Double-Edge Embedding Based Provenance Recovery for Low-Latency Applications in Wireless Networks

J. Harshan[†], Amogh Vithalkar[†], Naman Jhunjhunwala[†], Manthan Kabra[†], Prafull Manav[†], Yih-Chun Hu[*]

[†]Indian Institute of Technology Delhi, India.
[*]University of Illinois Urbana-Champaign, U.S.A

Abstract—A number of applications in next-generation multi-hop networks, e.g. vehicular networks, impose low-latency requirements on data transmission thereby necessitating the underlying relays to introduce negligible delay when forwarding the packets. While traditional relaying techniques such as amplify-and-forward protocols may help the packets to satisfy latency-constraints, such strategies do not facilitate the destination in learning the path traveled by the packets, which in turn could be used for either learning the topology of the network or detecting security threats on the network. In addition to low-latency constraints, vehicular networks also result in variable network topology owing to the mobility of the nodes, which in turn imposes additional challenges to the destination in learning the path traveled by the packets. Thus, with potential applications to vehicular networks, we address the problem of designing provenance embedding algorithms that reduce the delays on the packets and yet assist the destination in determining the path traveled by the packets with no knowledge of the network topology. We propose a new class of provenance embedding techniques, referred to as double-edge (DE) embedding techniques, wherein a subset of the relay nodes in the path strategically skip the provenance embedding process to reduce the delays on the packets. Using fixed-size bloom filters as tools to implement the double-edge embedding ideas, first, we derive upper bounds on the error-rates of the DE embedding techniques so that the parameters of the bloom filter can be chosen to facilitate provenance recovery within a given quality of service. Subsequently, we present experimental results on a test bed of XBee devices and Raspberry Pis to demonstrate the efficacy of the proposed techniques, and show that the DE embedding techniques offer latency benefits upto 17 $\%$ along with remarkable reduction in error-rates in comparison with the baselines. We also present a security analysis of the proposed provenance embedding methods to asses their vulnerabilities against various attacks including impersonation threats.

Index Terms—Provenance, multi-hop networks, low-latency, bloom filters, security

1 Introduction

Multi-hop networks have been extensively studied in the past as a means of achieving pervasive and ubiquitous communication over a broad class of wireless devices, e.g., in satellite communications and wireless sensor networks. In such networks, a group of wireless devices (henceforth referred to as nodes) are interconnected in such a way that packets from a source node are communicated to the intended destination in a multi-hop manner through several intermediate nodes. While a majority of contributions in this topic have proposed protocols to achieve a reliable end-to-end multi-hop network, a number of contributions have also proposed protocols to facilitate the so-called *provenance* recovery at the destination, wherein provenance refers to the information on the path traced by the packet in the network. One of the objectives of provenance recovery methods is in determining the topology of the network since the intermediate relay nodes cannot directly communicate with the destination. This objective is particularly relevant in sensor network applications wherein the centralized control station is interested in learning the coverage area of sensor deployment [3]- [10]. Other objectives include determining the trustworthiness of the received packets to protect the multi-hop network against various security threats [11]- [21]. For either of these applications, the protocols for provenance recovery take the assistance of the intermediate relay nodes by asking them to embed their signatures on a dedicated portion of the packets. This way, upon receiving the packet, the destination learns the path traced by the packet by verifying the participation of the legitimate nodes using their pre-shared signatures.

As part of the recent developments, next-generation networks have envisioned to accommodate the network requirements of a wide range of wireless devices under the settings of Device-to-Device (D2D), and Vehicle-to-Vehicle/Infrastructure (V2X) communication, wherein the networks must also support low-latency and ultra-reliability features [1] on its packets. In this context, the low-latency feature refers to the requirement that the packets from a source are expected to reach the destination within a given deadline (say in milliseconds). Example applications include a network of autonomous vehicles that intend to communicate their observations to a central infrastructure in a multi-hop fashion, thereby expecting commands from the infrastructure for subsequent actions. Due to the control nature of the underlying messages, the turn-around time from vehicle-to-infrastructure-to-vehicle is desired to

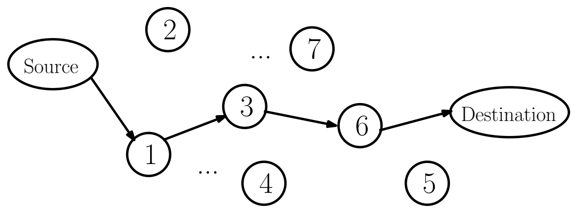

Fig. 1: A multi-hop network model comprising 8 mobile nodes and one fixed destination. With no prior knowledge on the network topology, the destination has to determine the path traced by the packets while satisfying latency-constraints on them.

be bounded within a given time-interval, beyond which the response may be categorized as irrelevant. Furthermore, with applications involving critical infrastructures, these events of deadline-violation in the response may also lead to undesirable outcomes, sometimes even catastrophic.

Although provenance recovery techniques have been studied on a wide range of applications [3]- [21], we highlight that they have not been studied in low-latency applications, e.g., V2X communications. Towards envisaging provenance recovery methods using low-latency packets, we immediately notice that the additional processing-delay incurred because of provenance embedding algorithm at each node is an overhead, and this may not be favorable to packets supporting low-latency applications. Furthermore, we note that the existing provenance embedding methods are not directly applicable in the context of V2X communication owing to the mobility of the underlying relay nodes. Thus, identifying the unique challenges in the mobility of relay nodes in V2X communication along with the conflict between the objective of provenance recovery and the associated delay incurred by embedding the signatures, we revisit the design of provenance recovery mechanisms that support low-latency applications in next-generation wireless networks.

1.1 Problem Statement

With potential applications to V2X and wireless sensor networks, we consider an abstract network model comprising $n-1$ mobile nodes and a fixed destination, as exemplified in Fig. 1. One of the $n-1$ nodes acts as a source to communicate its packets to the destination through a subset of the remaining $n-2$ nodes in a multi-hop fashion. The network is also vulnerable to security threats by an external attacker which is capable of executing a number of adversarial manipulations. Towards facilitating low-latency routing of packets on the above network model, our problem statement is to design new provenance embedding algorithms that (i) reduce the delay introduced on the packets during their journey, (ii) assist the destination in determining the path traced by the received packets with no knowledge on the network topology, and (iii) detect a wide range of security threats posed by an external attacker. Henceforth, throughout the paper, we only consider the delays contributed by the provenance embedding algorithm and the corresponding crypto-primitives. Other network-related delays due to queues, packet re-transmissions, etc. are not considered in order to keep the focus on provenance embedding methods.

1.2 Contributions

Towards handling the premise of no knowledge of network topology at the destination, we first identify that embedding the identity of the relay nodes in the provenance portion of the packet does not convey information on the order of the nodes that forwarded the packet. Therefore, to circumvent this problem, we introduce a new framework of provenance embedding algorithms that assist the destination in recovering the path traced by the packet, and also reduce the delay on the packets during their journey. Specifically:

1) We propose a bloom filter [4], [14] based provenance embedding method, referred to as the edge embedding technique, wherein each relay node, instead of embedding its identity in the bloom filter, embeds the identity of the edge with its preceding node in the bloom filter. As a result, the destination can verify the membership of the participating edges from the bloom filter, and then recover the path traced by the packet using a depth first search algorithm despite not knowing the network topology. In order to identify the right choice of the bloom filter parameters, we propose upper bounds on the error-rates of the edge embedding technique as a function of the number of nodes in the network, number of hops, bloom filter size, and the number of hash functions used by each node. Subsequently, we verify the tightness of the bounds by comparing them with the results from simulations, and show that the number of hash functions which minimizes the upper bound is close to that which minimizes the exact expression (see Section 3).

2) Capitalizing on the idea of edge embedding technique, we propose the Deterministic Double-Edge (DDE) embedding mechanism, wherein a relay node embeds the identity of the pair of edges linked to it: one through which the packet is received and the other through which the packet is sent. With this idea, it is straightforward to observe that a node can cover two edges of the path in one-shot, and as a consequence, the next node in the path need not modify the provenance. Thus, at most half the nodes on the path skip the provenance embedding process thereby resulting in reduced delay on the packet. We show that the hop-counter of the provenance part, which is used by the destination to learn the hop-length, can be used to coordinate the skipping strategy among the nodes. Similar to the edge embedding technique, we use bloom filters to implement the DDE embedding technique, and then propose upper bounds on the error-rates of the DDE embedding technique as a function of the number of nodes in the network, number of hops, bloom filter size, and the number of hash functions used by each node. Subsequently, we verify the tightness of the bounds by comparing them with the results from simulations, and show that the number of hash functions which minimizes the upper bound is close to that which minimizes the exact expression (see Section 4).

3) We demonstrate the benefits of both edge and double-edge embedding techniques on a test bed of six nodes, each comprising a Raspberry Pi 3+ and a Digi XBee S2C, for computation and communication purposes, respectively. For a given bloom filter size, we compute the number of hash

TABLE 1: Summary of Contributions and Impact of the Proposed Methods. Notations are as defined as in Table 3.

Provenance Embedding Method	**Latency**	**Coordination overhead**	**Complexity at the destination to verify the bloom filter**	**Provenance size to achieve a given error-rate**
Edge embedding	$h(T_E + T_P)$	Not needed	$O(n^2)$	
DDE embedding with $N = 1$	$hT_E + \lfloor \frac{h}{2} \rfloor T_P$	Uses hop-counter	$O(n^3)$	Less than edge embedding
DDE embedding with $N = \lfloor \frac{h}{2} \rfloor$	$hT_E + T_P$	Uses hop-counter	$N \times O(n^3)$	

functions that minimizes the error-rates of the edge and the double-edge embedding techniques, and subsequently implement them on the test bed to measure the latency and the error-rates. With hop-lengths of $3, 4$ and 5, we show that the double-edge embedding method outperforms the edge embedding method both in terms of delay (reduction of at most 15-17%) as well as error-rates (see Section 5).
4) We present a security analysis of the proposed techniques against various threats posed by an external attacker. We specifically focus on impersonation threats [22], wherein an external attacker, after compromising one of the nodes in the path, modifies the bloom filter contents to force the destination in recovering an incorrect path that differs from the actual path at the position of the compromised node. Although the identities of the edges and the double-edges of a node can be held private to mitigate such attacks, we show that owing to false-positive characteristics of the bloom filter, impersonation attacks can be executed with non-zero success-rate. However, through experiments, we show that impersonation attacks can be detected with high accuracy at the destination provided the bloom filter size is sufficiently large (see Section 6).

A summary of our contributions along with a comparison between the proposed techniques are listed in Table 1. Since the identity of the edges (or double-edges) are embedded into the provenance at a single layer, our framework falls under the well known class of linear provenance as against aggregated provenance [24], wherein information from different layers are embedded in the provenance.

1.3 Related Work

Typical use-cases of network provenance include discerning the origination of a message, the path traveled by it, and the details on how the messages were derived and which parties were involved in its derivation. The related work on network provenance can be broadly classified into two groups: (i) contributions in non-adversarial environments wherein the focus is on designing provenance schemes to support diagnostics by minimizing the transmission overhead due to provenance [3]- [10], and (ii) contributions in adversarial environments wherein the provenance schemes are proposed to detect and mitigate specific form of threats on the network [11]- [21].

