Selecting Bloom-filter header lengths for secure information centric networking

Selecting Bloom-filter Header Lengths for Secure Information Centric Networking

Bander A. Alzahrani*, Vassilios G. Vassilakis**, and Martin J. Reed*
*School of CSEE, University of Essex, Colchester, U.K.
**Dept. of Electronic Engineering, University of Surrey, Guildford, U.K.

Abstract—**Information-centric networking (ICN) is a new communication paradigm that shifts the focus from end hosts to information objects. Recent studies have shown that ICN can provide more efficient mobility support and multicast/anycast content delivery compared to traditional host-centric solutions. Nevertheless, the ICN solutions proposed so far are not very mature from the security viewpoint. In this paper, we study one of the most important Bloom-filter based ICN forwarding mechanisms and discuss its security vulnerabilities. Next, we propose some enhancements to this mechanism, which aim at increasing its resistance to brute-force attacks. Our proposed solutions are supported by simulation studies.**

Keyword—Information centric networks; Bloom filter; Brute force attacks;

I. INTRODUCTION

It is widely recognised that the current Internet architecture does not cope well with the today's user demands [1, 2]. The Internet is suffering from security, mobility, scalability, and quality-of-service issues, just to name a few. In recent years, a new communication paradigm, namely information-centric networking (ICN), appeared with the ambitious goal of solving the abovementioned shortcomings [3-5]. ICN marks a fundamental shift from the current host-centric communication paradigm. Rather than naming the hosts, ICN uses the information objects themselves as first class citizens.

The ICN approach has recently been explored by a number of research projects, including the Data Oriented Network Architecture (DONA) [6], Named Data Networking (NDN) [7], Scalable and Adaptive Internet Solutions (SAIL) [8], Content Mediator Architecture for Content Aware Networks (COMET) [9], and Publish-Subscribe Internet Technology (PURSUIT) [10].

Our work is based on the PURSUIT ICN architecture. This architecture relies on the concept of the publish/subscribe paradigm. This is different from the current send/receive paradigm used in today's Internet. The PURSUIT architecture comprises of the following network entities: publishers, subscribers, and brokers [11]. Publishers advertise the available information objects by issuing the publication messages. Subscribers issue the subscription messages indicating their interest in receiving some information objects. Brokers perform three functions, namely rendezvous (RV) [12], topology management (TM) [13], and forwarding (FW) [14]. The RV is responsible for matching subscriptions with publications. The TM is responsible for constructing a delivery path for the publisher to the subscriber. The FW is responsible for delivering the information object from the publisher to the subscriber.

Each information object in the network is distinguished through a pair of identifiers, the rendezvous identifier (RID) and the scope identifier (SID). The RID uniquely identifies the object within a scope, whereas the SID is used to organise the information objects.

Research efforts on ICN have already provided notable solutions in the areas of network efficiency, reduced complexity, scalability and reliability, mobility support, multicast and caching performance, traffic engineering, network coding, etc. However, in the security domain much work still needs to be done.

ICN can be subject to new types of denial-of-service (DoS) attacks [15, 16]. One possible attack scenario can be achieved by exploiting the forwarding function. In [14] the authors propose the LIPSIN forwarding mechanism for publish/subscribe networks. In [17], it has been shown that the LIPSIN mechanism is vulnerable to replay attacks which may lead to DDoS. Therefore, in [17] the Z-formation technique has been introduced to offer a distributed-denial-of-service resistant forwarding. In [18], the authors analyse the Z-formation technique in terms of the scalability of the TM function. In [19], the works of [17] and [18] have been extended to prevent brute-force attacks on the LIPSIN mechanism. The work of [19] has been further improved in [20], where the authors calculate an optimal Bloom filter fill factor for reducing the DDoS attack probability.

In this paper, we demonstrate the problem of brute-force attacks and improve on the analysis shown in [20]. We also propose a solution that makes a balance between mitigating brute-force attacks and in the same time be scalable for large networks. The rest of this paper is organised as follows. In Section II, we provide a brief overview of the LIPSIN forwarding mechanism and discuss its security vulnerabilities. In Section III, we present the brute force attack probability and its reaming issue. Section IV, we analyse our proposed solution and present the results. We conclude the paper in Section V.

II. BACKGROUND

A. The LIPSIN forwarding mechansim

The LIPSIN [14] is one of the proposed forwarding mechanisms in ICN and it is based on Bloom filter [21] forwarding. Bloom filters are probabilistic, space-efficient data structures that enable fast and cost-effective creation and membership tests. LIPSIN uses a Bloom filter to encode the source routing path from the publisher to

subscriber. The encoded path is sent by the TM function to the publisher and is used as the forwarding identifier (FID) by the forwarding function.

