BLAM: Lightweight Bloom-filter based DDoS Mitigation for Information-Centric IoT

Gang Liu*, Wei Quan*, Nan Cheng†, Bohao Feng*, Hongke Zhang*, Xuemin (Sherman) Shen‡
*School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China
†School of Telecommunication Engineering, Xidian University, Xi'an, China
‡Electrical and Computer Engineering, University of Waterloo, Waterloo N2L3G1, Canada
Email:{gangliu, weiquan, bhfeng, hkzhang}@bjtu.edu.cn; {n5cheng, sshen}@uwaterloo.ca

***Abstract*—Information-Centric Networking (ICN) provides great potential to promote the development of the Internet of Things (IoT) due to its multicast nature and mobility support. However, the stateful forwarding peculiarity introduces new varietal attacks named Interest Flooding Attacks (IFA), which is stealthy but destructive for the resource-limited IoT devices. In this paper, we propose a lightweight BLoom-filter based Attack Mitigating (BLAM) mechanism to reduce the detecting memory cost, while guaranteeing both the detecting accuracy and delay. Specifically, each IoT node employs a small Bloom filter to check attack behaviors instead of the traditional memory-consuming operations, *i.e.*, recording malicious requests. Bloom filter values by hashing the published data names with a set of hash functions, are encapsulated and distributed via a new message named Ba-NACK. Based on this design, two specific schemes are further proposed for the attack detecting and Bloom filter updating. We formulate the memory cost minimum problem and theoretically analyze that BLAM can reduce the memory cost. We also implement BLAM in a realistic network testbed to evaluate its performance. The results show that BLAM reduces the memory cost by 78.6%, and reduces the delay from millisecond to microsecond with slight sacrifice of the accuracy by 0.4% compared with other state-of-the-art mechanisms.**

I. Introduction

Due to the evolving of the Internet of Things (IoT), billions of connected "things" can interact with each other in the world, including the air-ground interactions [1], [2]. As Gartner forecasts, this number will reach 20 billion by 2020 [3]. In such a magnificent IoT ecosystem, security becomes one of the biggest concerns, which directly affect the full potential of the IoT promise [4]–[7]. Recently, a new wave of research starts using the promising Information-Centric Networking (ICN) paradigm in IoT, because ICN solutions focus on naming data, rather than endpoints, which can greatly simplify IoT applications and improve data delivery efficiency [8]. It is also stated that ICN can mitigate traditional Distributed Denial-of-Service (DDoS) attacks for the certain data providers. Concretely, Amadeo *et al.* [9] proposed that in-network caching in ICN can greatly avoid DDoS attacks, and name-based forwarding in ICN can trace the attackers easily. Baccelli *et al.* [10] made an experimental comparison between information-centric IoT solutions and IP-based ones (*e.g.*, 6LoWPAN), which shows that information-centric IoT is more effective to solve the security problems.

Unfortunately, ICN brings a new varietal DDoS attack called Interest Flooding Attacks (IFA), which has become a big threat for information-centric IoT [11], [12]. Typically, attackers issue a large number of fake *Interest* messages to request non-existing *Data*, which can lead to the memory overflow for the ICN-IoT nodes. Recently, many mechanisms have been proposed to mitigate the IFA attacks [13], [14]. However, most of state-of-the-art mechanisms are too heavy to suit for the tiny and resource-constrained IoT devices, even if they can, the performance is not prominent in the IoT domain. For example, Wang *et al.* [15] proposed a mechanism to maintain a table to record the malicious *Interest* name. However, if attackers issue malicious *Interest* with various content names, the table size will increase exponentially. It must cost too much memory to be used in the IoT scenarios, where devices usually have triple-limited resources, memory, compute, and power capacities [16]. Therefore, a *lightweight* IFA mitigating solution is highly pursued for information-centric IoT.

Inspired by the previous works on simple yet efficient name lookup, named TB2F and NLAPB [17], [18], we propose a lightweight BLoom-filter-based Attack Mitigating (BLAM) mechanism for IoT-tailored IFA mitigating solution. Such a solution aims to effectively reduce the memory cost while guaranteeing the detecting accuracy and delay. As far as we know, this work is the first one to study how to adopt the bloom-filter to mitigate the IFA in IoT scenario.

Particularly, BLAM enables each IoT node to access a local Bloom filter array to check whether the incoming *Interest* is matched. If the *Interest* is matched, it continues the following memory-consuming process. This Bloom filter array is lightweight and dynamically generated by a data producer which hashes each published data name using hash functions. Besides, each IoT node synchronizes local Bloom filter array with the palingenetic array reactively according to a specific Ba-NACK packet. When a node receives the Ba-NACK message, it first extracts Bloom filter array field from the packet, then updates the local Bloom filter array, finally forwards this packet to other IoT nodes. Based on this lightweight process, BLAM is able to reduce the memory cost while guaranteeing the detecting accuracy and delay. The main contributions of this paper can be summarized as follows.