In the latter class of contributions, which is the subject matter of this work, [11] studied false data injection attacks in wireless sensor networks and proposed a combined packet marking- and logging-scheme for traceback to reconstruct the entire path. A provenance based mechanism is also proposed in [12] to handle packet dropping attacks. This study utilized the inter-packet delay based provenance transmission technique and devised a detection mechanism based on the distribution of these delays. [13] used in-packet bloom filter to encode the IDs of the nodes that are on the path of the packets. In [14], the authors proposed a light-weight provenance encoding and decoding scheme based on bloom filters to securely transmit provenance in sensor networks. They also extended the scheme to incorporate data provenance binding, and to include packet-sequence information that supports detection of packet-loss attacks. In [15], the authors design energy-efficient provenance encoding schemes and demonstrate their feasibility in IP traceback to wireless sensor networks. Recently, the authors in [16] proposed a light-weight data protocol based on bloom filters and analyzed the leakage of information to an adversary having partial knowledge of the bloom filter. The authors in [17] proposed a path tracing approach for routing protocol for low-power and lossy-networks, wherein the proposed method identifies packet drops and misbehaving nodes in the network. While contributions in [11]- [17] proposed embedding techniques to facilitate the destination to detect various attacks, another class of provenance methods [19], [20], [21] focused on designing vigilance policies at the intermediate nodes against several threat models before forwarding the packets. We highlight that our work falls in the category of [11]- [17], but not [19], [20], [21]. For a detailed survey on the state-of-the-art developments in secure network provenance, we refer the readers to [18]. To highlight the novelty of our work, we have listed the main differences between our work over the existing contributions in Table 2.

2 Network Model

We consider a wireless network, as exemplified in Fig. 1, comprising a set of n nodes, denoted by $\mathcal{N} = \{1, 2, \ldots, n\}$. One of the nodes in $\mathcal{N}$ is the destination, one of them acts as the source, whereas a subset of the remaining $n-2$ nodes assist the source in relaying the packets to the destination. The destination is also connected to a network of gate-way nodes via secure back-haul links, in order to form the core network.[1] We make the following assumptions on our network model: The destination is geographically fixed, whereas the other $n-1$ nodes are mobile, and as a result, the network topology is unknown to the destination. The destination has more computational power than the other nodes. Each node has a unique secret-key pre-shared with the core network, using which it can authenticate itself through one of the gate-way nodes. Each node in the network authenticates its identity with its neighboring nodes using a public-key cryptosystem based authentication protocol [26]. As an example,

1. In the context of vehicular networks, we can envisage gate-way nodes as access points deployed at the entry points of the geographical area over which V2X communication is enabled.

TABLE 2: Novelty of our approach with respect to existing contributions

Reference	Existing contributions	Limitation with respect to our work
[4]	Use of multi-dimensional bloom filter for provenance	The destination has to wait for multiple packets to determine the provenance
[5]	Use of arithmetic coding to learn the topology and to reduce the provenance size	This method works in a static topology but fails in a changing topology Also, needs prior probability to execute arithmetic coding
[6]	Use of dictionary-based leaning to determine the topology and reduce the provenance size	This method works in a static topology. The efficiency of this method reduces in changing topology
[9]	Use of compressed sensing methods to reduce provenance size	Provenance can be tampered with a low chance of detection, by an impersonator in an adversarial environment
[12]	Bloom filter based provenance is used in a known topology	The embedding method used in this work fails to detect the provenance in an unknown topology
[13]	This work uses a light-weight in-packet bloom filter type provenance to securely transmit the packet in the network	The method used in this work fails to evaluate the provenance and find a path in a changing topology
[14]	Bloom filter based provenance is used with a node embedding technique in a known topology to identify packet dropping attacks	This method fails in an unknown topology

the nodes may use a traditional pseudonym-based signatures [25], wherein either a gate-way node or the destination distributes a list of pseudonyms along with the public-key certificates and private-key signatures of each node. This distribution can be accomplished through a secure channel using the unique secret-key of the node pre-shared with the core network. During the neighbor discovery phase, node b, for $b \in \mathcal{N}$, signs a message based on its pseudonym and the private-key signature to its neighbor, say node a, and subsequently gets verified using the corresponding public-key certificates.[2] Among the nodes in $\mathcal{N}$, there exists a directed edge from node a to node b, for $a, b \in \mathcal{N}$, if (i) the latter node is within the coverage range of the former, and (ii) the two nodes successfully authenticate each other.

Typical applications that support the above assumptions include V2X communication [2] and wireless sensor networks [14], wherein the role of the destination is played by the road-side unit and the central control station, respectively, and the roles of the remaining $n-1$ nodes are played by the mobile vehicles and mobile sensors, respectively. Since the nodes are mobile and their coverage areas may be limited, the set of edges that exists among the nodes is a subset of $\mathcal{E}$, where $\mathcal{E} = \{e_{a,b} \mid a, b \in \mathcal{N} \text{ s.t. } a \neq b\}$, where $e_{a,b}$ denotes the direct edge from node a to node b.

The source node, say node i_1, for $i_1 \in \mathcal{N}$, has N packets, denoted by their identities $\{p_l, 1 \leq l \leq N\}$, to communicate to the destination within a given deadline. These N packets are routed through h hops, for $1 \leq h \leq n-1$, with the help of $h-1$ relay nodes, denoted by $i_2, i_3, \ldots, i_h \in \mathcal{N}$. Suppose that the l-th packet traverses the following nodes in the order $i_1 \rightarrow i_2 \rightarrow \ldots \rightarrow i_h$ before reaching the destination. Upon receiving the l-th packet, the destination intends to determine the path traveled by the packet, henceforth referred to as provenance of the l-th packet. To assist provenance determination, the source node uses a packet structure, which includes dedicated bits to provide the provenance information. We apply fixed-size bloom filters [4] of size m bits to convey the provenance information on the packet. In particular, each relay node uses its unique-key to embed the identity of its attribute in the bloom filter by setting k random positions, for $1 \leq k \leq m$, to one. We also include a hop-counter in the provenance portion to provide information on the number of hops to the destination. Upon receiving the bloom filter contents and the hop-counter, the destination verifies the nodes' attributes (using their pre-shared keys) to determine the path traveled by the packet.

When facilitating low-latency communication of packets over a multi-hop network, we address the following two types of provenance recovery constraints at the destination: (i) Strict recovery constraint, wherein the destination must recover the path traced by every packet, and (ii) Relaxed recovery constraint, wherein the destination must recover the path traced by a set of N consecutive packets (assuming all the N packets travel the same path).

2.1 Threat Model

We consider both passive and active threats on our network model. Under the class of passive threats, we assume the presence of an external attacker that is keen on eavesdropping the provenance information. In order to address this passive threat, we propose to keep the provenance part of the packet confidential by either using pair-wise keys between adjacent nodes, or using a group-key, which is shared among the n nodes in the network. When bloom filters are used to convey provenance information in the packet, confidentiality of the identity of the node's attributes is implicitly preserved due to the use of collision resistant one-way hash functions. However, the hop-counter value which is also a part of provenance must be kept confidential since hop-counter in plain-text could potentially reveal the origin of the packet to an external eavesdropper. Therefore, we propose to encrypt the provenance portion in our network model. As a result, when forwarding the packet, a relay node decrypts the provenance portion from the packet, modifies it, and then encrypts it before forwarding the packet to the next node.

2. It is well known that frequent distribution of public-key certificates and private-key signatures is a drawback of this authentication mechanism. However, we remark that optimization of the communication-overheads of the authentication protocol is out of scope of this work, and we have used this technique only to emphasize the use of a strong authentication algorithm in our network model.

Under the class of active threats, we are interested in (i) *External attacks*, wherein an attacker can execute a man-in-the-middle attack (MitM) by manipulating the contents of the packet, as well as (ii) *Insider attacks*, wherein an external attacker compromises one of the nodes in the network [22], [23], and then executes a number of adversarial manipulations on the packet. With respect to the MitM attack, we point out that link-level strategies such as message-authentication-codes (MAC) can be employed for attack detection. However, with respect to insider attacks, it is well known that designing mitigation techniques are challenging since the scope of adversarial manipulations by an insider is unbounded owing to complete access to a legitimate node. Inline with the objective of determining the provenance, which forms the heart of this work, we consider the class of impersonation attacks, wherein one of the nodes on the path manipulates the bloom filter so as to misguide the destination in learning a path that differs from the actual path at the position of the compromised node. In order to detect the impersonation attack, we propose to keep the identities of the edges (or the double-edges) private from the rest of the nodes, and only share it with the destination. A detailed analysis on how the idea of obfuscating the identities of the edges (or the double-edges) helps detecting the impersonation attack is presented in Section 6.

Henceforth, throughput the paper, we use the notations listed in Table 3. Furthermore, in a network of n nodes, the path traveled by the packet from the source to the destination is referred to as the main path. The notation $c!$ is used to represent factorial of a positive integer c. Given two numbers a and b, such that $b \leq a$, we use $\binom{a}{b}$ to denote the number of ways of choosing b objects out of a. We use $\Pr(\cdot)$ to denote the usual probability operator. We use $\{0,1\}^m$ to represent the set of all m-length sequences over the alphabet $\{0,1\}$. If $\mathcal{E}$ is a subset of $\mathcal{U}$, then $\overline{\mathcal{E}}$ represents the complement of $\mathcal{E}$ in $\mathcal{U}$.

TABLE 3: Notations used in this paper

Notation	Meaning of the symbol
n	Number of nodes in the network
N	Number of packets
p_l	Identity of l-th packet
h	Number of hops in the path
BF_l	bloom filter of l-th packet
m	Size of the bloom filter
k	No. of hash functions used in bloom filter
T_E	Delay introduced by crypto-primitive operations
T_P	Delay introduced by embedding the provenance

3 Edge Embedding Technique

With no knowledge of the network topology, we point out that the idea of embedding the identity of the participating nodes into the bloom filter [14] does not help the destination to recover the information on the order of the participating nodes. Therefore, to circumvent this problem, we propose to embed the identity of the participating edges into the bloom filter since each relay node has the knowledge of the preceding node in the path. Furthermore, we intend to keep the identity of the edges of a node private from the other nodes, and only share it with the destination so that it can verify the memberships of the edges using the bloom filter contents. In the following section, we explain the key derivation protocol which is used by node a, for $a \in \mathcal{N}$, to generate a secret-key corresponding to its edges, denoted by $\mathcal{E}_a \triangleq \{e_{b,a} \mid b \in \mathcal{N} \text{ s.t. } a \neq b\}$.

3.1 Key Derivation Algorithm for Edges

Based on the authentication mechanism discussed in Section 2, we assume that all the nodes (including the destination) have the list of pseudonyms of all the authenticated nodes, denoted by $\mathcal{P} = \{\pi_r \mid \forall r \in \mathcal{N}\}$. Also, let k_a denote the unique pre-shared key of node a, for $a \in \mathcal{N}$, using which it authenticates itself to a gate-way node. If node b, for $b \neq a$, successfully authenticates with node a, to forward a packet, then node a derives the secret-key representing the edge from node b to node a as $k_{b,a} = f_E(k_a, \pi_b, \pi_a)$, where $f_E(\cdot)$ is an appropriate pseudorandom function (PRF) to derive a unique key based on the pseudonyms π_a and π_b along with the private key k_a. Since the destination has the list of pseudonyms, and also the list of unique pre-shared keys of all the nodes in the network, it can locally derive the same set of shared secret-keys associated to the edges of the network. Furthermore, since k_a is private to node a, the secret-keys associated with its edges cannot be generated by the other nodes in the network. In the above key derivation algorithm, the secret-key k_a can also be replaced by a private pseudonym to provide privacy to the identity of the nodes.

3.2 Edge Embedding Algorithm

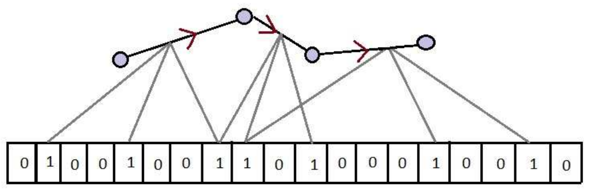

Fig. 2: An example for embedding edges in the bloom filter. In this example, each node in the path embeds the information of the preceding edge by using $k = 3$ hash functions into the bloom filter of size 19 bits.