In order to enable a Bloom filter based forwarding, each link in the network is assigned a unique link identifier (LID). An LID is an m-bit array with k bits set to 1. That is, k is the number of hash functions used to create a LID. Typically k is much smaller than m and its value may be selected based to some optimization objectives [22], e.g. m = 256 bits and k = 5 are typically chosen values [14, 13]. The number, n, of statistically unique LIDs is given by [14]:

$$n = \frac{m!}{(m-k)!} \quad (1)$$

For instance, if m = 265-bit and k = 5, then $n \approx 1.2 \times 10^{12}$.

Recall that the TM function is responsible for the construction the delivery path from publisher to subscriber. The LIDs of the links belonging to the constructed path are encoded into a Bloom filter and sent to the publisher. The fraction of ones in this filter is called fill factor, ρ. In each network, a maximum value, ρ_m, of fill factor is allowed and hence any packet with a Bloom filter of ρ, where $\rho > \rho_m$, is automatically dropped. This is required because otherwise an attacker could simply create a packet of all one values and launch a DDoS attack or send unwanted traffic to every user in the network [25, 24]. A Bloom filter with a maximum fill factor is called maximally filled and hence no more LIDs can be inserted. In [17] the Bloom filter is called a zFilter and we use this term throughout the paper.

An example of the LIPSIN packet forwarding is presented in Fig. 1. Consider one publisher (Pub-A), two subscribers (Sub-A and Sub-B) and three forwarding nodes (FW-1, FW-2, and FW-3), connected as shown in Fig. 1. Each link is given an m-bit LID. For illustrational purposes we use m = 8. Our aim is to construct the zFilter for the path from Pub-A to Sub-A. This is performed by OR-ing the LIDs of the links belonging to the delivery path. In this example, the constructed zFilter includes LID-1, LID-2, and LID-4, as shown in the figure. When the zFilter is ready, the TM sends it to Pub-A. The latter will insert the zFilter in the packet header and will send it to its attached forwarding node (FW-1). FW-1 will perform the forwarding test on each of its outgoing links. The test is performed by AND-ing the zFilter with the LID of the link. If, and only if, the result is equal to the LID, the packet is forwarded via this link. As shown in the figure, the packet will be forwarded to FW-2 and then to Sub-A, whereas the forwarding test will fail for LID-3 and the packet will not reach FW-3.

B. False positives in LIPSIN

One of the limitations of the Bloom filter based forwarding is the probability of false positives. That is, there is a chance that the packet will be forwarded via a link that was not included during the Bloom filter creation. For example, assume that LID-3 = 00110000 in Fig. 1. Since this link is not part of the delivery path from Pub-A to Sub-A, the created zFilter is still as shown in Fig. 1. However, the forwarding test at FW-1 will give a positive result for LID-3, and therefore a copy of the packet will be forwarded also to FW-3.

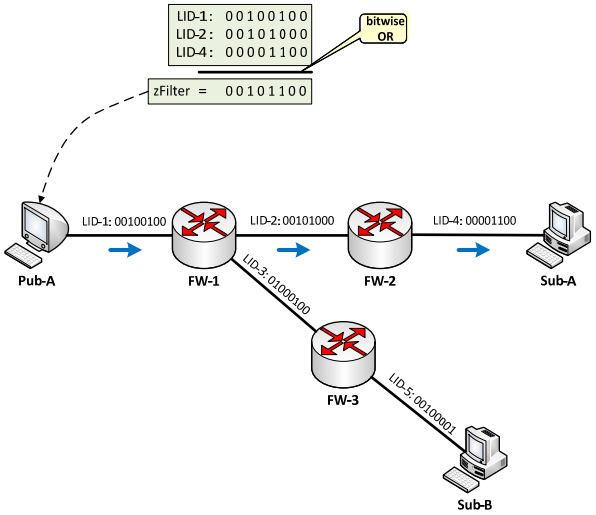

Figure 1. Example of LIPSIN forwarding.

One obvious drawback of the false positives is that some packets are unnecessarily duplicated over some links, thus causing the waste of communication and processing resources. In addition, as shown in [14, 20], false positives can be exploited by malicious nodes for launching brute-force attacks in the network. Also other types of attacks, such as packet storms and forwarding loops are possible [25].

The probability of false positives f can be calculated if the fill factor ρ of a zFilter is known [26, 27]:

$$f = \rho^k \quad (2)$$

A number of proposals have been introduced to mitigate the effect of false positives [14, 22, 28]. In [14] the link ID tag (LIT) mechanism is proposed. According to this approach, d different zFilters are created and evaluated in terms of false positives rate. Next, the best performing zFilter is selected and its index is included in the packet header.