1) We firstly analyze three main factors, accuracy, delay, and memory cost, on detecting IFA in Information-centric IoT. Focusing on these factors, we formulate the IFA mitigation as a memory cost minimum problem

978-1-5386-4727-1/18/$31.00 ©2018 IEEE

while guaranteeing the accuracy and delay.

2) We propose a lightweight BLAM solution, which employs a small Bloom filter to check the attack behaviors instead of the traditional memory-consuming operations. The details are introduced for the attack detection and the Bloom filter update operations in BLAM. With high query rate and low memory cost, BLAM can avoid the overflow memory for the information-centric IoT nodes.
3) We implement BLAM in a realistic network testbed and evaluate its performance in terms of memory cost, delay and accuracy. Experimental results show that BLAM can reduce the memory cost by 78.6%, and the delay from millisecond to microsecond with slight sacrifice of the accuracy by 0.4% compared with other state-of-the-art mechanisms.

The remainder of this paper is organized as follows. Section II presents the related works. Section III analyzes three detecting factors and describes the BLAM details comprehensively. Section IV evaluates the performance of BLAM mechanism. Finally, we conclude this work in Section V.

II. Related Works

Early works on IFA mostly focus on detecting the attacks accurately. Wang *et al.* [15] proposed a threshold-based detection mechanism which maintains a table to record malicious *Interest* name to mitigate the attacks. Salah *et al.* [19] designed three coordinating components to resist the attacks. However, the two mechanisms are difficult to be integrated on constrained IoT devices because both of them require to maintain a heavyweight table.

Recently, Xin *et al.* [14] proposed a cumulative-entropy based mechanism to detect IFA and used an *Interest* traceback countermeasure to restrain the attacker after detection. Moreover, Zhi *et al.* [13] proposed a Gini-impurity-based detecting mechanism, which enables the routers to distinguish the malicious *Interest* with legitimate one. However, these two mechanisms need more time to statistics the distribution of requested *Interest* names, they have a long-term detecting delay and is not suitable for the IoT scenarios. The delay, which is an important factor in detecting IFA, is found by our previous work [20]. Compared with these works, this paper aims to propose a lightweight IoT-specific attack mitigating mechanism.

On the other hand, Bloom filters have been widely adopted in the ICN domain to solve many problems. Li *et al.* [21] presented a novel data forwarding processes using mapping Bloom filters to minimize the on-chip memory consumption. Guo *et al.* [22] proposed a probabilistic-counting and bloom-filter based algorithm to mitigate false-locality pollution attack. Besides, our previous work proposed an efficient name lookup engine using Bloom filters [17], [18]. Marandi *et al.* [23] proposed a novel Bloom filter based Routing (BFR) scheme for content discovery. However, we have not seen how the Bloom filter is used to mitigate IFA in the IoT domain.

Based on the above analysis, this paper firstly utilizes Bloom filter to mitigate IFA in the IoT domain, which aims to cost a little memory and ease IoT devices with triple-limited resources. We propose a lightweight Bloom filter based IoT-specific attack mitigating mechanism to reduce the memory cost while guaranteeing the detecting accuracy and delay.

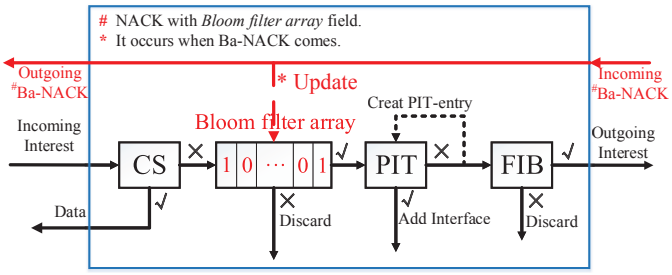

Fig. 1: BLAM forwarding process

III. System Model and BLAM Solution

This section first introduces three detecting factors in the IoT environments comprehensively and formulates the problem. To solve the problem, this section introduces the details of the BLAM mechanism and gives some theoretical analysis on BLAM advantages.