Embedding Process: On the l-th packet, the source node initializes the bloom filter values to zero, i.e., $BF_l = [0,\ 0, \ldots, 0] \in \{0,1\}^m$, sets the hop-counter to one, and then encrypts the provenance before sending the packet to node i_2. The node i_j, for $1 < j \leq h$, receives the l-th packet from i_{j-1}, decrypts the provenance, and then extracts the bloom filter portion, given by $BF_l = [BF_l(1), BF_l(2), \ldots, BF_l(m)] \in \{0,1\}^m$, and the hop-counter value. Using k_{i_{j-1},i_j}, as derived in Section 3.1, the bloom filter components are modified by node i_j on k positions as

$$BF_l\left(v^{(r)}_{i_{j-1},i_j}\right) = 1, \qquad (1)$$

where

$$v^{(r)}_{i_{j-1},i_j} = \mathrm{H}\left(k_{i_{j-1},i_j}, p_l, n^{(r)}_{i_{j-1},i_j}\right) \in [m], \qquad (2)$$

such that $1 \leq r \leq k$, and $\mathrm{H}(\cdot,\cdot,\cdot)$ is a hash function which generates a random number in $[m] = \{1, 2, \ldots, m\}$ using the inputs (i) p_l, which is the identity of the l-th packet, (ii) $n^{(r)}_{i_{j-1},i_j}$ is the nonce used to generate the r-th index, and (iii) k_{i_{j-1},i_j} is the pre-shared key with the destination. In the embedding process, nonce values are used to generate k independent bit-locations on the bloom filter. After updating the bloom filter, node i_j updates the hop-counter, and then encrypts the provenance before forwarding the packet to the next node, denoted by node i_{j+1}. The above procedure is followed by each relay node in the path $i_2, \ldots, i_h$. An example for the edge embedding technique using bloom filter is shown in Fig. 2. Due to distinct nonce values and distinct identities given to the edges, we note that the index values chosen at the output of the hash functions are statistically independent. In practice, the value $1 \leq r \leq k$ could be used as the nonce value to generate the r-th index.

Verification and Path Retracing: The total number of directed edges available in the network is $2\binom{n}{2}$. Furthermore, since the destination is the sink node, and has the knowledge of its preceding node, the total number of valid edges for verification is $2\binom{n}{2} - 2(n-1)$. Thus, the total set of secret-keys at the destination is $\mathcal{K}_{SE} \subset \{k_{b,a} = f_E(k_a, \pi_b, \pi_a) \mid a, b \in \mathcal{N} \text{ s.t. } a \neq b\}$. Using $\mathcal{K}_{SE}$ and the pre-shared nonce values, the destination verifies the identity of edges with the received bloom filter, and a list of candidate edges, denoted by $\mathcal{E}^{(l)}_{BF}$, is obtained as

$$\mathcal{E}^{(l)}_{BF} = \{e_{b,a} \in \mathcal{E} \mid BF(v^{(r)}_{b,a}) = 1, 1 \leq r \leq k\},$$

where

$$v^{(r)}_{b,a} = \mathrm{H}\left(k_{b,a}, p_l, n^{(r)}_{b,a}\right). \tag{3}$$

Using $\mathcal{E}^{(l)}_{BF}$ and the hop-counter, the destination recovers the path traced by the l-th packet from the subgraph formed by $\mathcal{E}^{(l)}_{BF}$. This task is accomplished using a path retracing algorithm that uses a depth-first-search (DFS) on the graph formed by $\mathcal{E}^{(l)}_{BF}$. When m is sufficiently large with respect to k and $\binom{n}{2}$, then the graph $\mathcal{E}^{(l)}_{BF}$ is likely to have only the set of edges traced by the packet. However, when m is comparable with respect to k and $\binom{n}{2}$, then $\mathcal{E}^{(l)}_{BF}$ may have edges other than those participated in forwarding the packet owing to probabilistic nature of bloom filter. We define a false-positive (also referred to as error-rate) event in provenance recovery, denoted by E_{fp}, when more than one path of hop-length h is recovered at the destination using $\mathcal{E}^{(l)}_{BF}$. Formally, error-rate of the edge embedding technique, given by

$$p_{error} = \Pr(E_{fp}), \tag{4}$$

which is the fraction of packets for which the destination recovers more than one path of hop-length h from $\mathcal{E}^{(l)}_{BF}$.

3.3 False-Positive Analysis of Edge Embedding Technique

Error-rate, as defined in (4), is a measure of the efficacy of the bloom filter to help the destination in recovering the provenance. In particular, for a given n, m, and h, we must choose $k \in [m]$ such that the error-rate is minimized. In the rest of this section, we propose an upper bound on the error-rate of edge embedding technique so that it can be used to arrive at the best value of k as a function of m, n and h. From first principles, error-rate is given by

$$\Pr(E_{fp}) = \sum_{i=1}^{min(m,kh)} \Pr(E_{fp}|C_i)\Pr(C_i), \tag{5}$$

where $\Pr(C_i)$ is the probability that i positions of the bloom filter have been chosen by the edges on the path traversed by the packet, and $\Pr(E_{fp}|C_i)$ is the probability that more than one path of hop-length h is recovered at the destination using $\mathcal{E}^{(l)}_{BF}$ conditioned that i positions of the bloom filter are set. In other words, $\Pr(C_i)$ can be written as the fraction of the number of ways of distributing kh objects in i distinct boxes so that no box remains empty and the number of ways of distributing kh objects in m boxes. Considering two sets $\mathcal{Y}$ and $\mathcal{Z}$ with kh and i elements, respectively, the numerator of the fraction represents the number of onto functions from $\mathcal{Y} \to \mathcal{Z}$, and this can be calculated using the inclusion-exclusion principle. Similarly, considering two sets $\mathcal{Y}$ and $\mathcal{Z}'$ with kh and m elements, respectively, the denominator of the fraction represents the number of functions from $\mathcal{Y} \to \mathcal{Z}'$. Thus, we can write $\Pr(C_i)$ as

$$\Pr(C_i) = \frac{\binom{m}{i}\sum_{\gamma=0}^{i}((-1)^{\gamma})\binom{i}{\gamma}(i-\gamma)^i}{m^{kh}}. \tag{6}$$

Furthermore, $\Pr(E_{fp}|C_i)$ can be written as

$$\Pr(E_{fp}|C_i) = 1 - \Pr(\overline{E_{fp}}|C_i), \tag{7}$$

where $\Pr(\overline{E_{fp}}|C_i)$ can be calculated by exhaustively counting all the cases that do not result in false positives. Since handling all possible cases that do not result in false positives is intractable, the following theorem provides a lower bound on $\Pr(\overline{E_{fp}}|C_i)$ by considering a subset of cases that are guaranteed not to result in false positives.

Theorem 1. For a given value of n, m, h, and k, a lower bound on $\Pr(\overline{E_{fp}}|C_i)$ can be given as

$$\Pr(\overline{E_{fp}}|C_i) > g_E(n, m, h, k, i), \tag{8}$$

where $g_E(n, m, h, k, i)$ is on the Right Hand Side of (18).

Proof: With no knowledge on the network topology, the number of edges we need to consider for calculating false positives is $E = 2\binom{n}{2} - h - 2(n-1) + 1$. The set of such edges, denoted by $\bar{\mathcal{E}}$, is obtained by discounting the edges on the path traversed by the packet, the edges that terminate at the destination, and the edges that originate from the destination. Note that edges that terminate at the destination are discounted since the destination has the knowledge of the penultimate node due to the neighbor discovery process.

The bloom filter values received at the destination will have i positions with ones and $m - i$ positions with zeros. Given that the edges on the path traversed by the packet have chosen i positions in the bloom filter, an edge in $\bar{\mathcal{E}}$ is said to be lit if the k positions chosen by that edge is a subset of the i positions chosen by the main path. Therefore, the probability that an edge in $\bar{\mathcal{E}}$ is not lit in the bloom filter is given by $q_i = 1 - \left(\frac{i}{m}\right)^k$. To compute the above expression, we assume that the output of the hash functions across the nodes are statistically independent. Note that since i can vary from 1 to $\min(kh, m)$, we take into consideration all possible values of i in this proof. To calculate a lower bound on $\Pr(\overline{E_{fp}}|C_i)$, we count those events wherein although a subset of $\bar{\mathcal{E}}$ is lit in the bloom filter, they do not contribute to false positives. Henceforth, throughput the proof, we refer to the path traced by the packet as the *main path*.

Case 1: We consider a scenario wherein no edge in $\bar{\mathcal{E}}$ is lit in the bloom filter. Since the index values chosen by the hash functions are statistically independent across the edges, the probability that no edge in $\bar{\mathcal{E}}$ is lit in the bloom filter is

$$P_1^{\bar{\mathcal{E}}} = q_i^E. \tag{9}$$

Case 2: We consider a scenario wherein edges that are isolated from the main path are lit in the bloom filter. By using $k' \geq 1$ to denote the number of such edges, the probability of such a scenario is given by

$$P_2^{\bar{\mathcal{E}}} = \sum_{k'=1}^{2\binom{n-h-1}{2}} \binom{2\binom{n-h-1}{2}}{k'} q_i^{E-k'}(1-q_i)^{k'}, \tag{10}$$

where $2\binom{n-h-1}{2}$ denotes the total number of isolated edges from the main path. This case is as illustrated in Fig. 3. In general, the number of ways to choose $k'-\kappa$ isolated edges from $n-h-1-\gamma$ nodes is defined as

$$T(\gamma,\kappa) \triangleq \binom{2\binom{n-h-1-\gamma}{2}}{k'-\kappa}. \tag{11}$$

Henceforth, throughout the proof, we denote the term $q_i^{E-k'}(1-q_i)^{k'}$ as G.

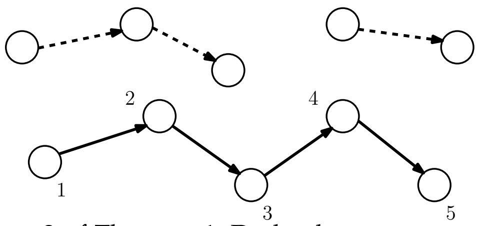

Fig. 3: Case 2 of Theorem 1: Dark edges represent the path traversed by the packet, whereas dashed edges represent isolated edges.

Case 3: We consider three types of edges in $\bar{\mathcal{E}}$: (i) inward directed edges which merge on one of the nodes (except the second node) on the main path, (ii) outward directed edges which originate from a node on the main path and may join another path which does not merge on one of the nodes on the main path, and (iii) edges that are isolated from the main path and not terminating at edges of type-(i). An example for this case is captured in Fig. 4.

We consider $k' \geq 1$ edges of $\bar{\mathcal{E}}$ lit in the bloom filter, out of which y edges are connected to the main path. Out of those y edges, let z be the number of outward direct edges from the main path, and $y-z$ be the number of inward directed edges which terminate on the main path. The number of ways in which we can select y nodes (outside the main path) that connect to the main path is $\binom{n-h-1}{y}$. The number of ways in which we can choose z nodes (corresponding to outward edges) out of these is $\binom{y}{z}$. The total number of nodes on which $y-z$ edges can merge on the main path is $h-1$, and the total number of nodes from which z edges can diverge from the main path is h. With that, y edges of type-(i) and type-(ii) can be chosen in $\binom{n-h-1}{y}\binom{y}{z}h^z(h-1)^{y-z}$ ways. Furthermore, we can select the remaining $k'-y$ isolated edges in

$$R = \binom{2\binom{n-h-1-y+z}{2} + (y-z)(n-h-1-y+z)}{k'-y} \tag{12}$$

ways, where $2\binom{n-h-1-y+z}{2}$ represents the number of edges after removing $y-z$ nodes, and $(y-z)(n-h-1-y+z)$ represents the number of outward directed edges from the nodes where inward directed edges merge on the main path. Overall, the probability of events in this case is given in (13).