C. Security vulnerabilities in LIPSIN

As discussed earlier, the LIPSIN mechanism is vulnerable to brute-force attacks. These attacks are launched by trying all possible zFilters until one of them causes false positives and then will pass the forwarding tests along the desired path. Consider a network with m = 256 bits, maximum fill factor ρ_m = 0.5, and the number of hash functions k = 5. Below we investigate two possible scenarios of brute-force attacks. In both scenarios we assume that the attacker has obtained a valid zFilter, z, representing the path between the source node and the target node.

The first attack strategy depends on whether the zFilter z is maximally filled or not. If it is not, that is $\rho < \rho_m$, then the attack could convert some 0 bits into 1. The new zFilter, z_i, will be sent by the attacker to the network and may cause some false positives over the links from the attacker to the valid path. To increase the chance of success, the attacker may try all possible combinations of maximally filled z_i that includes z.

The second attack strategy can be used when the attacker has no any existing valid zFilter, or the existing zFilter is maximally filled. In that case the attacker is generating random zFilters with the hope that some of them will reach the target node due to false positives. If packet acknowledgment is supported by the application,

then the attacker will know that the randomly generated zFilter can reach the target.

In addition to brute-force attacks, the LIPSIN forwarding mechanism is also vulnerable to replay attacks and computational attacks. A replay attack is when the attacker uses a previously created valid zFilter for sending non-requested traffic, this type of attack is of course possible, and used, in existing IP networks. A computational attack is launched by collecting a number of valid zFilters and analysing the correlation between their bit patterns. However, these types of attacks are not covered in this paper.

D. Security enhancements to the forwarding plane

Many of the LIPSIN security vulnerabilities, discussed in the previous subsection, exist due to the fixed LIDs used in the network. That is, when a link is assigned a LID, it keeps this LID forever. To overcome these limitations, the Z-formation approach is proposed in [17]. The main idea is to make zFilters expire after a certain time by periodically changing the LIDs. As a consequence, the forwarding decisions become dynamic and depend on the packet contents, zFilter, and processing context. Also a time-bound shared key between the TM and FWs is used [29].

To compute the LIT of each link, a cryptographically secure hash function, called Z-formation, is used. This function takes four input parameters and produces an *m*-bit LIT. The input parameters are the following:

- Some in-packet content-based information, *I* (e.g. flow ID or RID)
- Periodically changing time-based secret key, *K(t)*.
- Port numbers of incoming/outgoing interfaces, In and Out, respectively.
- The optimization index, *d*, for mitigating false positives.

That is, *LIT = Z(I, K(t), In, Out)* . Hence, the LITs become dynamic and bound to a specific flow, path, and time period. The key *K* is used when creating the corresponding LITs of the path. Therefore, the Z-formation approach essentially assigns a finite lifetime to zFliters. As a consequence, it provides better protection against replay and computational attacks. However, it does not have any impact on the brute-force attacks. Therefore, in the following we propose an enhancement of the Z-formation technique, which performs well in protecting against brute-force attacks.

III. BRUTE FORCE ATTACK PROBABILTY

As discussed in Section II.C, the brute force attack is launched by causing some false positives over a path. Hence, this type of attack depends on the false positives probability of (2). Therefore, to attack a target h hops away, the attacker would use a random maximally filled zFilter and the probability of this attack is given by:

$$P_a = \rho_m^{k.h} \tag{3}$$

It is also possible to find the number of expected attack attempts to reach a target and this can be calculated by:

$$x = log_{1-P_a}(1 - p_r) \tag{4}$$

where p_r is the probability of obtaining at least one zFilter after x attempts. Full calculations of attack probability and number of expected attempts can be found in [20].

In [20], it has been demonstrated that, with a given scenario used in [17], an attacker is able to reach a victim 5 hops away in approximately 23s, using brute force attack with 50% probability of success. This scenario requires having an attacking node capability of 10^6 packets/s which we call attacking rate a_r, and also using a zFilter with the parameters $\rho_m = 0.5$, $k = 5$ and $m = 256$-bit. Therefore, the authors have proposed a solution that gives some improvements of reducing the attack probability and subsequently increasing the time required for the attacker to successfully guess a valid zFilter with a certain probability p_r. This required time is called *safe window,* and the aim is to make the safe window as long as possible. To achieve a longer safe window would require using a lower value of the fill factor ρ when creating the zFilter. However, this restricts the number of 1s to be inserted into the zFilter and subsequently a limited number of LIDs can be supported. This is a negative impact as the lengths of supported delivery paths become shorter.