A. Modeling and Problem Formulation

On the one hand, state-of-the-art mechanisms can detect IFA attacks with high accuracy. The detecting accuracy is an important factor in the IoT environment and is quantized with FPR (denoted as p_{FP}) and FNR (denoted as p_{FN}). Concretely, FPR means that a legitimate *Interest* is considered as malicious one, while FNR means that a malicious *Interest* is considered as legitimate one. On the other hand, state-of-the-art mechanisms neglect the detecting delay and suffer from long-term memory occupation. The detecting delay (denoted as τ) is a hidden and important factor in the IoT domain, it is first introduced in our previous work [20]. Besides, owing to the triple-limited resources of IoT devices, the memory cost (denoted as C_m) becomes a significant important factor in detecting the attacks. To ease IoT devices with limited memory resources, we formulate the memory cost using the other two factors. Typically, in the period of Δt, the detecting memory cost includes two parts. First, the size of a malicious *Interest* table which consists of many unsatisfied *Interest* name; Second, the size of pending *Interest* table occupied by malicious *Interest* (*i.e.*, malicious *Interest* occupy pending *Interest* table due to the false negative). Therefore, in the period of Δt, the detecting memory cost can be denoted as the following equation:

$$C_m = \mathcal{N}(\mathbb{I}_u)\,\Phi(MIT) + [\mathcal{N}(\mathbb{I}_m)/\Delta t - \mathcal{N}(\mathbb{I}_u)]\,\Phi(PIT) \quad (1)$$

where $\mathcal{N}(\cdot)$ denotes the number of elements in a set, $\mathbb{I}_u$ denotes a set of unsatisfied *Interest*, $\mathbb{I}_m$ denotes a set of all malicious *Interest* in the time scale T. Besides, $\Phi(\cdot)$ denotes the average memory cost of every element. Notably, $\mathbb{I}_u$ is in the time scale Δt. However, in the time scale T, it can be denoted as the following equation:

$$\begin{aligned}\mathcal{N}_T\left(\mathbb{I}_u\right) &= \int_0^T \mathcal{N}\left(\mathbb{I}_u\right)\ d\Delta t \\ &= \left(1-p_{FN}\right)\mathcal{N}\left(\mathbb{I}_m\right) + p_{FP}\mathcal{N}\left(\mathbb{I}_l\right)\end{aligned} \tag{2}$$

where $\mathbb{I}_l$ denotes a set of all legitimate *Interest* in the time scale T. That is, in the time scale T, the number of unsatisfied *Interest* includes two types. First, the number of malicious *Interest* without a false negative; Second, the number of legitimate *Interest* with a false positive. Typically, p_{FN}, p_{FP} and the number of legitimate *Interest* is small, thus $\mathcal{N}_T\left(\mathbb{I}_u\right)$ satisfied the following equation approximatively:

$$0 \leq \mathcal{N}_T\left(\mathbb{I}_u\right) \leq \mathcal{N}\left(\mathbb{I}_m\right) \tag{3}$$

Based on the analysis above, we can denote the problem as the following equations:

$$\min \int_0^T C_m\ d\Delta t \tag{4}$$

subject to:

$$\begin{cases} 0 \leq \mathcal{N}_T\left(\mathbb{I}_u\right) \leq \mathcal{N}\left(\mathbb{I}_m\right) \\ 0 < \Phi\left(MIT\right) \\ 0 \leq p_{FN}\ \leq\ 1 \\ 0 \leq p_{FP}\ \leq\ 1 \\ \tau \leq\ \bar{\tau} \end{cases}$$

where $\bar{\tau}$ is the detecting delay of the state-of-the-art mechanisms. Typically, $\Phi\left(PIT\right)$ is an unalterable constant which is determined by the ICN stack. Thus we can obtain the following solution:

$$\Phi\left(MIT\right) \to 0,\ \mathcal{N}_T\left(\mathbb{I}_u\right) = \mathcal{N}\left(\mathbb{I}_m\right) \tag{5}$$

That is, to minimize the memory cost, a lightweight mechanism need to have two characteristics. First, in the time scale T, the mechanism has to mark all malicious *Interest* as unsatisfied *Interest*; Second, the malicious *Interest* table should have a small size as far as possible. Inspired by our previous works, we find the Bloom filter owns such two characteristics. Therefore, to minimize the detecting memory cost in the IoT domain, we embed Bloom filter into BLAM mechanism which is introduced in the next subsection.

B. BLAM mechanism

1) Operating Principle: Fig.1 demonstrates BLAM forwarding process. In which cache storage (CS), pending Interest table (PIT), and forward information base (FIB) are the same as those in original Interest-based ICN. Besides, the Bloom filter array, which is an array consisted of N bits, indicants whether the incoming *Interest* could be satisfied. Such an array is generated dynamically by a data producer, and reactively forward to other IoT nodes. To this end, BLAM embeds a Ba-NACK packet which is inherited from the NACK packet. Concretely, after generating a Bloom filter array, the producer encapsulates the array into Ba-NACK, and forward Ba-NACK packet to other IoT nodes according to reverse path retrieve.