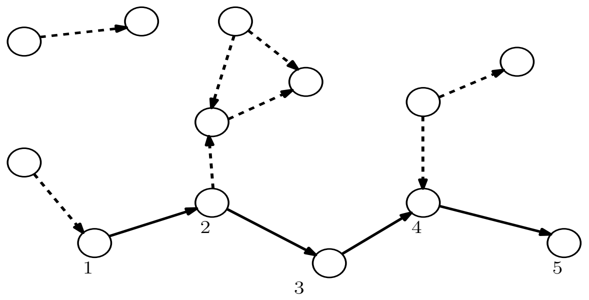

Fig. 4: Case 3 of Theorem 1: Outward directed edges that originate from nodes on the main path (dark edges), inward directed edges that terminate at nodes on the main path, and isolated edges.

Case 4: We consider two types of edges in $\bar{\mathcal{E}}$: (i) backward loops, which start from a node on the main path and merge at one of the preceding nodes on the main path (in the anti-clockwise direction), and (ii) isolated edges which do not have any connection with the edges on the main path and also with the loops considered in (i). An example for this case is presented by Fig. 5.

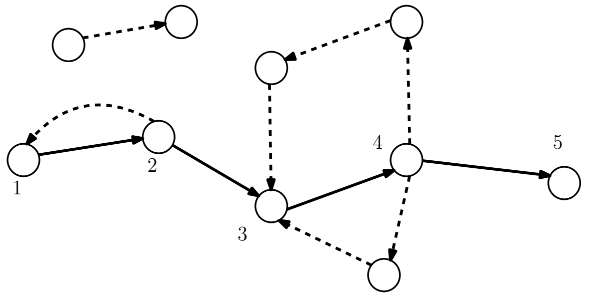

Fig. 5: Case 4 of Theorem 1: Backward loops that originate from a node on the main path (dark edges) and terminate at one of the preceding nodes on the main path, and isolated edges. As shown, the 3-edged backward loop from node '4' to node '3' results in false positives, and therefore we have omitted such cases.

First, we count single-edged backward loops under type-(i). Based on Fig. 5, if we select node '1' as the terminating node of the loop, then we can choose its originating node in $h-1$ ways (by excluding the destination and the node itself). With node '2' as the terminating node, we can choose the originating node in $h-2$ ways, and similarly, with node '$h-1$', we can choose the terminating node in only one way. Thus, the total number of single-edged backward loops is

$$P_3^{\bar{\mathcal{E}}} = \sum_{k'=1}^{E}\sum_{y=1}^{k'}\sum_{z=0}^{y}\binom{n-h-1}{y}\binom{y}{z}h^z(h-1)^{y-z}RG. \tag{13}$$

$L_0 = h-1+\sum_{\beta=1}^{h-2}\beta$. Out of these L_0 choices, we can select t loops which correspond to t edges in $\binom{L_0}{t}$ ways. With t edges as backward loops as per type-(i), the rest of the $k'-t$ edges can be selected as isolated edges in $T(0,t)$ ways under type-(ii), where $T(\cdot,\cdot)$ is as defined in (11).

To generalize, we consider backward loops formed with $\alpha+2$ nodes, out of which the originating and the terminating nodes are on the main path, whereas the α intermediate nodes are not on the main path. Similar to the case when $\alpha=0$, the total number of $(\alpha+1)$-edged backward loops is given by $L_\alpha = h-1+\sum_{\beta=\alpha}^{h-3}(h-2-\beta)$. We can select t loops out of these L_α choices in $\binom{L_\alpha}{t}$ ways. These t loops will correspond to $(\alpha+1)t$ edges. For these loops, we can select αt nodes which are not on the main path in $\binom{n-h-1}{\alpha t}(\alpha t)!$ ways. Subsequently, we can select the remaining $k'-(\alpha+1)t$ edges as isolated edges in $T(\alpha t,(\alpha+1)t)$ ways. Thus, the probability of the events in this case is given in (14).

Case 5: We consider two types of edges in $\bar{\mathcal{E}}$: (i) forward loops, which originate from a node on the main path and terminate at one of the nodes in the downstream of the main path, and (ii) isolated edges that have no connection to the main path and also to the forward loops mentioned in (i). An example for this case is shown in Fig. 6

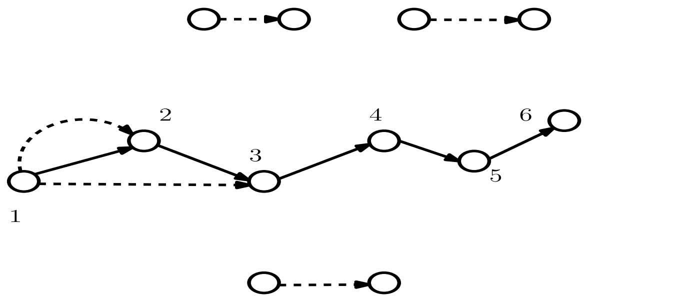

Fig. 6: Case 5 of Theorem 1: Forward loops that originate from a node on the main path (dark edges) and terminate at one of the nodes in the downstream on the main path, and isolated edges.

First, we consider loops that are formed by nodes on the main path. As illustrated in Fig.6, if we select node '1' as the originating node, then we can choose the terminating node in $h-2$ ways (excluding the destination, node '2' and the node itself). Similarly, for node '2', we can select the terminating node in $h-3$ ways, and finally for node '$h-1$', there are no available choices. Thus, the number of forward loops of type-(i) are $L'_0 = \sum_{\beta=1}^{h-2}\beta$. Out of these L'_0 choices, we can select t loops in $\binom{L'_0}{t}$ ways. These t loops will result in t edges under type-(i). With that the remaining $k'-t$ edges of type-(ii) can be selected in $T(0,t)$ ways. Generalizing the above result to forward loops with $\alpha+2$ nodes such that α of them are outside the main path, the probability of this case is given in (15), where $L'_\alpha = \sum_{\beta=0}^{h-2-\alpha}\beta$.

Case 6: We consider (i) inward directed paths of length at least two which terminate at one of the nodes on the main path, (ii) outward directed edges (which do not form a loop) that originate from a node on the main path, and (iii) isolated edges with no connection to the edges on the main path and those considered in (i). In particular, at any node on the main path, there can be only one inward directed path, however, there can be more than one outward directed path. This case is illustrated using Fig. 7.

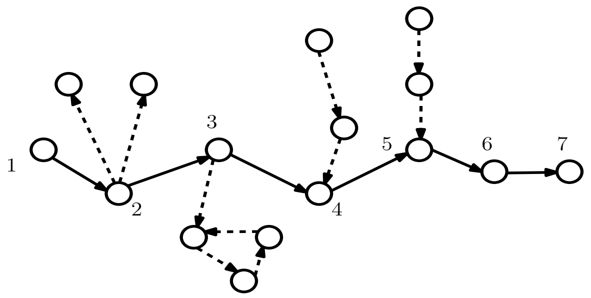

Fig. 7: Case 6 of Theorem 1: Inward directed paths of length at least two that terminate at one of the nodes on the main path (dark edges), outward directed edges that originate from a node on the main path, and isolated edges.

Let the length of an inward directed path be $\beta \geq 2$. Inward directed paths can terminate at any of the $h-\beta$ nodes on the main path. Let us choose l of these paths. These correspond to $l\beta$ edges. The total number of ways in which inward directed paths of length β can be formed at l nodes is $A=\binom{h-\beta}{l}\binom{n-h-1}{l\beta}(l\beta)!$. We then choose $k'-l\beta$ more edges, out of which x edges are of type-(ii) and $k'-l\beta-x$ edges are of type-(iii). Since these x outward edges can originate from any of the h nodes on the main path, we can select them in $B=\binom{n-h-1-l\beta}{x}h^x$, ways. The rest of the $k'-l\beta-x$ edges can be chosen in $T(l\beta,l\beta+x)$ ways. Overall, the probability of this case is given in (16).

$$P_6^{\bar{\mathcal{E}}} = \sum_{k'=1}^{E}\sum_{\beta=2}^{h-1}\sum_{l=1}^{h-\beta}A\sum_{x=0}^{k'-l\beta}BT(l\beta,l\beta+x)G. \tag{16}$$

Case 7: As a generalization of Case 6, we consider a scenario wherein there exists an outward directed edge at that node which has an inward directed path. This case is illustrated in Fig. 8 (see paths connected to nodes '4' and '5'). Using counting arguments similar to that of Case 6, the probability of this case is given by

$$P_7^{\bar{\mathcal{E}}} = \sum_{k'=1}^{E}\sum_{\beta=2}^{h-1}\sum_{l=1}^{h-\beta}A\sum_{x=0}^{k'-l\beta-l}BT(l\beta,l\beta+l+x))G, \tag{17}$$

where $A=\binom{h-\beta-1}{l}\binom{n-h}{l\beta+l}(l\beta+l)!$ and $B=\binom{n-h-1-l\beta-l}{x}h^x$.

From Case 1 to Case 7, we have considered a set of events which are guaranteed not to generate false positives given that i positions of the bloom filter are lit by the edges on the main path. Therefore, summing the probabilities of these

$$P_4^{\bar{\mathcal{E}}} = \sum_{k'=1}^{E} \sum_{\alpha=0}^{h-3} \sum_{t=0}^{L_\alpha} \binom{L_\alpha}{t} \binom{n-h-1}{\alpha t} ((t\alpha)!)(T(\alpha t, (\alpha+1)t))G \tag{14}$$

$$P_5^{\bar{\mathcal{E}}} = \sum_{k'=1}^{E} \sum_{\alpha=0}^{h-2} \sum_{t=0}^{L'_\alpha} \binom{L'_\alpha}{t} \binom{n-h-1}{\alpha t} ((t\alpha)!)(T(\alpha t, (\alpha+1)t))G \tag{15}$$

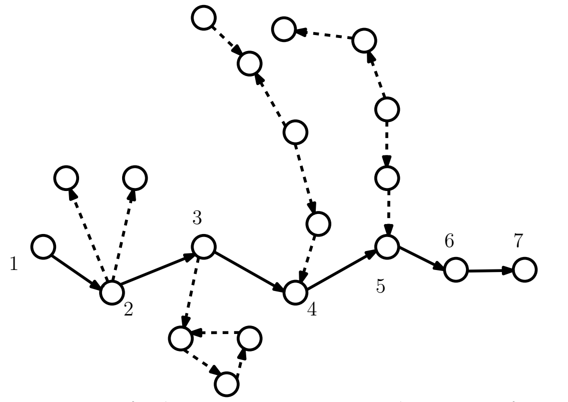

Fig. 8: Case 7 of Theorem 1: A generalization of Case 6, wherein there is an outward directed edge at a node which also has an inward directed path of length at least two terminating on it.

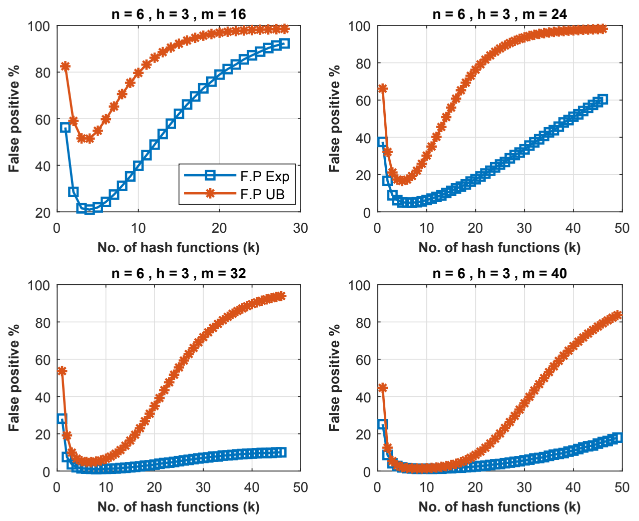

Fig. 9: Comparison between the error-rate of edge embedding scheme and the upper bound for $[n, h] = [6, 3]$.

discussed events gives us a lower bound on the probability of not witnessing false positives. Thus, we get

$$\Pr(\overline{E_{fp}}|C_i) > \sum_{j=1}^{7} P_j^{\bar{\mathcal{E}}}, \tag{18}$$

where $P_1^{\bar{\mathcal{E}}}, P_2^{\bar{\mathcal{E}}}, \ldots, P_7^{\bar{\mathcal{E}}}$ are respectively given in (9), (10), (13), (14), (15), (16), and (17). □

Note that $g_E(n, m, h, k, i)$ can be numerically computed given $n, m, h,$ and k. By substituting the lower bound given by Theorem 2 in (7), we obtain an upper bound on $\Pr(E_{fp}|C_i)$. Subsequently, substituting this upper bound on $\Pr(E_{fp}|C_i)$ and the exact expression of $\Pr(C_i)$ in (5), we obtain an upper bound on the average probability of false positives of the edge embedding technique.