To mitigate this issue, the authors of [20] have suggested using the parameters $\rho_m = 0.48$, $k = 6$ and $m = 256$-bit as optimal values in terms of making a balance between security and scalability perspectives. These parameters support a delivery path containing up to 24 LIDs, and in the same time give a safe window of approximately 60 min. This is considered a significant improvement in terms of the safe window which makes the brute force attack much harder. In the same study, it also has been suggested that LIDs are changed every $\Delta t = 40$ min using Z-formation technique. More details of updating LIDs can be found in [21].

However, here we show that it is more likely that an attack can be successful within Δt of 40 min. In practice, when using the same suggested parameters of k, m, ρ_m and same assumed attacking rate, a_r, the probability of successful attack p_r within 40 min is calculated from (3) and (4), and as follows:

$$x = 40 \times 60 \times 10^6 = 2.4 \times 10^9 \text{ attempts.}$$

$$P_a = \rho_m^{k.h} = 0.48^{5\times6} = 0.00535$$

$$p_r = 1 - (1 - P_a)^x = 0.4815$$

This probability is considered unacceptable, and this is because an attacker could reach a victim before the coming LIDs update takes place.

From the above, it can be seen that there is still a high possibility of successfully launching brute force attacks. Changing the LIDs more frequently, for instant every 10 min or reducing ρ_m further, may help decreasing this attack probability within a certain Δt. However, this would increase the load on the network entities as well as the traffic overhead when updating LIDs very frequently. Moreover, reducing the maximum fill factor may help mitigating the attack but this would negatively impact the network scalability. Recall that the results in [20] can support up to 24 LIDs, and it has been demonstrated that such size of delivery paths can perform well in small and medium realistic network topologies. However, it fails to support multicast communications in relatively large networks (e.g. a network size of |V|= 113, |E|= 183), hence reducing the fill factor further is not an efficient option.

Therefore, a possible solution is to increase the current size of the zFilter, m = 256-bit, used in the literature. This is because having larger size of zFilters allows a higher

number of LIDs to be inserted and also keeps the maximum fill factor as small as possible. This would improve the security in the network by significantly decreasing the probability of launching brute force attacks within a certain period.

In this work, we analyse the impact of the parameter *m* on attack probability, safe window and also the network scalability in terms of maximum number of supported LIDs. We also compare the scalability performance of the forwarding plane when using larger *m* with the current used one. For each size of *m*, we also investigate the effect on the traffic overhead. This is because using larger sizes of zFilter requires more space to be dedicated in the packet header, which increase the traffic utilization.

IV. BRUTE FORCE ATTACK ANALYSIS

A. Simulation Setup

When comparing zFilters of different sizes, one of the aspects is to investigate the capacity of maximum allowed fill factor. This is done by observing how many LIDs, *n,* the zFilter can support. We experimentally investigate this by running a simulation with 2×10^6 iterations. In each run we select the parameters *m*, *k,* and *n*, and in each iteration we create a zFilter *z*. Then we observe the number of unique positions of 1s, *s,* which tells us how many 1s are included in *z*. Then we calculate ρ_m according to the following:

$$\rho_m = \frac{s}{m} \tag{5}$$

After running the experiment, we record the maximum value of ρ_m found among the results and consider it as the maximum allowed fill factor for the given parameters. The experiment is performed for different sizes of *m* while keeping the other parameters fixed.

Another important factor to look at is the impact of the delivery path size on the safe window when using different values of *m* and *k*. According to the results of ρ_m and *n* obtained from the experiment, and using the same attack scenario explained in Section III, we calculate the number of the attack attempts, *x*, using (4). Then, with the attacking rate a_r which in this paper is assumed to be 10^6 packet/s, we find the safe window by:

$$Safe\ Window = \frac{x}{a_r} \tag{6}$$

B. Results

Figure 2 shows the maximum capacity of ρ_m for different values of *n*. Each curve represents a different size of zFilter. We consider the following sizes: *m* = {256, 384, 512, 768}. Generally, it can be noticed that when using larger zFilters, a smaller value of ρ_m can support longer delivery paths containing *n* LIDs. For example to support a delivery path of 23 LIDs with the case of using a zFilter of *m* = 256-bit, the maximum fill factor ρ_m has to be at least 0.429, whereas for same *n* but with larger *m* of 512-bit, we can support the same path with smaller value of ρ_m and is equal to 0.224. Recall that small values of ρ_m are desirable to mitigate brute force attacks. Also it can be seen that as *n* increases, the difference of ρ_m between each *m* increases.

