor $\rho$ for iBF, e.g. $\rho = 0.5 + \epsilon$, the average number of false positive in a given router depends on its degree $d$, the number of hash functions $k$ it uses, and the number of outgoing branches $b$ such that the average number of false positives is $\rho^k \cdot (d - b - 1)$. Hence, we proposed [26] that each router sets $k$ locally such that $\rho^k \cdot d < \alpha$, where $\alpha < 1$ sets the routers preference for the average false positive rate. As routers compute the hash functions for the *collecting* iBF themselves, the number $k$ is purely a local matter. In other words, the number of bits $k$ set to 1 in the link identifiers is not a global parameter, but can be defined per node.

*Permutations $P_i(iBF)$:* We use per hop bit permutations to prevent loops and flow duplications. A bit permutation is a (pseudo) random rearrangement of the bit array. Each router can use the same bit permutation for all iBFs passing through it making it easy to implement with programmable logic.

First, after passing through the intended path to a router R, the $f$orwarding iBF has to match the $k$ hash values that the router added when the *collecting* iBF was forwarded through it. When the iBF is collected, the routers between R and source S change some bits from 0 to 1 and permute the packet. S then combines a set of *collecting* iBFs in to a *forwarding* iBF and the routers between S and R (including R) perform reverse permutations on the iBF. Hence, once the packet arrives in R, the bits that R set to 1 will be in exactly the same positions as they were when the iBF was collected. Since no operation changes a value of a bit from 1 to 0, the matching process works correctly.

Second, if the path taken is different from the one intended for the packet, the iBF should not match the $k$ hash values. Per hop bit permutations enable the iBF itself to carry a "memory" of the path it has traversed [26]. As each router modifies the iBF when forwarding the packet, after passing through two different edges and entering back the initial node, i.e. after a loop, the iBF is changed with a random bit permutation. Hence, it will likely not match the same edge-pair labels again. Each router permutes $P_i(iBF)$ when the iBF is initially collected and then reverse permutes $P_i(iBF)^{-1}$ the iBF when a packet is sent using the iBF.

## 5 Security Evaluation

We now analyze how BloomCasting mitigates the security threats (described in Section 2) against Bloom filter based multicast forwarding. As mentioned in Section 2.2, we focus on malicious host-initiated attacks. Further architectural considerations w.r.t scalability, bandwidth efficiency, state requirements, control overhead, etc. are out of scope of this evaluation and left for future work.

Table 1 presents an overview of the mapping between the available techniques (Section 4) and the attacks addressed. As can be seen, BloomCasting combines four techniques to prevent the six security threats described in Section 2.2.

*Packet storms* are prevented with the combination of limiting the maximum fill factor $\rho_{max}$ and the *varying* $k_{var}$ *technique*. Globally enforced $\rho_{max}$ values enable each router to compute $k_{var}$ locally so that every Bloom filter with a valid fill factor produces, on average, less than 1 false positives. Since the Bloom filters are collected on path with the `BC_JOIN` packet, it is easy to set $k_{var}$ locally. Additionally, this optimization of $k$ reduces the actual false positive rate [26].

*Loops* are a serious threat to any network infrastructure. The combination of maximum fill factor $\rho$ and z-Formation makes it difficult for an attacker to construct looping Bloom filters. The first removes the easy attack of just adding bits into the Bloom until every link matches and the z-Formation ensures that guessing the right Bloom filter is difficult (see [13] for details).

To prevent accidental loops, each router performs a bit permutation on the Bloom filter before performing the outport matching – when using the Bloom filter for forwarding (and after matching – when collecting a Bloom filter). If a packet does go through a loop, either because of a false positive or a malicious source, the Bloom filter has been modified with a random permutation (a product of the permutations performed by the set of routers participating in the loop).

**Table 1.** Mapping of solutions to attacks

| Attack - Technique | $\rho_{max}$ | $k_{var}$ | $z$-$F$ | $P(iBF)$ |
|---|---|---|---|---|
| Packet storms | + | + | | |
| Loops | + | | + | + |
| Flow duplication | + | | + | + |
| Injection | + | | + | |
| Correlation | | | + | + |
| Replay | | | + | |

Using permutations ensures a high probability that the packet will not continue looping and that it will not be forwarded to the downstream tree for a second, or nth time. As an example, the chances of an infinite loop in a three node loop configuration with $\rho = 0.5$, $k = 6$, and $m = 256$ are in the order of $O(10^{-12})$. The chances that a packet will be forwarded through the subtree once are $\rho^{\kappa}$, where $\kappa = \sum k_i$ is the sum of all hash functions used in the subtree.