To verify the tightness of the proposed upper bound, we compare it with the error-rates obtained through simulations for several values of n and h. In particular, for a given value of n and h, we vary the bloom filter size (m), and accordingly compute the error-rates as a function of the number of hash functions. The plots, which are presented in Fig. 9 and Fig. 10, highlight that for a given n and h, (i) the upper bound gets tighter, especially around the value of k that minimizes the error-rate, as the bloom filter size increases, and (ii) the value of k that minimizes the upper bound is close to that which minimizes the exact value. These results imply that when the bloom filter size is large, our proposed upper bound can be used to arrive at an appropriate value of k for a given m, h and n.

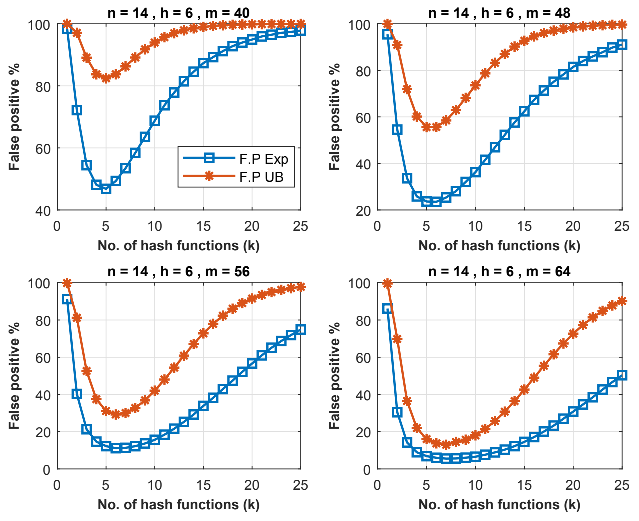

Fig. 10: Comparison between the error-rate of edge embedding scheme and the upper bound for $[n, h] = [14, 6]$.

4 Double-Edge Embedding Technique

In the edge embedding technique, each relay node executes the following sequence of operations: (i) Decryption of the provenance information from the packet, (ii) Update process of the provenance information, and (iii) Encryption of the provenance information. As a result, the total delay on each packet is $T_{total} = h(T_E + T_P)$, where T_E and T_P are the delays introduced by the crypto-primitives (which include decryption and encryption operations) and the provenance embedding operations, respectively.

In this section, we present an enhancement to the edge embedding technique in order to reduce the delay on the packets. The basic idea is that each node, except the source and the destination, has a pair of edges linked to it: one through which the packet is received and the other through which the packet is sent. If the relay node can embed the information of these two edges into the provenance, then it can cover two edges of the path in one-shot. As a consequence, the next node in the path need not modify the provenance. Overall, with this idea, at most half the nodes in the path modify the provenance thereby reducing the delay on the packets. We refer to this scheme as the double-edge embedding technique. To execute double-edge embedding, we formally define a double-edge of a node as below:

Definition 1. Given three nodes $a, b, c \in \mathcal{N}$, we define the two-tuple $(e_{b,a}, e_{a,c})$ as a double-edge of node a if there exists a path $b \to a \to c$.

In the following definition, we introduce an embedding pattern on the packets to capture the participation of various nodes in the embedding process.

Definition 2. On the l-th packet that traverses a path of hop-length h, the binary vector $\mathbf{e}_l = [\mathbf{e}_l(1), \mathbf{e}_l(2), \ldots, \mathbf{e}_l(h)] \in \{0,1\}^h$ is referred to as the embedding pattern on the l-th packet, wherein $\mathbf{e}_l(j) = 1$ if the j-th node modifies the provenance, otherwise $\mathbf{e}_l(j) = 0$.

As an example, when $h = 4$, $\mathbf{e}_l = [0\ 1\ 1\ 0]$ indicates that the second and the third nodes in the path modify the provenance on the l-h packet but not the first and the fourth nodes. With the double-edge embedding technique, we identity the structure of the embedding pattern that minimizes the delay on the packet and also assists the destination in retracing the path traveled by the packet.

Proposition 1. With h denoting the number of hops, the optimal embedding pattern that minimizes the delay on the l-th packet, for some l, such that $1 \leq l \leq N$, is

$$\mathbf{e}_l = \begin{cases} [0\,1\,0\,1\ \ldots\ 0\,1] \in \{0,1\}^h, & \text{if } h \text{ is even;} \\ [0\,1\,0\,1\ \ldots\ 1\,0] \in \{0,1\}^h, & \text{otherwise.} \end{cases}$$

Proof: Owing to double-edge embedding, a necessary condition on the optimal embedding pattern is the absence of consecutive zeros. While this can be achieved in two ways, we choose the one wherein the first component is zero. Since the destination has the knowledge of its preceding node, the path traced by the packet can be recovered despite having $\mathbf{e}_l(h) = 0$ when h is odd. □

A practical solution to follow the embedding pattern of Proposition 1 is to use the hop-counter in the packet structure, and then ask the nodes to modify the provenance depending on the received hop-counter value. In particular, the nodes can be asked to update the bloom filter contents when the received hop-counter value is odd. Using the embedding pattern given in Proposition 1, the total delay incurred by such an embedding technique is

$$T_{total} = hT_E + \left\lfloor \frac{h}{2} \right\rfloor T_P. \tag{19}$$

Note that each node adds a delay of T_E seconds irrespective of its participation in the embedding process since the information on whether to modify the provenance is communicated using the hop-counter, which is also kept confidential along with the bloom filter portion.

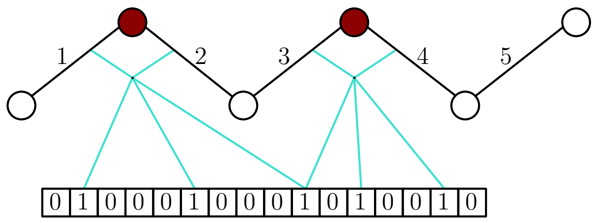

Fig. 11: An example for embedding double-edges in the bloom filter. In this example, the second node embeds the identity of the double-edge '1'-'2', and the fourth node embeds the identity of the double-edge '3'-'4' by using $k = 3$ hash functions into a bloom filter of 16 bits.

4.1 Key Derivation Algorithm for Double-Edges

Similar to the edge embedding technique, we intend to keep the identity of the double-edges of a node private from the rest of the nodes, and only share it with the destination so that it can verify the memberships of the double-edges using the bloom filter contents. Along the lines of Section 3.1, if node b and node c, for $b \neq a$ and $c \neq a$, successfully authenticate with node a, to forward and receive the packet, respectively, then node a derives the secret-key representing the double-edge between node b and node c as $k_{b,a,c} = f_{DE}(k_a, \pi_b, \pi_a, \pi_c)$, where $f_{DE}(\cdot)$ is an appropriate PRF to derive a unique key based on the pseudonyms π_a, π_b, and π_c along with the private key k_a. Meanwhile, since the destination has the list of pseudonyms, and also the list of unique pre-shared keys of all the nodes in the network, it can locally derive the same set of shared secret-keys.

A detailed explanation on the double-edge embedding process is given below. We refer to this scheme as the Deterministic Double-Edge (DDE) embedding scheme since the embedding pattern is a priori decided.

4.2 Deterministic Double Edge Algorithm

Embedding Process: The initial values of the bloom filter and the hop-counter are set to zero. The node i_j, for $1 \leq j \leq h$, extracts the hop-counter value and the bloom filter portion, given by $BF_l = [BF_l(1), BF_l(2), \ldots, BF_l(m)] \in \{0,1\}^m$. Depending on whether the hop-counter value is even or odd, it then uses $k_{i_{j-1},i_j,i_{j+1}}$ (as derived in Section 4.1) to update the bloom filter contents on k random positions as

$$BF_l\left(v^{(r)}_{i_{j-1},i_j,i_{j+1}}\right) = 1, \tag{20}$$

where

$$v^{(r)}_{i_{j-1},i_j,i_{j+1}} = \mathrm{H}\left(k_{i_{j-1},i_j,i_{j+1}}, p_l, n^{(r)}_{i_{j-1},i_j,i_{j+1}}\right), \tag{21}$$

for $1 \leq r \leq k$ such that $\mathrm{H}(\cdot,\cdot,\cdot)$ is a hash function which generates a random number in $[m]$ using the inputs (i) p_l, which is the identity of the l-th packet, (ii) $n^{(r)}_{i_{j-1},i_j,i_{j+1}}$ is the nonce used to generate the r-th index, and (iii) $k_{i_{j-1},i_j,i_{j+1}}$ is the key shared with the destination. After updating the bloom filter, node i_j updates the hop-counter. An example for double-edge embedding technique using bloom filter is shown in Fig. 11. Due to distinct nonce values and distinct identities given to the double-edges, we note that the index values chosen at the output of the hash functions are statistically independent.

Verification and Path Retracing: The total number of double-edges including the destination is $6\binom{n}{3}$, where the term 6 takes care of various double-edge patterns using a given set of three nodes. However, among them, the destination must not appear as either the first node or the second node of the double-edge, and therefore, the total number of valid double-edges is at most $6\binom{n}{3} - 2(n-1)(n-2)$. Thus, the total set of secret-keys at the destination for provenance verification is $\mathcal{K}_{DE} \subset \{k_{b,a,c} = f_{DE}(k_a, \pi_b, \pi_a, \pi_c) \mid a, b, c \in \mathcal{N} \text{ s.t. } b \neq c\}$. Using $\mathcal{K}_{DE}$ and the pre-shared nonce values, the destination verifies the identity of the double-edges with the received bloom filter values, and a list of candidate double-edges, denoted by $\mathcal{DE}_{BF}^{(l)}$, is obtained as

$$\mathcal{DE}_{BF}^{(l)} = \{(e_{b,a}, e_{a,c}) \in \mathcal{E} \times \mathcal{E} \mid BF_l(v_{b,a,c}^{(r)}) = 1, 1 \leq r \leq k\},$$

where

$$v_{b,a,c}^{(r)} = \mathrm{H}\left(k_{b,a,c}, p_l, n_{b,a,c}^{(r)}\right). \tag{22}$$

When m is sufficiently large with respect to $\binom{n}{3}$ and k, the list $\mathcal{DE}_{BF}^{(l)}$ provides only the set of double-edges traced by the packet. However, when m is comparable with respect to $\binom{n}{3}$ and k, the shortlisted candidates of double-edges in $\mathcal{DE}_{BF}^{(l)}$ may have more edges than those participated in forwarding the packet. Similar to the edge embedding technique, the destination recovers the path traced by the l-th packet using a DFS algorithm on the graph formed by $\mathcal{DE}_{BF}^{(l)}$. As a consequence, we define error-rate as the fraction of packets for which the destination recovers more than one path of hop-length h from $\mathcal{DE}_{BF}^{(l)}$.

Although the double-edge embedding technique gives an advantage in terms of delay over the edge embedding technique, note that the destination has to verify more candidates in the bloom filter than in the latter technique.