In Fig. 3 we present the safe window for different *n.* Note that the safe window scale is logarithmic to allow fitting the results into one plot, so the difference between the results seems less, at first site, than it actually is. The results are presented for two sizes of the zFilter *m* = {256,

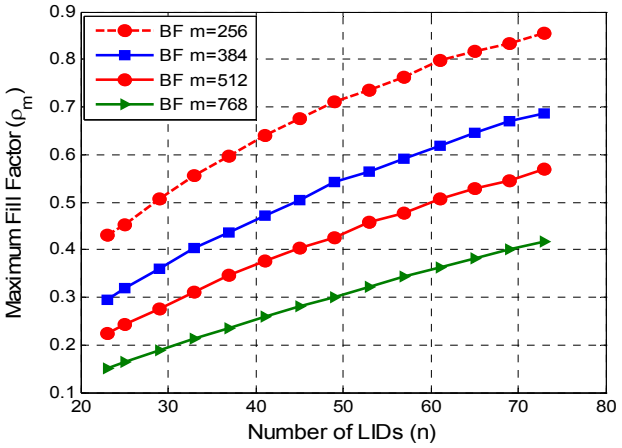

Figure 2. Impact of increasing the number of LIDs on the maximum fill factor for different size of *m and k* = 5.

512} and three different numbers of hash functions, *k* = {4, 5, 6}. Overall the safe window decreases as *n* increases for both sizes of *m*. Also it can be seen that when using *m* = 512-bit, the safe window becomes in the magnitude of approximately 10^6 longer than that when *m* = 256-bit. For instance, when using a zFilter of (*m*=256, *k*=5 and *n*=23), this gives a safe window of approximately 10^3s whereas in a zFilter with same parameters but larger *m* of 512-bit the safe window increases to approximately 6×10^9s. This is because in the larger zFilter, 23 LIDs can be accommodated with a smaller ρ_m, and this makes the attack time longer to be achieved.

One advantage of using a larger zFilter is the ability to achieve a longer delivery path than possible with a small zFilter while keeping the safe window almost same. For example, in the figure, a safe window of approximately 10^3s can be gained when using a zFilter of (*m*=256, *k*=6, *n*=23), whereas it also can be obtained with a larger zFilter of (*m*=512, *k*=6, *n*=49). This is a significant improvement of almost a double in the number of LIDs to be inserted in the zFilter. This is applied to the other sizes of *m.*

Therefore, a network setup with the optimal parameters as suggested in [20], which uses a 256 bits zFilter and *k* = 6 can support up to 24 LIDs for a safe window of approximately 60 min. However, a network that uses 512-bit zFilters and same parameters of *k* and *n*, can resist up to 76×10^7 min before a successful attack. Moreover, in the later network if we change the LIDs every 40 min then the

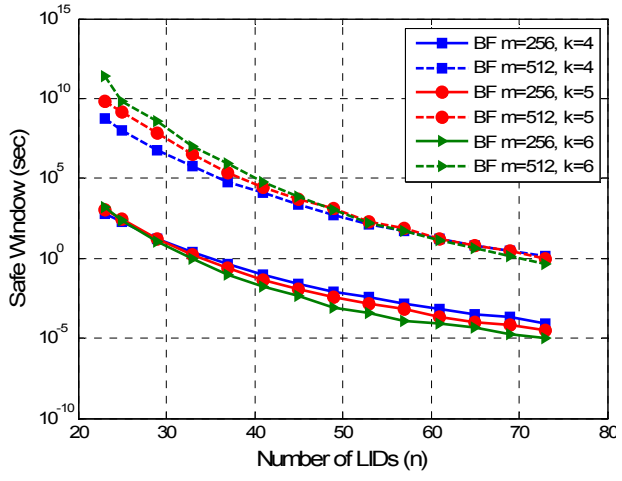

Figure 3. Impact of increasing *n* on the safe window for different *m* and *k* with number of hops (*h*) = 5 and p_r = 0.5.

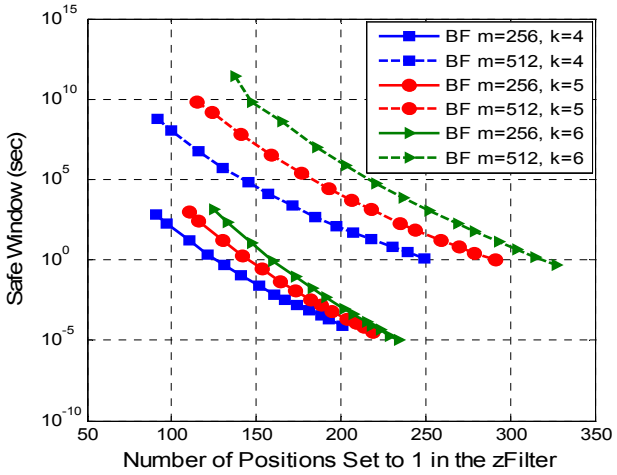

Figure 4. Impact of s on the safe window for different m, k, when attacking a victim 5-hop away with probability (p_r) of 0.5.

probability p_r that the attacker gets succeed within this period is 0.001 whereas it is 0.481 in the network using 256-bit zFilters. This significant reduction in the attack probability helps to discover an attack in progress by observing the number of packets dropped at ingress due to non-matching zFilters.