Finally, while the security is not dependent on the secrecy of the permutations performed in each router, it is dependent on the secrecy of the edge-pair labels. Consider a *known permutation* attack, in which an attacker knows the network topology and the permutations used by a set of routers. It can now compute the cycle sets of the combined permutation and choose a combination that has approximately the size of the maximum fill factor. However, it does not know a combination of a Bloom filter and source and group address that will ensure that the routers on path and in the loop will have edge-pair labels that match the Bloom filter. The best it can do is vary the group address. In this case, the probability of success is $\rho^{\kappa}$, where $\kappa$ is the total number of bits that need to be set on path to the point of loop and in the loop.

*Flow duplication:* Similarly to loops, flow duplication can be effectively prevented with the combination of restricting fill factor $\rho$, edge-pair labels, and per hop bit permutations. The result gives an attacker $\rho^{\kappa}$ probability of creating a specific subtree by accident.

*Packet injection attacks, correlation attacks, and replay attacks* can be efficiently prevented using the z-Formation technique [13]. It uses cryptographically secure edge-pair labels that are computed based on the flow, and time, and path. This makes it impossible to share iBFs from different points of network, at different time instants, or to different destinations.

Consequently, the best strategy for a successful *packet injection attack* is reduced to a brute force attack consisting of generating random labels and hoping that at least one of them reaches the target(s). An attacker needs malformed iBFs to cause $h$ consecutive false positives to get packets forwarded through a valid iBF path of length $h$. The chances of success in one attempt can be approximated to $p = {\rho_{max}}^{h \cdot k}$, which is very low for typical configurations (e.g., $p = 2^{-36}$ for $h = 4, k = 8, \rho = 0.5$, i.e., over $10^{10}$ attempts are required for a 1/2 probability successful attack). Such brute force attacks can be easily detected, rate limited and pushed back, for instance after the false positive rate

from a given source exceeds some threshold. Additionally, a forged iBF would work through the target path attack as long as the distributed secret K(t) is not renewed.

*Source and receiver control:* As the group management in BloomCasting is end-to-end, it gives source control over the receivers it accepts. If it wishes to, it can require receiver authentication before adding a receiver into the group. Similarly, multicasting to a receiver requires knowing the iBF that forms the path between source and destination. Since the iBF is cryptographically bound to (S,G), each router's secret key, and the path (via permutations and edge-pair labels), guessing an iBF for a path is difficult, as shown above.

*Resource consumption attacks* against the memory and processing capacity of routers do not become easier than they are in unicast forwarding. The routers do not need to maintain multicast state and the iBF collection and forwarding processing can be done in line speed in hardware and in parallel for each peer. The multicast source needs to maintain state for receivers. This is a needed feature, since this makes it possible source control over who can and who cannot join the multicast group. Simultaneously, it leaves the source vulnerable to attacker who creates a storm of multicast join packets. A source can use a receiver authentication protocol, which pushes the authentication protocol state to the initiator (e.g., the base exchange of Host Identity Protocol [21] could be used for that purpose) to limit the state requirements to authenticated receivers.

False positive forwarded packets may compromise the *ephemeral secrecy* of the multicast data to non group-members, i.e., some packets may reach unintended destinations. The time- and bit-varying iBFs contribute to spreading false multicasted packets across different links over time, preventing thus a complete reception of a multicast packet flow.[5]

*Anonymity* of source is not an option in source specific, since the group is identified with combination (S,G) where S is the sender address and G the group address. However, even though the protocol uses source routing, the actual paths, or nodes on path are not revealed to the source and the source can only use the related iBFs in combination with traffic destined to (S,G).

Receivers do not need to reveal their identifies or addresses to the network, or the source – the receiver (IP) address is not necessary in the protocol. The authentication, should the source require it, can be done end-to-end without revealing the identities to the intermediate routers. As the keys used to compute iBFs are changed periodically, correlation attacks between two or more Bloom filters used at different times become impossible. Similarly, since the edge-pair labels are tied to group identifier (S,G), an attacker cannot use a set of iBFs with different group addresses to determine whether the set contains one or more common receivers. These techniques effectively prevent traffic analysis and related vulnerabilities such as clogging attacks (cf. [5]).

[5] As assumed in Section 2, *data authenticity* is kept out of scope of the iBF forwarding service and can be provided by orthogonal security policies and group key management techniques (e.g., following the guidelines of [15]).