4.3 False-Positive Analysis of DDE Embedding Technique

Similar to the edge embedding technique, we analyze the error-rate of the deterministic double-edge embedding technique. From first principles, the error-rate is defined as

$$\Pr(E_{fp}) = \sum_{i=1}^{min(m,k\lfloor \frac{h}{2} \rfloor)} \Pr(E_{fp}|C_i)\Pr(C_i), \tag{23}$$

where $\Pr(C_i)$ is the probability that i positions of the bloom filter have been chosen by the double-edges on the path traversed by the packet, and $\Pr(E_{fp}|C_i)$ is the probability that more than one path of hop-length h is recovered at the destination using $\mathcal{DE}_{BF}^{(l)}$ conditioned that i positions of the bloom filter are set. With that $\Pr(C_i)$ is written as

$$\Pr(C_i) = \frac{\binom{m}{i} \sum_{\gamma=0}^{i} ((-1)^{\gamma}) \binom{i}{\gamma} (i-\gamma)^n}{m^{k\lfloor \frac{h}{2} \rfloor}}. \tag{24}$$

Furthermore, $\Pr(E_{fp}|C_i)$ is the probability of false positive given that i positions of the bloom filter are lit by the double-edges of the path traversed by the packet. We can write it as

$$\Pr(E_{fp}|C_i) = 1 - \Pr(\overline{E_{fp}}|C_i) \tag{25}$$

where $\Pr(\overline{E_{fp}}|C_i)$ can be calculated by exhaustively counting all the cases that do not result in false positives. Since handling all possible cases is intractable, the following theorem provides a lower bound on $\Pr(\overline{E_{fp}}|C_i)$ by considering a subset of cases that are guaranteed not to result in false positives.

Theorem 2. For a given value of n, m, h, and k, the term $\Pr(\overline{E_{fp}}|C_i)$ can be lower bounded as

$$\Pr(\overline{E_{fp}}|C_i) > g_{DE}(n, m, h, i, k), \tag{26}$$

where $g_{DE}(n, m, h, i, k)$ is on the Right Hand Side of (34).

Proof: With no knowledge on the network topology, the number of double-edges we need to consider for calculating false positives depends on whether $h+1$ is even or odd. When $h+1$ is even, the number of double-edges, denoted by D, for calculating false positives is $D = \binom{n}{3}3! - \binom{n-1}{2}3! - \frac{h-1}{2}$, where the first term represents the total number of double-edges in a network of n nodes, the second term represents the number of double-edges which includes the destination, and the third term represents the number of double-edges on the main path traversed by the packet. When $h+1$ is odd, the number of double-edges for calculating false positives is $D = \binom{n}{3}3! - \binom{n-1}{2}3! + \binom{n-2}{1} - \frac{h}{2}$, where the third term is the number of double-edges which end at the destination. Since the second node of such double-edges is known to the destination, the term $\binom{n-2}{1}$ takes care of various possible candidates for the first node of the double-edges. Note that the first, the second and the fourth terms are similar to the case when $h+1$ is even. Henceforth, we denote the above set of double-edges by $\bar{\mathcal{DE}}$.

Similar to the proof of Theorem 1, the probability that a double-edge in $\bar{\mathcal{DE}}$ is not lit in the bloom filter is given by $q_i = 1 - \left(\frac{i}{m}\right)^k$. To calculate a lower bound on $\Pr(\overline{E_{fp}}|C_i)$, we consider specific cases of double-edges (being lit in the bloom filter) which are guaranteed not to result in false positives.

Case 1: We consider a scenario wherein no double-edge in $\bar{\mathcal{DE}}$ is lit in the bloom filter. Since the output of the hash functions across double-edges are statistically independent, the probability of this event is given by

$$P_1^{\bar{\mathcal{DE}}} = q_i^D. \tag{27}$$

Case 2: We consider a scenario wherein double-edges that are isolated from the main path are lit in the bloom filter. By using $k' \geq 1$ to denote the number of such double-edges, the probability of such a scenario is given by

$$P_2^{\bar{\mathcal{DE}}} = \sum_{k'=1}^{\binom{n-h-1}{3}3!} \binom{\binom{n-h-1}{3}3!}{k'} q^{D-k'}(1-q)^{k'}. \tag{28}$$

An example for this case is shown in Fig. 12. In general, the number of ways to choose $k' - \kappa$ isolated double-edges from $n - h - 1 - \gamma$ nodes is defined as

$$V(\gamma, \kappa) \triangleq \binom{\binom{n-h-1-\gamma}{3}3!}{k' - \kappa}. \tag{29}$$

Henceforth, in the rest of the proof, we denote the term $q_i^{D-k'}(1-q_i)^{k'}$ as H.

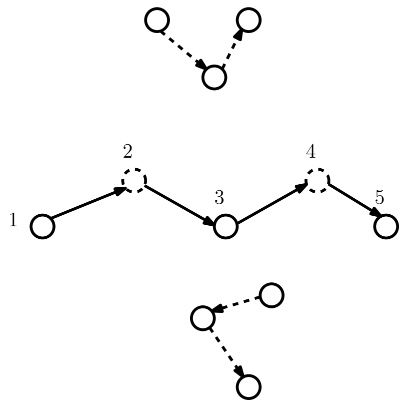

Fig. 12: Case 2 of Theorem 2: Dark double-edges represent the path traversed by the packet, whereas dashed edges represent isolated double-edges.

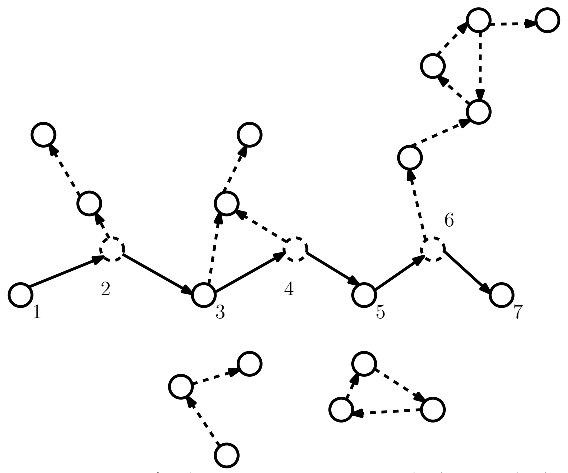

Fig. 13: Case 3 of Theorem 2: Outward directed double-edges that originate from nodes on the main path (dark edges), and isolated double-edges.

Case 3: We consider two types of double-edges: (i) outward directed double-edges that originate from one of the nodes on the main path such that both the second and the third nodes of the double-edge lie outside the main path, and (ii) isolated double-edges which do not have any connection with the main path. This case is illustrated in Fig. 13. Under type-(i), we assume that a node on the main path has at most one outward directed double-edge.

With a total of k' double-edges lit in $\bar{\mathcal{DE}}$, β of them can be of type-(i) and the remaining $k'-\beta$ can be of type-(ii). Under type-(i), the number of ways of selecting the originating nodes from h nodes on the main path $\binom{h}{\beta}$ (excluding the destination). With a given originating node on the main path, the number of ways of choosing a double-edge (directed outward) is $(n-h-1)(n-h-2)$ since the other two nodes lie outside the main path. Thus, the total number of ways of choosing β double-edges of type-(i) is $\binom{h}{\beta}((n-h-1)(n-h-2))^{\beta}$. With that the remaining $k'-\beta$ isolated double-edges can be chosen in $V(0,\beta)$ ways. Overall, the probability of this scenario is

$$P_3^{\bar{\mathcal{DE}}} = \sum_{k'=1}^{D} \sum_{\beta=1}^{min((h),k')} \binom{h}{\beta}((n-h-1)(n-h-2))^{\beta} V(0,\beta)H. \tag{30}$$

Note that (30) is valid when $h+1$ is either even or odd.

Case 4: We consider two types of double-edges: (i) Double-edges such that the first edge is on the main path, in the direction of the packet-flow, whereas the second edge is directed outward connecting a node that lies outside the main path, and (ii) isolated double-edges that do not have any connection with the main path. An example for this case is captured in Fig. 14.

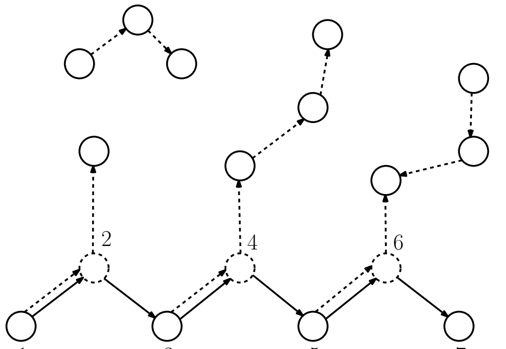

Fig. 14: Case 4 of Theorem 2: Outward directed double-edges such that the first edge is on the main path (dark edges) in the direction of the packet-flow whereas the second edge is directed outwards, and isolated double-edges.

With a total of k' double-edges lit in $\bar{\mathcal{DE}}$, β of them can be of type-(i) and the remaining $k'-\beta$ can be of type-(ii). Under type-(i), the total number of ways of selecting β nodes as the second node of the double-edges is $\binom{h-1}{\beta}$ (excluding the source and the destination). Further, the number of ways of selecting the third node which lies outside the main path is $(n-h-1)^{\beta}$. Thus, the total number of ways of choosing β double-edges of type-(i) is $\binom{h-1}{\beta}(n-h-1)^{\beta}$. With that the remaining $k'-\beta$ isolated double-edges can be chosen in $V(0,\beta)$ ways. Overall, the probability of this scenario is

$$P_4^{\bar{\mathcal{DE}}} = \sum_{k'=1}^{D} \sum_{\beta=1}^{min((h-1),k')} \binom{h-1}{\beta}(n-h-1)^{\beta} V(0,\beta)H. \tag{31}$$

Note that (31) is valid when $h+1$ is either odd or even.

Case 5: We consider two types of double-edges: (i) Double-edges such that the first edge merges with a node on the main path whereas the second edge is on the main path, however, in the direction opposite to that of packet-flow, and (ii) isolated double-edges that are not connected to the edges on the main path. An example for this case is shown in Fig. 15.

With a total of k' double-edges lit in $\bar{\mathcal{DE}}$, β of them can be of type-(i) and the remaining $k'-\beta$ can be of type-(ii). Similar to the counting arguments used in Case 4, the probability of the events in this scenario can be written as

$$P_5^{\bar{\mathcal{DE}}} = \sum_{k'=1}^{D} \sum_{\beta=1}^{min((h-1),k')} \binom{h-1}{\beta}(n-h-1)^{\beta}(V(0,\beta))H. \tag{32}$$

Note that (32) is also valid when $h+1$ is either odd or even.

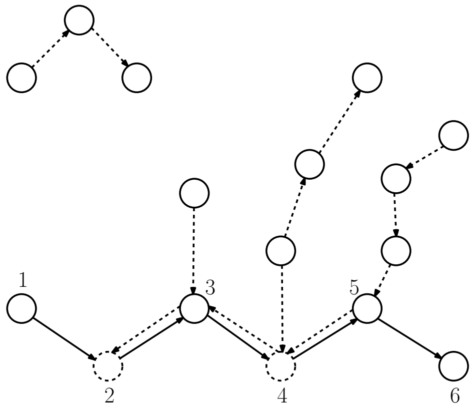

Fig. 15: Case 5 of Theorem 2: Inward directed double-edges such that the first edge merges at a node on the main path (dark edges) whereas the second edge is on the main path against the direction of packet-flow.

Case 6: We consider three types of double-edges: (i) Double-edges such that the first edge merges with a node on the main path whereas the second edge is on the main path either in or opposite to the direction of packet-flow, (ii) Double-edges that merge with the main path such that only the third node of the double-edge is on the main path (the first two nodes of the double-edge do not lie on the main path), and (iii) isolated double-edges that are not connected to the main path. An example for this case is in Fig. 16.

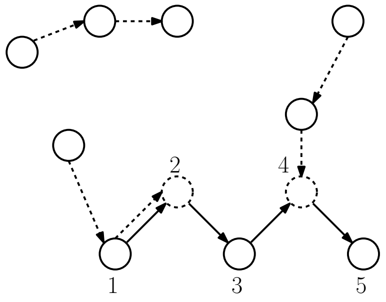

Fig. 16: Case 6 of Theorem 2: Inward directed double-edges such that the first edge merges at a node on the main path (dark edges) whereas the second edge is on the main path either in the direction of the packet-flow or against it. This case also considers inward directed double-edges that terminate at nodes on the main path.