Algorithm 1: Detecting attacks
Input: *Class* Interest
Output: *Class* Interest or not
1. **procedure** Detect (Interest)
2. Name ⟸ Interest.GetName();
3. **if** Bloom_filter_query(Name) == True **then**
4. Interest.Outgoing() ⟸ True;
5. **else**
6. Interest.Discarding() ⟸ True;
7. **end if**
8. **end procedure**

Algorithm 2: Updating Bloom filter array
Input: *Class* Ba-NACK
Output: *Class* Ba-NACK
1. **procedure** Updating_Bloom_filter_array (Ba-NACK)
2. Bloom_filter_array = Ba-NACK.extract ();
3. Bloom_filter_array.updating(Bloom_filter_array);
4. Ba-NACK.Outgoing ⟸ True;
5. return Ba-NACK;
6. **end procedure**

Notably, the process of generating a Bloom filter array is described in sub-subsection 2).

Particularly, on the one hand, after receiving an *Interest*, each IoT node executes as follows to make its forwarding decisions. First, the IoT node searches its CS using the longest-name-prefix-match method. If matching successfully, the node sends the *Data* back to the *Interest* incoming interface. Otherwise, the node queries its Bloom filter array. If there is a matching element, the node continues to match with PIT and sequent process is the same as original Interest-based ICN. If it is not the case, the node discards the *Interest* because the *Interest* is malicious one. On the other hand, after receiving a *Data*, each IoT node acts the same as original Interest-based ICN. Besides, after receiving a Ba-NACK packet, the IoT node decapsulates Ba-NACK packet and extracts the Bloom filter array filed to update local Bloom filter array. Then the node forwards the Ba-NACK packet to other IoT nodes according to reverse path retrieve. **Algorithms 1 and 2** show the pseudo-code of detecting the attacks and updating the Bloom filter array respectively. With the algorithms, each IoT node recognizes the attackers quickly and locally because it has a local Bloom filter array which indicates whether the request can be responded.

2) Bloom Filter Array: A Bloom filter can be denoted as the set $\mathbb{B} = \{k, m, n\}$. In which, m represents a Bloom filter array of m bits; k represents the number of different hash functions which maps some set element to one of the m array positions; n represents the number of a set element (legitimate *Interest* name in BLAM mechanism) in Bloom filter.

Fig.2 depicts the Bloom filter operating process. In the case of adding an element, the Bloom filter first feeds the element to each of the k hash functions to get k array positions. Then, the Bloom filter sets the bits at all these positions to 1. Besides, in the case of querying for an element, the Bloom filter first feeds it to each of the k hash functions to get k array positions. Then, if any of the bits at these positions is 0, the element is definitely not in the set. Otherwise, the element is in the set. Particularly, in this paper, we set $k = 8, m = 2560, n = 221$

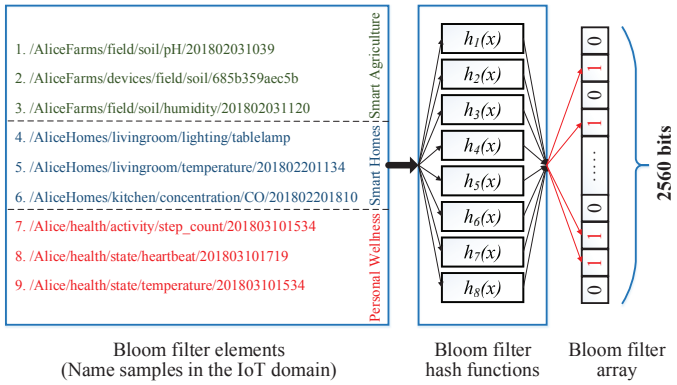

Fig. 2: Bloom filter operating process

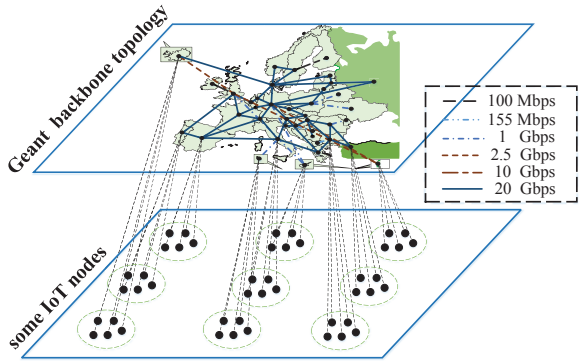

Fig. 3: The network topology

and the reason is described in sub-subsection 3).