## 6 Related Work

Compared with unicast, multicast communication is at a substantially increased risk from specific security threats due to the lack of effective group access control and larger opportunities for link attacks. Over the last decade, much effort has been put in the field of multicast security, see e.g. [6,18,11,15,27].

At the IETF, earlier work has provided a taxonomy of multicast security issues [11] and a framework for secure IP multicast solutions [14] to address the three broad core problem areas identified: (i) fast and efficient source authentication (e.g. [6,17]), (ii) secure and scalable group key management techniques, and (iii) methods to express and implement multicast-specific security policies. Our focus, however, has been on DoS attacks against the network infrastructure.

Service availability attacks due to routing loops and blackholes were discussed in [28]. The proposed solution was the keyed HIP (KHIP) protocol to allow only trusted routers joining the multicast tree. Our aim is a general and open SSM architecture that does not require group access restrictions provided by the infrastructure.

Free Riding Multicast [25] proposes an open any source multicast service in which each link is encoded as a set of hashes from the AS number pair. This leaves the forwarding plane open to a variety of attacks. Odar [29] showed that Bloom filters can be used for anonymous routing in adhoc networks. Limiting fill factor as a security feature in Bloom filter based (unicast) capabilities was first proposed in [30]. LIPSIN [16] uses Bloom filter forwarding plane for publish/subscribe architecture with a separate topology management system that helps to keep the link identifiers secret. However, an attacker can still use target path attacks. Z-formation [13] prevents target path attacks by using edge-pair labels that depend on flow identifier, but is still open to e.g. chain reaction attacks.

$Si^3$ [1] proposed a secure overlay to solve problems related to secure multicast. While distributed hash tables spread load efficiently across the system, they lack e.g. policy compliant paths and control over who is responsible for particular connection.

## 7 Conclusions

In this paper, we evaluated the security of Bloom filter based forwarding. False positives inherent to Bloom filters enable a host of attacks on target service and network infrastructure availability. These attacks include chain reaction attacks, which use the Bloom filter properties (e.g. false positives) to ensure that the network forwarding infrastructure multiplies every packet sent using the Bloom filter and targeted attacks in which the attacker enables many nodes to target the same path in the network.

We show that these problems can be solved by the combination of limiting Bloom filter fill factor, both minimum and maximum, using cryptographically computed edge-pair labels in the Bloom filters, varying the number of hash functions locally based on the router degree, and using per hop bit permutations on the Bloom filter.

We also proposed BloomCasting, a secure source-specific multicasting technique based on in-packet Bloom filters. The technique is based on end-to-end signaling of group membership and hop-by-hop collection of the needed Bloom filters. As future work, we intend to study the possibility of collecting multiple paths in advance as a technique for increasing fault tolerance to route failures.

## References


1. Adkins, D., Lakshminarayanan, K., Perrig, A., Stoica, I.: Towards a more functional and secure network infrastructure (2003)
2. Anderson, T., Roscoe, T., Wetherall, D.: Preventing Internet denial-of-service with capabilities. ACM SIGCOMM Computer Communication Review 34(1), 44 (2004)
3. Atwood, W., Islam, S., Siami, M.: Authentication and Confidentiality in Protocol Independent Multicast Sparse Mode (PIM-SM) Link-Local Messages. RFC 5796 (Proposed Standard) (March 2010), http://www.ietf.org/rfc/rfc5796.txt
4. Aura, T., Nikander, P.: Stateless Connections. In: Han, Y., Quing, S. (eds.) ICICS 1997. LNCS, vol. 1334, pp. 87–97. Springer, Heidelberg (1997)
5. Back, A., Möller, U., Stiglic, A.: Traffic Analysis Attacks and Trade-Offs in Anonymity Providing Systems. In: Moskowitz, I.S. (ed.) IH 2001. LNCS, vol. 2137, pp. 245–257. Springer, Heidelberg (2001)
6. Ballardie, T., Crowcroft, J.: Multicast-specific security threats and countermeasures. In: SNDSS 1995: Proceedings of the 1995 Symposium on Network and Distributed System Security (SNDSS 1995), p. 2. IEEE Computer Society, Washington, DC (1995)
7. Barbir, A., Murphy, S., Yang, Y.: Generic Threats to Routing Protocols. RFC 4593 (Informational) (October 2006), http://www.ietf.org/rfc/rfc4593.txt
8. Bates, T., Chandra, R., Katz, D., Rekhter, Y.: Multiprotocol Extensions for BGP-4. RFC 4760 (Draft Standard) (January 2007), http://www.ietf.org/rfc/rfc4760.txt
9. Bhattacharyya, S.: An Overview of Source-Specific Multicast (SSM). RFC 3569 (Informational) (July 2003), http://www.ietf.org/rfc/rfc3569.txt
10. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM 13(7), 422–426 (1970)
11. Canetti, R., Pinkas, B.: A taxonomy of multicast security issues. IRTF Internet-Draft (draft-irtf-smug-taxonomy-01) (August 2000)
12. Diot, C., Dabbous, W., Crowcroft, J.: Multipoint communication: A survey of protocols, functions, and mechanisms. IEEE Journal on Selected Areas in Communications 15(3), 277–290 (1997)
13. Esteve, C., Jokela, P., Nikander, P., Särelä, M., Ylitalo, J.: Self-routing Denial-of-Service Resistant Capabilities using In-packet Bloom Filters. In: Proceedings of European Conference on Computer Network Defence, EC2ND (2009)
14. Hardjono, T., Canetti, R., Baugher, M., Dinsmore, P.: Secure ip multicast: Problem areas, framework, and building blocks. IRTF Internet-Draft (draft-irtf-smug-framework-01) (September 2000)
15. Hardjono, T., Weis, B.: The Multicast Group Security Architecture. RFC 3740 (Informational) (March 2004), http://www.ietf.org/rfc/rfc3740.txt
16. Jokela, P., Zahemszky, A., Esteve, C., Arianfar, S., Nikander, P.: LIPSIN: Line speed publish/subscribe inter-networking. In: SIGCOMM (2009)