When $h+1$ is odd, double-edges of type-(i) can merge at any of the $\frac{h-2}{2}$ nodes (excluding the source node) on the main path, whereas double-edges of type-(ii) can merge at any of the $\frac{h}{2}$ nodes (excluding the source node) on the main path. Similarly, when $h+1$ is even, double-edges of type-(i) and type-(ii) can merge at any of the $\lfloor\frac{h}{2}\rfloor - 1$ and $\lfloor\frac{h}{2}\rfloor$ nodes (excluding the source node) on the main path, respectively. In addition to the above options, source node can be a part of both type-(i) and type-(ii) since the resultant path does not lead to false positives.

With a total of k' double-edges lit in the bloom filter, let j of them be of type-(i), and β be them be of type-(ii). We consider $j \geq 0$ and $\beta > 0$ to avoid over counting double-edges in Case 5. The total number of ways to select j nodes of type-(i) and β nodes of type-(ii) are B and A respectively, where $B = \binom{\lfloor\frac{h}{2}\rfloor-1}{j}$ and $A = \binom{\lfloor\frac{h}{2}\rfloor}{\beta}$, when $h+1$ is even, and $B = \binom{\frac{h-2}{2}}{j}$ and $A = \binom{\frac{h}{2}}{\beta}$, when $h+1$ is odd. Under type-(i), the first node of the double-edge can be selected in $n-h-1$ ways. With that the number of double-edges under type-(i) can be chosen in $B(n-h-1)^j$ ways. After merging on a node on the main path, double-edges can be formed in two ways, either in the direction of packet-flow or against it. As a result we have to multiply the number of possible double-edges under type-(i) by 2^j. However, when $h+1$ is even, the double-edge of type-(i) terminating at the penultimate node of the main path cannot have its second edge in the direction of the packet flow. Therefore, we multiply the number of possible double-edges under type-(i) by $D = 2^{j-(h \mod 2)}$.

Under type-(ii), the first node and the second node of the double-edge can be respectively selected in $n-h-1$ and $n-h-2$ ways. With that the number of double-edges under type-(ii) can be chosen in $A(n-h-1)^{\beta}(n-h-2)^{\beta}$ ways. In addition to j double-edges of type-(i) and β double-edges of type-(ii), the source node can have one of the following possibilities of double-edges: double-edge of type-(ii), double-edge of type-(i), double-edges of both type-(i) and type-(ii), neither type-(i) nor type-(ii).

Overall, including the above possibilities of double-edges, the probability of this case is given by $P_6^{\bar{\mathcal{DE}}}$, as given in (33), where the value of the pair $\{a,b\}$ can be $\{1,1\}$,$\{1,0\}$,$\{2,1\}$, and $\{0,0\}$, which takes care of one of the possible types of double-edges on the source, and $V(0, \beta + j + a)$ takes care of the ways to pick $k' - \beta - j - a$ double-edges that are isolated from the main path.

From Case 1 to Case 6, we have considered only a subset of cases which are guaranteed not to generate false positives. As a result, given that i positions in the bloom filter were lit by the double-edges of the main path, we can lower bound the probability of not witnessing false positives as

$$\Pr(\overline{E_{fp}} \mid C_i) > \sum_{j=1}^{6} P_j^{\bar{\mathcal{DE}}}, \tag{34}$$

where $P_1^{\bar{\mathcal{DE}}}, P_2^{\bar{\mathcal{DE}}}, \ldots, P_6^{\bar{\mathcal{DE}}}$ are respectively given in (27), (28), (30), (31), (32), (33). □

By substituting the above lower bound given by Theorem 2 in (25), we get an upper bound on the probability of false positives given that i positions in the bloom filter are lit by the double-edges on the main path. Furthermore, substituting this upper bound and the exact expression of $\Pr(C_i)$ in (23), we get an upper bound on the probability of false positives of the double-edge embedding scheme.

Similar to the edge embedding scheme, we verify the tightness of the above upper bound by comparing it with the results on error-rates obtained through simulations. The plots, which are presented in Fig. 17 and Fig. 18, highlight that the inferences made in the case of edge embedding technique can be extended to the double-edge embedding technique as well.

4.4 DDE Embedding with Relaxed Recovery Constraints

The delay expression in (19) is computed with the assumption that the destination recovers the path traced by

$$P_6^{\bar{\mathcal{DE}}} = \sum_{k'=1}^{D} \sum_{\beta>0} \sum_{j\geq 0} \sum_{\{a,b\}} ABD(n-h-1)^{\beta+j+a}(n-h-2)^{\beta+b}(V(0,\beta+j+a))H \quad (33)$$

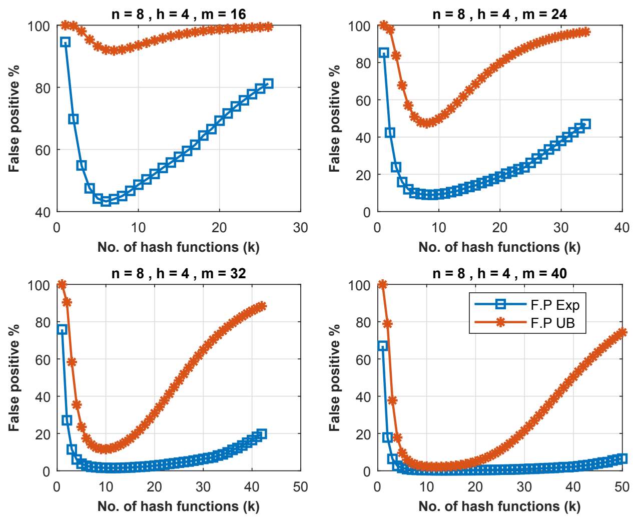

Fig. 17: Comparison between the error-rate of double-edge embedding scheme and the upper bound for $[n, h] = [8, 4]$.

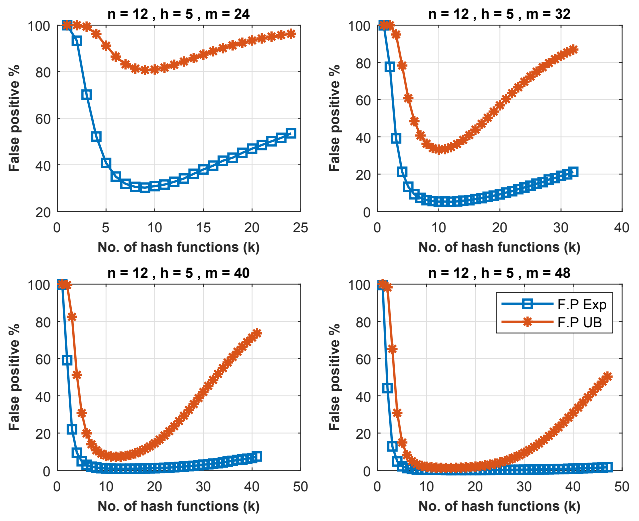

Fig. 18: Comparison between the error-rate of double-edge embedding scheme and the upper bound for $[n, h] = [12, 5]$

every packet. However, when the destination has relaxed constraint to determine the provenance after observing a number of packets, say $N > 1$, then the delay introduced on each packet can be much lower than when $N = 1$. This reduction can be accomplished by distributing the optimal embedding pattern in Proposition 1 across $N = \lfloor \frac{h}{2} \rfloor$ packets. By distributing the provenance embedding process over N packets, delay incurred by the embedding process per packet is $T_{total} = hT_E + T_P$. This is because each node has to execute the crypto-primitives irrespective of whether it is embedding the provenance, and out of the h nodes, only one node embeds the provenance on a packet, which in turn contributes T_p seconds. When $N \geq \lfloor \frac{h}{2} \rfloor$, only the

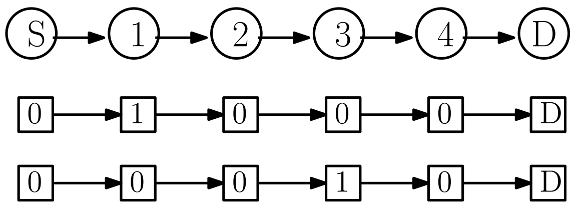

Fig. 19: When the destination has relaxed constraint to determine the provenance over two packets, the embedding pattern $[0\ 1\ 0\ 1\ 0]$ can be distributed over two packets as $\mathbf{e}_1 = [0\ 1\ 0\ 0\ 0]$ and $\mathbf{e}_2 = [0\ 0\ 0\ 1\ 0]$ to reduce the delay.

first $\lfloor \frac{h}{2} \rfloor$ packets need to carry the provenance, whereas the rest of $N - \lfloor \frac{h}{2} \rfloor$ need not. Therefore, the destination needs to wait for at least $\lfloor \frac{h}{2} \rfloor$ packets to determine the path traced by the packets. An example for the embedding patterns of the 5-hop case is as shown in Fig. 19. A brief description of the embedding process with relaxed recovery constraints is given below when $N = \lfloor \frac{h}{2} \rfloor$.

Embedding Process and Path Retracing: The node i_j, for $1 \leq j \leq h$, extracts the hop-counter and the bloom filter portion BF_l. Based on the hop-counter value and the value of l, then node i_j embeds the identity of its double-edge, and then updates the hop-counter before forwarding the packet to node i_{j+1}. This way, the destination receives BF_l, which is modified by at most one node in the path. Using the bloom filter values on all the N packets, i.e., $\{BF_l \mid 1 \leq l \leq N\}$, the destination constructs N sets of double-edges, denoted by $\{\mathcal{DE}_{BF}^{(l)} \mid 1 \leq l \leq N\}$, where $\mathcal{DE}_{BF}^{(l)}$ is the list of shortlisted double-edges from BF_l. Subsequently, the union of the shortlisted double-edges is obtained as $\mathcal{DE}_{union} = \cup_{l=1}^{N} \mathcal{DE}_{BF}^{(l)}$. Finally, by feeding $\mathcal{DE}_{union}$ to the path retracing algorithm, the destination determines the path traced by the set of N packets.

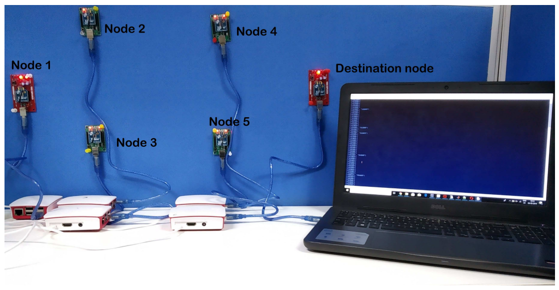

Fig. 20: Experimental setup of a network comprising six nodes using Raspberry Pi 3+ and Digi XBee S2C devices.

5 Experimental Results

In this section, we demonstrate the efficacy of the proposed provenance embedding techniques on a test bed involving a network of several XBee devices. The test bed setup as shown in Fig. 20 consists of $n = 6$ nodes, wherein each node uses a Raspberry Pi 3+ and a Digi XBee S2C for computation and communication purposes, respectively. To implement $H(\cdot, \cdot, \cdot)$ in the provenance embedding process, we use the standard SHA-256 protocol.

Our metrics of interest are the error-rates in provenance recovery and the average latency offered per packet. In this section, the total delay introduced by a provenance embedding technique is referred to as the latency offered on the packets. To measure the error-rates, we fix the number of hops, and then route a sequence of 10^5 packets by varying the paths. Subsequently, using the bloom filter observations on these 10^5 packets, we compute the error-rates as defined in (4). We define delay at a given node as the time taken to transfer a packet from XBee's receiving buffer to XBee's transmitting buffer after executing the required computations at Raspberry Pi 3+. The average delay numbers are measured by routing an ensemble of 3000 packets through a given node in the network. Using the experiments, we observe that the average delay introduced by the crypto-primitives at a node is $35ms$, whereas the average delay incurred when executing the combination of crypto-primitives and the provenance embedding process is more than $35ms$, and this additional delay is due to the implementation of k hash functions on Raspberry Pi 3+. For instance, we observe that the additional delay contributed by implementing the sequence of k hash functions range from $4ms$ to $18ms$ when k ranges from 1 to 15.