3) Advantages of BLAM mechanism: In this sub-subsection, we analyze the performance of BLAM theoretically in terms of the detecting accuracy, delay and memory cost. First, there happens a false positive when querying an element. Theoretically, the FPR can be denoted as follows:

$$p_{FP} = \left(1 - e^{-kn/m}\right)^k \tag{6}$$

whereas $k = (m/n)ln2$, the value of p_{FP} can obtain the minimum. According to the reference [24], we set $k = 8, m = 2560, n = 221$ in the last sub-subsection, thus, we have,

$$p_{FP} = \left(1 - e^{-8*221/2560}\right)^8 = 3.83 * 10^{-3} \tag{7}$$

Obviously, the value of p_{FP} is very small and can be neglected. Moreover, the time complexity of our proposed mechanism is $O(k)$ because it needs to compute hash function k times. While the time complexity of other state-of-the-art mechanisms is $O(n)$ [13], [14]. Typically, $O(n)$ is larger than $O(k)$ because the number of hash functions is far less than the number of *Interest*. In other words, BLAM has less detecting delay compared other state-of-the-art mechanisms. Finally, we analyze the performance of BLAM mechanism in terms of memory cost. Concretely, BLAM has a Bloom filter array of 2560 bits (i.e. 320 Bytes), and meets the requirements of constrained IoT devices whose available RAM is in the order of 10 KBytes [16]. Therefore, we argue that BLAM costs less memory resources while guaranteeing the detecting accuracy and delay, which is efficient and lightweight to be applied to mitigate IFA on the IoT devices.

IV. Performance Evaluation

This section evaluates the performance of BLAM mechanism in the IoT domain. Particularly, we deploy the BLAM in a large-scale IoT topology, and execute many experiments in terms of the detecting accuracy, delay and memory cost. For comparison, other state-of-the-art mechanisms including DPE and NACK are also implemented.

A. Experimental Setup

The experiments are carried out in our smart identifier network platform, which is used to test new mechanisms for future Internet [25], [26]. This experimental platform has a controller node, a network node, a storage node and twelve compute nodes. Concretely, the controller node is responsible for managing all virtual network functions. The network node is responsible for delivering the flow/routing tables to manage the global virtual subnetwork. Besides, the storage and compute nodes are responsible for providing a resource pool for all virtual machines. Based on the platform, we construct the IoT topology (shown in Fig.3), in which each node is embedded with a paradigm ICN stack. Typically, the IoT network topology has 40 backbone nodes which connect 200 IoT nodes randomly. In addition, the topology has 259 links, the bandwidth of which vary from 100 Mbps to 20 Gbps. We randomly select 119 IoT nodes as legitimate nodes, 80 IoT nodes as malicious nodes and one IoT node as the data producer. Particularly, the legitimate nodes issue 20 *Interest* packets per second, with the prefixes like "AliceHomes/, AliceFarms/, Alice/". Different from the legitimate ones, the malicious nodes issue 200 *Interest* packets per second, with the prefixes like "BobHomes/, BobFarms/, Bob/". Notably, there are 200 legitimate and malicious prefixes respectively. The producer can only acknowledge the legitimate *Interest* and issue the *Data* packets whose size are around 4 Kilobytes. Moreover, each experimental duration is set to 60 seconds. To reduce the error, the number of experiments is set to 500.

B. Experimental Results

1) Memory Cost: Figs 4 and 5 depict the detecting memory cost of BLAM, NACK and DPE mechanisms on different detecting nodes, where we denote UK and ES as the United Kingdom and Spain respectively. Particularly, DPE consumes 15∼30 Kbytes memory resource because it needs to maintain a malicious *Interest* table. Such a table consists of many malicious *Interest* name which cannot be responded by any producer. The table is too large to be embedded in the IoT nodes, because the IoT nodes have constrained memory resource which is in the order of 10 Kbytes [16]. Furthermore, NACK has a lower memory cost (2∼4 Kbytes) in detecting the attacks, because NACK does not have any tables. It is