17. Judge, P., Ammar, M.: Gothic: a group access control architecture for secure multicast and anycast. In: INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, vol. 3, pp. 1547–1556 (2002)
18. Judge, P., Ammar, M.: Security issues and solutions in multicast content distribution: A survey. IEEE Network 17, 30–36 (2003)
19. Kleinrock, L., Kamoun, F.: Hierarchical routing for large networks Performance evaluation and optimization. Computer Networks 1(3), 155 (1976/1977)
20. Krawczyk, H.: LFSR-Based Hashing and Authentication. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 129–139. Springer, Heidelberg (1994)
21. Moskowitz, R., Nikander, P.: Host Identity Protocol (HIP) Architecture. RFC 4423 (Informational) (May 2006), http://www.ietf.org/rfc/rfc4423.txt
22. Moyer, M., Rao, J., Rohatgi, P.: A survey of security issues in multicast communications. IEEE Network 13(6), 12–23 (1999)
23. Paul, P., Raghavan, S.V.: Survey of multicast routing algorithms and protocols. In: ICCC 2002: Proceedings of the 15th International Conference on Computer Communication, pp. 902–926. International Council for Computer Communication, Washington, DC (2002)
24. Rafaeli, S., Hutchison, D.: A survey of key management for secure group communication. ACM Computing Surveys (CSUR) 35(3), 329 (2003)
25. Ratnasamy, S., Ermolinskiy, A., Shenker, S.: Revisiting IP multicast. ACM SIGCOMM Computer Communication Review 36(4), 26 (2006)
26. Särelä, M., Rothenberg, C.E., Aura, T., Zahemszky, A., Nikander, P., Ott, J.: Forwarding Anomalies in Bloom Filter Based Multicast. Tech. rep., Aalto University (October 2010)
27. Savola, P., Lehtonen, R., Meyer, D.: Protocol Independent Multicast - Sparse Mode (PIM-SM) Multicast Routing Security Issues and Enhancements. RFC 4609 (Informational) (October 2006), http://www.ietf.org/rfc/rfc4609.txt
28. Shields, C., Garcia-Luna-Aceves, J.J.: Khip—a scalable protocol for secure multicast routing. In: SIGCOMM 1999: Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, pp. 53–64. ACM, New York (1999)
29. Sy, D., Chen, R., Bao, L.: Odar: On-demand anonymous routing in ad hoc networks. In: Proc. of IEEE Mobile Adhoc and Sensor Systems (MASS), pp. 267–276 (2006)
30. Wolf, T.: A credential-based data path architecture for assurable global networking. In: Proc. of IEEE MILCOM, Orlando, FL (October 2007)
31. Yuksel, K.: Universal hashing for ultra-low-power cryptographic hardware applications. Ph.D. thesis, Citeseer (2004)
32. Zahemszky, A., Jokela, P., Särelä, M., Ruponen, S., Kempf, J., Nikander, P.: MPSS: Multiprotocol Stateless Switching. In: Global Internet Symposium 2010 (2010)