In Fig. 4, we show the relation between the number of positions set to 1 in the zFilter, s, and the safe window for two sizes of zFilter m = {256, 512} and each with k = {4, 5, 6}. Generally, it can be noticed that as the number of 1s increase the safe window decreases. This is because having more 1s increases the probability of false positives which helps the attacker to reach a victim in short time. For example, using a large zFilter of 512 bits, we can insert more 1s and subsequently accommodate more LIDs with an acceptable safe window. This can be seen from the figure when using a zFilter with the parameters (m = 256-bit, k = 5, s = 142), the safe window is approximately 2 min whereas for zFilter of (m = 512-bit, k = 5, s = 141) the safe window is around 7×10^6 min.

In Fig. 5, we present the probability of a successful attack within 40 min, against the maximum fill factor. We show how ρ_m and p_r may be traded off against each other to achieve desired network design goals. The results are based on the attack scenario assumed in this paper (i.e. an attacker is 5 hops away from his victim and the attacking rate is 10^6 packets/s). For instance, to have an attack probability as low as 1.2×10^{-9} using any size of zFilter with k = 7, ρ_m should not exceed 0.3. We can also see that a zFilter with k = 4 is not secure to be used when Δt = 40 min, especially when $\rho_m \geq 35$ as p_r becomes almost 1.

Using a large zFilter does not only improve the network security, but it also keeps the false positives rate, f, sufficiently low. This can be seen in Fig. 6, where the influence of increasing the number of LIDs on the false positives is presented. In this figure, we show the average false positives rate when using different sizes of zFilter m = {256, 512} with k = {4, 5, 6} and h = 5 using (1). Generally it can be noticed that when m is 512-bit, the false positive rate is significantly lower than when m is 256-bit. Also in this figure, while the performance of f is almost the same when using m of 256-bit for all values of k, it gives the best performance when using the parameters m = 512 bits and k = 6, and especially for higher n.

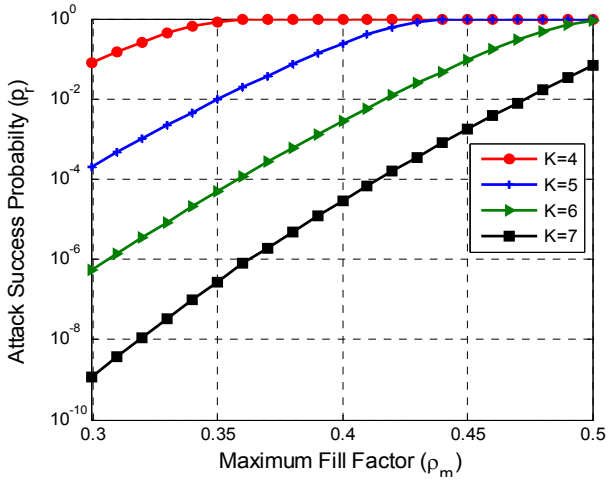

Figure 5. Impact of increasing the fill factor (ρ_m) on attack probability to reach a victim 5-hop away using any size of m.

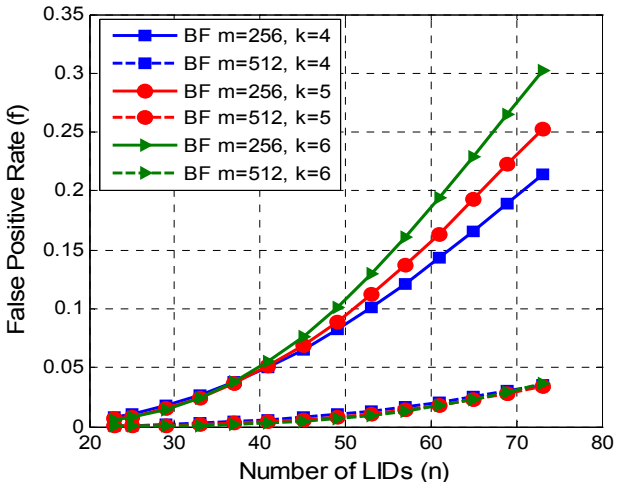

Figure 6. False positives rate (f) as function of the number of LIDs (n), for 5 subsequent hops.