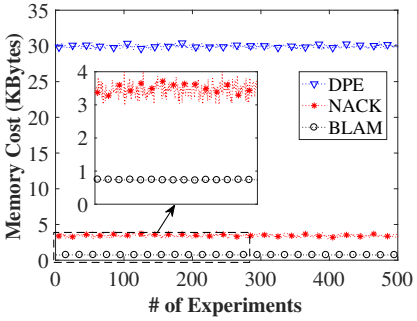

Fig. 4: the memory cost of node UK

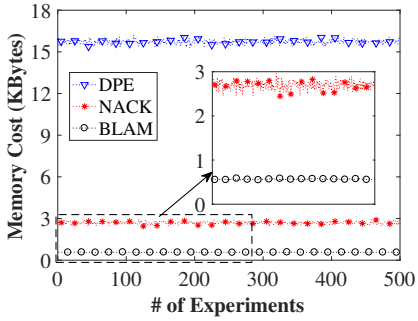

Fig. 5: the memory cost of node ES

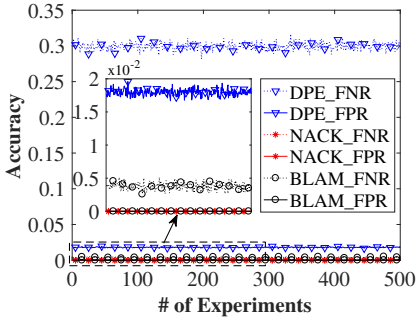

Fig. 6: the detecting accuracy of node UK

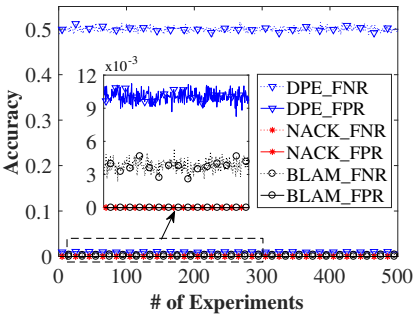

Fig. 7: the detecting accuracy of node ES

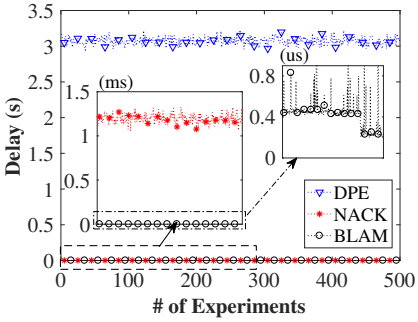

Fig. 8: the detecting delay of node UK

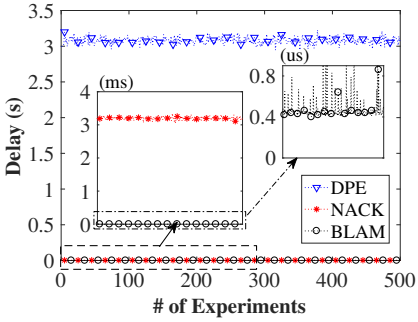

Fig. 9: the detecting delay of node ES

evident that NACK can meet the requirement of constrained devices in the IoT domain. However, because BLAM detects the attacks using a lightweight Bloom filter array whose size is only 2560 bits, it outperforms NACK mechanism. In other words, BLAM has the lowest memory cost (approximately 0.7 Kbytes, reduced by 78.6%) compared with other state-of-the-art mechanisms.

As shown in Figs 4 and 5, the memory cost of each node is significantly different using DPE and NACK mechanisms, while it is more stable using BLAM mechanism. Concretely, in terms of DPE mechanism, the node UK costs around 30 Kbytes memory resources while the node ES costs around 15 Kbytes. In terms of NACK mechanism, the node UK costs 3.5 Kbytes memory resources approximately while the node ES costs 2.6 Kbytes approximately. In terms of BLAM mechanism, both nodes UK and ES cost about 0.6 Kbytes memory resources. This phenomenon has two reasons. First, the malicious Interest table size and the number of NACK packets are proportional to the number of the *Interest*, while the Bloom filter array size does not have such a relationship; Second, node UK receives more malicious and legitimate *Interest* than node ES because node UK is more closer to the data producer in the IoT topology. Therefore, the more malicious *Interest* the attackers send, the more memory cost DPE and NACK have. However, BLAM has the lowest memory cost whatever the number of malicious *Interest*.

2) Accuracy: We can obtain the performance of accuracy in the Figs. 6 and 7. In terms of FPR, because BLAM and NACK mechanisms can process legitimate *Interest* normally as the origin ICN does, there is no FPR using the BLAM and NACK mechanisms. However, the FPR increases to 1∼2% using DPE mechanism because DPE detects the IFA by comparing the *Interest* expired rate with a threshold. Concretely, many legitimate *Interest* may aggregate at a node, expire at the node due to the congestion, and reach the threshold. In that case, because legitimate *Interest* expiration rate reaches the threshold, DPE mechanism regards the legitimate *Interest* as malicious one and eventually lead to the false positive.