C. Discussion

One of the negative impacts of using large size zFilters (e.g. m = 512), is the introduction of extra traffic in the network. Therefore, in this section we compare the total network utilisation, including the actual useful data sent and traffic overhead, for two different sizes of zFilter, m = {256, 512}. To do so, we calculate the header size of the ICN packet assuming basic fields are used as shown in Fig. 7. Using larger zFilters comes at the expense of actual data required to send per packet. This is because the current maximum transmission unit is 1500 bytes including data, zFilters and the other fields. Therefore, when using zFilters with sizes of 256 bits and 512 bits, the maximum actual data to be included in the payload are 1404 and 1372 bytes, respectively. The traffic overhead when using a specific zFilter size depends on the distribution of Internet packets size. Therefore, we use an Internet traffic trace collected on March 2014, and containing 91×10^6 packets [30]. For each packet, we deduct the zFilter size from the packet length in order to find the useful utilised data. Using the traffic distribution from this data set we find that there is a total overhead of 31.2% when using a 256-bit zFilter and 38.3% when using a 512-bit zFilter. For the equivalent IP based transport there is 14.2% and 21.3% overhead for IPv4 and IPv6, re-

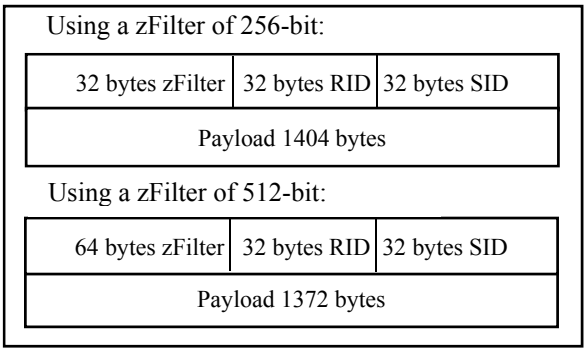

Figure 7. Basic fields on ICN packet header

spectively. While the larger 512-bit zFilter has a small, but significant, increase in overhead it comes with the advantage that it provides good mitigation against brute force attacks and with reduced false positive rate. With the use of the zFilter security mechanism we have an increase in the overhead compared to IP. However, this has the significant advantage that it is very difficult (or unlikely) for an attacker to send unauthorised traffic to an arbitrary end host, effectively stopping attacks such as DoS which are common in IP networks.

V. CONCLUSION

In this paper, we analyse the LIPSIN ICN forwarding mechanism and reveal its security vulnerabilities. In particular, we reveal that the Bloom-filter forwarding approach used in LIPSIN can be used for launching brute-force attacks to end users and network infrastructure. We study the impact of various LIPSIN parameters, such as Bloom filter size and maximum fill factor, from the security viewpoint. Next, we propose efficient solutions that increase the safe window and decrease the attack probability. Our proposed security enhancements are verified by simulation studies.

REFERENCES

[1] A. Feldmann, "Internet clean-slate design: what and why?", ACM SIGCOMM Computer Communication Review, vol. 37, pp. 59-64, 2007.

[2] M. Handley, "Why the Internet only just works", BT Technology Journal, vol. 24, pp. 119-129, 2006.

[3] J. Choi, J. Han, E. Cho, T. Kwon, and Y. Choi, "A survey on content-oriented networking for efficient content delivery", IEE Comm Mag, vol. 49, pp. 121-127, 2011.

[4] G. Xylomenos, et al., "A survey of information-centric networking research", IEEE Comm Surveys & Tutorials, no. 99, pp. 1-26, July 2013.

[5] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking", IEEE Comm Mag, vol. 50, pp. 26-36, 2012.

[6] T. Koponen, et al., "A data-oriented (and beyond) network architecture", in Conference of Applications, Technologies, Architectures, and Protocols for Computer Communications, pp. 181-192, Kyoto, Japan, 2007.

[7] V. Jacobson, D. K. Smetters, J. D. Thornton, M. Plass, N. Briggs, and R. Braynard, "Networking named content", Communications of the ACM, vol. 55, pp. 117-124, 2012.

[8] SAIL: Scalable and Adaptive Internet Solutions. Available at: www.sail-project.eu, accessed on (20-02-2014).

[9] COntent Mediator architecture for content-aware nETworks (COMET). Available at: http://www.comet-project.org/, accessed on (20-02-2014).

[10] N. Fotiou, P. Nikander, D. Trossen, and G. C. Polyzos, "Developing information networking further: From PSIRP to PURSUIT", in The 7th International ICST Conference on Broadband Communications, Networks and Systems, Athens, Greece, Oct. 2010.

[11] P. T. Eugster, P. A. Felber, R. Guerraoui, and A. M. Kermarrec, "The many faces of publish/subscribe", ACM Computing Surveys (CSUR), vol. 35, pp. 114-131, 2003.

[12] P. Nikander and G. Marias, "Towards understanding pure publish/subscribe cryptographic protocols", Security Protocols XVI, pp. 144-155, 2011.

[13] B. Gajic, J. Riihijarvi, and P. Mahonen, "Intra-domain topology manager for publish-subscribe networks", in 18th International Conference on Telecommunications (ICT), pp. 394-399, 2011.

[14] P. Jokela, A. Zahemszky, C. E. Rothenberg, S. Arianfar, and P. Nikander, "LIPSIN: line speed publish/subscribe inter-networking", in ACM SIGCOMM Conference on Data Communication, Barcelona, Spain, 2009.

[15] P. Gasti, G. Tsudik, E. Uzun, and Z. Lixia, "DoS and DDoS in Named Data Networking", in 22nd International Conference of Computer Communications and Networks (ICCCN), 2013.

[16] C. Seungoh, K. Kwangsoo, K. Seongmin, and R. Byeong-Hee, "Threat of DoS by interest flooding attack in content-centric networking", in International Conference on Information Networking (ICOIN), pp. 315-319, 2013.

[17] C. E. Rothenberg, P. Jokela, P. Nikander, M. Sarela, and J. Ylitalo, "Self-Routing Denial-of-Service Resistant Capabilities Using In-packet Bloom Filters", in European Conference of Computer Network Defense (EC2ND), pp. 46-51, 2009.

[18] B. Alzahrani, M. Reed, and V. Vassilakis, "Enabling z-Filter updates for self-routing denial-of-service resistant capabilities", in 4th Computer Science and Electronic Engineering Conference (CEEC), 2012, Colchester, U.K., pp. 100-105, Sept. 2012.

[19] B. Alzahrani, V. Vassilakis, and M. Reed, "Securing the Forwarding Plane in Information Centric Networks", in 5th Computer Science and Electronic Engineering Conference (CEEC), Colchester, U.K., Sept. 2013.

[20] B. Alzahrani, V. Vassilakis, and M. Reed, "Mitigating brute-force attacks on Bloom-filter based forwarding", in Conference on Future Internet Communications (CFIC),, pp. 1-7, 2013.

[21] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," Communications of the ACM, vol. 13, pp. 422-426, 1970.

[22] L. Carrea, A. Vernitski, and M. Reed, "Optimized hash for network path encoding with minimized false positives," Computer Networks, vol. 58, pp. 180-191, 2014.

[23] N. Fotiou, G. F. Marias, and G. C. Polyzos, "Fighting spam in publish/subscribe networks using information ranking", in 6th EURO-NF Conference of Next Generation Internet (NGI), 2010.

[24] M. Särelä, C. Rothenberg, T. Aura, A. Zahemszky, P. Nikander, and J. Ott, "BloomCasting: security in Bloom filter based multicast", ed: Springer, NordSec, Lecture Notes in Computer Science, 2010.

[25] M. Särelä, C. Rothenberg, T. Aura, A. Zahemszky, P. Nikander, and J. Ott, "Forwarding anomalies in Bloom filter-based multicast", in INFOCOM, IEEE, pp. 2399-2407, 2011.

[26] S. Lumetta and M. Mitzenmacher, "Using the power of two choices to improve Bloom filters", Internet Mathematics, vol. 4, pp. 17-33, 2007.

[27] F. Hao, M. Kodialam, and T. Lakshman, "Building high accuracy bloom filters using partitioned hashing", in ACM SIGMETRICS Performance Evaluation Review, pp. 277-288, 2007.

[28] J. Tapolcai, A. Gulyas, Z. Heszbergery, J. Biro, P. Babarczi, and D. Trossen, "Stateless multi-stage dissemination of information: Source routing revisited", in IEEE Global Communications Conference (GLOBECOM), pp. 2797-2802, 2012.

[29] B. Alzahrani, V. Vassilakis, and M. Reed, "Key management in information centric networking," International Journal of Computer Networks & Communications (IJCNC), vol. 5, no. 6. pp. 163-166, Nov. 2013.

[30] Measurement and Analysis on the WIDE Internet (MAWI). Available at: http://www.mawi.wide.ad.jp/mawi/samplepoint-F/2014/201403091400.html, accessed on (20-02-2014).