In terms of FNR, since malicious *Interest* expiration rate may be lower than the threshold, DPE regards the malicious *Interest* as the legitimate one and has a large FNR (around 30%∼50%). Notably, the FNR of node UK is lower than node ES because node UK is closer to the data producer and receives many malicious *Interest* packets, that is, the malicious *Interest* expiration rate in node UK is more likely to reach the threshold. Compared with DPE , the FNR reduces sharply by using BLAM mechanism. However, BLAM increases slightly comparing with NACK because BLAM can produce false positive when querying the Bloom filter array. It means a malicious *Interest* name is misjudged into the Bloom filter array, which leads to a very small FNR (around 0.4%) in our experiments.

3) Delay: The detecting delay of BLAM, NACK and DPE mechanisms are shown in the Figs 8 and 9. Obviously, BLAM mechanism has the least detecting delay compared with other state-of-the-art mechanisms. Concretely, the detecting delay of DPE mechanism is about 3 seconds. This is because DPE first waits for malicious *Interest* to be expired (2 seconds in the experiments), then takes another 1 second for the *Interest* expired rate to reach the threshold. However, NACK recog-

nizes the attackers using the NACK packets. Such the packets are generated by a data producer to inform downstream nodes that the malicious *Interest* cannot be satisfied. Therefore, the detecting delay of NACK mechanism is low (approximately 1$\sim$3 ms). Notably, from Figs 8 and 9, we can achieve that node UK (about 1.2 ms) has a lower detecting delay compared with node ES (about 3.1 ms) using the NACK mechanism. This is because node UK is closer to the data producer and can receive the NACK packets more quickly than node ES. In addition, because each node has a local Bloom filter array in the BLAM mechanism, the nodes have an ultra-low detecting delay, which is around 0.4 us shown in the Figs. 8 and 9. Based on the array, each node can recognize malicious *Interest* quickly and locally. In other words, they do not need to forward the *Interest* to the data producer to ask whether the *Interest* can be satisfied, while other state-of-the-art mechanisms do. Therefore, we can argue that BLAM sharp reduces memory cost while guaranteeing the accuracy and delay in detecting IFA, *i.e.*, BLAM mechanism is efficient and lightweight for being embedded in the IoT devices.

V. Conclusion

In this paper, we have proposed a lightweight IoT-tailored IFA mitigating mechanism, named BLAM. By introducing a lightweight Bloom filter, the BLAM mechanism is able to reduce the memory cost efficiently while guaranteeing the detecting accuracy and delay. We have analyzed its advantages theoretically, implemented the BLAM solution and evaluated its performance in a realistic experimental platform. In the future work, we will further improve the advanced structure of the Bloom filter to achieve even better performance. Particularly, we aim at adopting more advanced Bloom filter structure such as distance-sensitive Bloom filter to increase the detecting accuracy. We also intent to use hardware acceleration technologies such as GPU to further reduce the detecting delay.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (NSFC) (nos. 61602030, 61702439, and 91638204), the National Key R&D Program (no. 2016YFE0122900), the Fundamental Research Funds for the Central Universities of China (no. 2016RC036), the China Scholarship Council and Natural Sciences and Engineering Research Council (NSERC), Canada.

References

[1] N. Cheng, W. Xu, W. Shi, Y. Zhou, N. Lu, H. Zhou, and X. Shen, "Air-Ground Integrated Mobile Edge Networks: Architecture, Challenges and Opportunities," *IEEE Commun. Mag.*, 2018. [Online]. Available: http://arxiv.org/abs/1804.04763

[2] S. Zhang, W. Quan, J. Li, W. Shi, P. Yang, and X. Shen, "Air-Ground Integrated Vehicular Network Slicing with Content Pushing and Caching," *IEEE JSAC*, 2018. [Online]. Available: http://arxiv.org/abs/1806.03860

[3] Gartner, "Consumer Applications to Represent 63 Percent of Total IoT Applications in 2017," *Tech. Rep.*, 2017. [Online]. Available: https://www.gartner.com/newsroom/id/3598917

[4] J. Ni, K. Zhang, X. Lin, and X. Shen, "Securing Fog Computing for Internet of Things Applications: Challenges and Solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 601–628, Firstquarter 2018.

[5] Y. Zhang, C. Xu, H. Li, K. Yang, J. Zhou, and X. Lin, "HealthDep: An Efficient and Secure Deduplication Scheme for Cloud-Assisted eHealth Systems," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2018.

[6] W. Quan, Y. Liu, H. Zhang, and S. Yu, "Enhancing Crowd Collaborations for Software Defined Vehicular Networks," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 80–86, 2017.

[7] J. Ni, X. Lin, and X. Shen, "Efficient and Secure Service-oriented Authentication Supporting Network Slicing for 5G-enabled IoT," *IEEE JSAC*, pp. 1–1, 2018.

[8] J. Chen, S. Li, H. Yu *et al.*, "Exploiting ICN for Realizing Service-Oriented Communication in IoT," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 24–30, Dec. 2016.

[9] M. Amadeo *et al.*, "Information-centric Networking for the Internet of Things: Challenges and Opportunities," *IEEE Network*, vol. 30, no. 2, pp. 92–100, Mar. 2016.

[10] E. Baccelli, C. Mehlis, O. Hahm, T. C. Schmidt, and M. Wählisch, "Information Centric Networking in the IoT: Experiments with NDN in the Wild," *ACM ICN*, pp. 77–86, 2014.

[11] E. G. AbdAllah, H. S. Hassanein, and M. Zulkernine, "A Survey of Security Attacks in Information-Centric Networking," *IEEE Communications Surveys Tutorials*, vol. 17, no. 3, pp. 1441–1454, Thirdquarter 2015.

[12] Q. Li, X. Zhang, Q. Zheng, R. Sandhu, and X. Fu, "LIVE: Lightweight Integrity Verification and Content Access Control for Named Data Networking," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 2, pp. 308–320, Feb. 2015.

[13] T. Zhi, H. Luo, and Y. Liu, "A Gini Impurity-Based Interest Flooding Attack Defence Mechanism in NDN," *IEEE Communications Letters*, vol. 22, no. 3, pp. 538–541, Mar. 2018.

[14] Y. Xin, Y. Li, W. Wang *et al.*, "A Novel Interest Flooding Attacks Detection and Countermeasure Scheme in NDN," *IEEE Globecom*, pp. 1–7, Dec. 2016.

[15] K. Wang, H. Zhou, Y. Qin, J. Chen, and H. Zhang, "Decoupling malicious Interests from Pending Interest Table to mitigate Interest Flooding Attacks," *2013 IEEE Globecom Workshops*, pp. 963–968, Dec. 2013.

[16] C. Bormann, M. Ersue, and A. Keranen, "Terminology for Constrained-node Networks," *IETF RFC 7228*, 2014.

[17] W. Quan, C. Xu, A. Vasilakos, J. Guan, H. Zhang, and L. Grieco, "TB2F: Tree-bitmap and Bloom-filter for a scalable and efficient name lookup in Content-Centric Networking," *IFIP Networking*, pp. 1–9, June 2014.

[18] W. Quan, C. Xu, J. Guan, H. Zhang, and L. Grieco, "Scalable Name Lookup with Adaptive Prefix Bloom Filter for Named Data Networking," *IEEE Communi. Lett.*, vol. 18, no. 1, pp. 102–105, January 2014.

[19] H. Salah, J. Wulfheide, and T. Strufe, "Coordination Supports Security: A New Defence Mechanism against Interest Flooding in NDN," *IEEE 40th Conference on Local Computer Networks*, pp. 73–81, Oct. 2015.

[20] G. Liu, W. Quan, N. Cheng, K. Wang, and H. Zhang, "Accuracy or Delay? A Game in Detecting Interest Flooding Attacks," *Internet Technology Letters*, vol. 1, no. 2, p. e31, 2018.

[21] Z. Li, K. Liu, Y. Zhao, and Y. Ma, "MaPIT: An Enhanced Pending Interest Table for NDN With Mapping Bloom Filter," *IEEE Communications Letters*, vol. 18, no. 11, pp. 1915–1918, Nov. 2014.

[22] H. Guo, X. Wang, K. Chang, and Y. Tian, "Exploiting Path Diversity for Thwarting Pollution Attacks in Named Data Networking," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 9, pp. 2077–2090, Sept. 2016.

[23] A. Marandi, T. Braun, K. Salamatian, and N. Thomos, "BFR: A Bloom Filter-based Routing Spproach for Information-centric Networks," *2017 IFIP Networking Conference*, pp. 1–9, June 2017.

[24] H. Dai, J. Lu, Y. Wang, and B. Liu, "BFAST: Unified and scalable index for NDN forwarding architecture," *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 2290–2298, Apr. 2015.

[25] H. Zhang, W. Quan, H. Chao, and C. Qiao, "Smart Identifier Network: A Collaborative Architecture for the Future Internet," *IEEE Network*, vol. 30, no. 3, pp. 46–51, 2016.

[26] H. Zhang, P. Dong, W. Quan, and B. Hu, "Promoting Efficient Communications for High-speed Railway using Smart Collaborative Networking," *IEEE Wireless Communications*, vol. 22, no. 6, pp. 92–97, Dec. 2015.