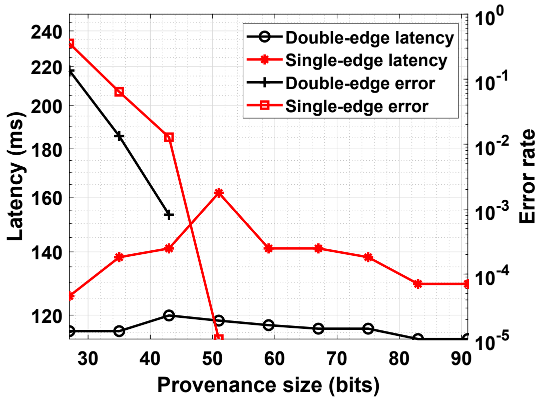

Fig. 21: Comparison between edge and double-edge embedding in terms of error-rate and latency on a network with $n = 6$ and $h = 3$. For a given size of the bloom filter, the value of k that minimizes the error-rate is determined, and the corresponding value of k is used to compute latency.

We compare the error-rates and the latency offered by the proposed edge and double-edge embedding techniques when the destination has strict recovery constraints to determine the provenance. Their comparisons are presented in Fig. 21, Fig. 22, and Fig. 23, for hop-lengths $h = 3, 4,$ and 5, respectively. In each of the above figures, we vary the provenance size from 27 bits to 91 bits in steps of 8, which includes 3 bits for the hop-counter and the rest for the bloom filter. For a given bloom filter size, we compute the optimal value of k that minimizes the upper bound on the error-rates in Theorem 1 and Theorem 2. Using the corresponding value of k, the average delay contributed

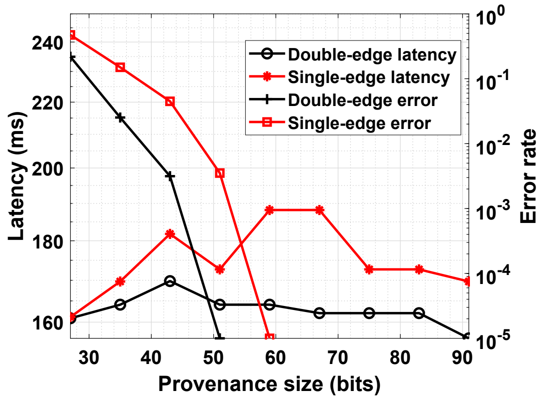

Fig. 22: Comparison between edge and double-edge embedding in terms of error-rate and latency on a network with $n = 6$ and $h = 4$.

by the provenance embedding process at a given node is computed. Subsequently, the total delay is computed using the number of nodes that modify the bloom filter and the number of hops. In each of Fig. 21, Fig. 22, and Fig. 23, the error-rates are plotted on the right-side of the y-axis, whereas the average total latency offered are presented on the left-side of the y-axis. As expected, all the three figures confirm that the average latency offered by double-edge embedding is lower than that of edge embedding. In Fig. 21, we observe that the benefits in latency numbers are not the same with respect to the provenance size. This behavior is attributed to the fact that the value of $k \in \{1, 2, \ldots, m\}$ that minimizes the error-rate depends on the bloom filter size, and as result, the latency numbers change since the processing time at each relay node depends on the number of hash functions used to embed the provenance. Similar behavior can also be observed with $h = 4$ in Fig. 22 and $h = 5$ in Fig. 23, wherein the latency numbers peak at the intermediate values of the provenance size.

Interestingly, the plots show that double-edge embedding outperforms edge embedding in terms of error-rates as well. Although the number of double-edges to be verified using the bloom filter is much larger than the number of edges, the process of searching two successive edges at a time in the DFS algorithm reduces the number of candidate paths when compared with that of edge embedding.

5.1 DDE Embedding with Relaxed Recovery Constraints

In this section, we present the latency numbers of the DDE embedding techniques when the destination has relaxed constraint to learn the provenance after observing N packets. In Fig. 24, we present the provenance size on the x-axis and the corresponding latency numbers on the y-axis. For a given provenance size, the latency numbers are obtained by solving for k to achieve the error-rate of $p_{error} \leq 10^{-2}$. From the plots, it is evident that with $N = 2$, in order to achieve an error-rate of 10^{-2}, the DDE embedding method with $N > 1$ offers lower latency than with $N = 1$, however at the cost of larger provenance size. We note that this behavior is attributed to higher false-positive rates with $N > 1$ owing to shortlisting of union of double-edges over the N packets.

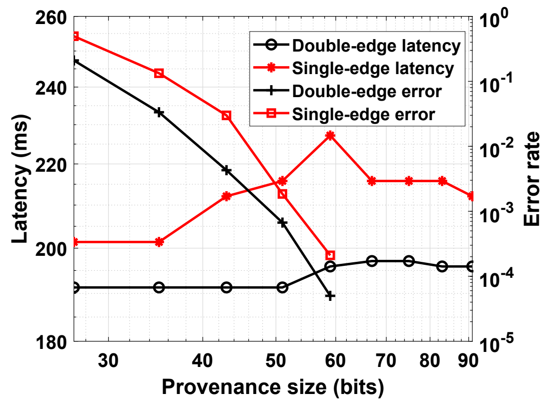

Fig. 23: Comparison between edge and double-edge embedding in terms of error-rate and latency on a network with $n = 6$ and $h = 5$.

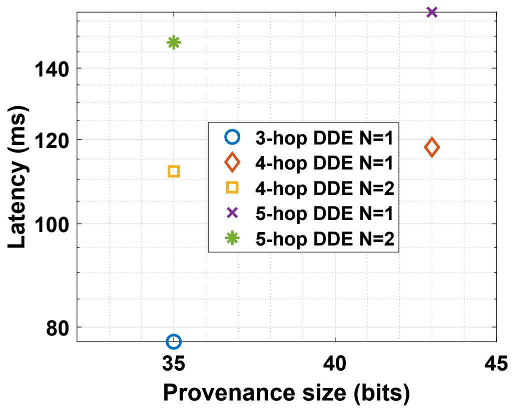

Fig. 24: Latency offered by the DDE embedding technique with $N = 1$ and $N > 1$ to achieve an error-rate of 10^{-2}.

We do not present experiment results with $N > 2$ since the minimum value of $\lfloor \frac{h}{2} \rfloor$ is 2 in our experimental settings.

6 Security Analysis

In this section, we analyze the vulnerabilities of edge and double-edge embedding methods against impersonation attacks that could be executed by an external attacker after compromising one of the nodes in the network. As described in Section 2.1, the objective of the external attacker is to manipulate the provenance information on the compromised node so as to misguide the destination to recover a path other than the one traveled by the packet. To mitigate such attacks, we have proposed the use of private keys as identities of the underlying edges (and the double-edges) of the nodes. However, although the identities of the edges and double-edges are private, we observe that the compromised node can randomly generate k bloom filter positions, which has a non-zero probability of collision with the positions chosen by another legitimate edge (or double-edge) in the network. In such events of collision, the destination is likely to recover a different path in the network, which in turn is not a desirable event in the network model.

Definition 3. A compromised node is said to successfully execute a *perfect impersonation attack* if the destination recovers exactly one path from the bloom filter such that the recovered path differs from the actual path at the position of the compromised node.

For instance, if the actual path traveled by the packet is $i_1 \rightarrow i_2 \rightarrow i_3 \rightarrow \ldots \rightarrow i_h$, and if node i_2 would like to impersonate as node i_2', for some $i_2' \in \mathcal{N}$, then in order to execute a successful attack the destination must recover only one path, i.e., $i_1 \rightarrow i_2' \rightarrow i_3 \rightarrow \ldots \rightarrow i_h$, from the bloom filter. Henceforth, we define the success-rate of the perfect impersonation attack as the fraction of packets for which the attacker is able to successfully misguide the destination in recovering a path which differs from the actual path at the position of the compromised node. The following propositions showcase results on vulnerabilities of edge and double-edge embedding methods against perfect impersonation attacks.

Proposition 2. With the edge embedding method, the success-rate of perfect impersonation attack is zero.

Proof: The result is straightforward to prove. □

Proposition 3. With the double-edge embedding method, the success-rate of perfect impersonation attack is bounded away from zero.

Proof: With double-edge embedding, the success-rate of perfect impersonation attack depends on whether the attacker has compromised a node that embeds its double-edge or the one that skips the embedding process. Since the information on the hop-counter is kept confidential by the legitimate nodes, an external attacker may compromise one of these two types of nodes. In the case of compromising a node that skips the embedding process, it is straightforward to note that the attacker cannot execute perfect impersonation attack since the preceding and the succeeding nodes will correctly embed their double-edges thereby embedding a path in the provenance. As a result, we consider the case when the attacker compromises a node that embeds the identity of its double-edges. Suppose that node b, for some $b \in \mathcal{N}$, is compromised by an external attacker, and node b was scheduled to embed the identity of the double-edge $(e_{a,b}, e_{b,c})$, where $e_{a,b}$ is the directed edge connecting node a and node b, and $e_{b,c}$ is the directed edge connecting node b and node c. Furthermore, suppose that the attacker attempts to impersonate node d, for some $d \neq b$. Since the attacker does not have the identity of the double-edge $(e_{a,d}, e_{d,c})$ (since it is private to node d), it attempts to randomly generate k statistically independent index values in the bloom filter with uniform distribution. In such a case, the success-rate of perfect impersonation attack is the probability with which the index values generated by node d using the double-edge $(e_{a,d}, e_{d,c})$ coincides with that of the randomly generated index values by the attacker. It is important to note that the success-rate of perfect impersonation attack also depends on the number of index values chosen by the other double-edges on the path. In particular, the success-rate can be formally written as

$$p_{success} = \sum_{i=1}^{(\lfloor \frac{h}{2} \rfloor)k} \Pr(C_i) \sum_{j=0}^{k} \binom{m-i}{j} (p_{match})^2, \qquad (35)$$

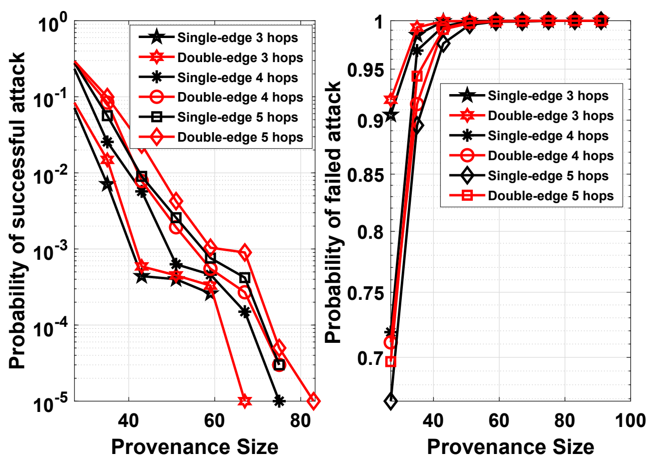

Fig. 25: Comparison of success-rate and failure-rate of impersonation attack for $n = 6$ and various values of h. For each value of m, the optimal value k is chosen based on the results in Theorem 1 and Theorem 2.

where $\Pr(C_i)$ is the probability that i positions, for $1 \leq i \leq (\lceil \frac{h}{2} \rceil - 1)k$, of the bloom filter have been set by the double-edges contributed by the $(\lceil \frac{h}{2} \rceil - 1)$ legitimate nodes in the path, the term p_{match} is the probability that the attacker chooses k index values in the bloom filter such that j distinct index values, for $0 \leq j \leq k$, are chosen outside the set of i index values (which are chosen by the other double-edges) and the remaining $k - j$ index values are chosen at any of those $i+j$ index values of the bloom filter. Note that the term p^2_{match} appears in (35) owing to statistical independence between the index values chosen by the attacker and that of node d. Furthermore, it can be shown that p_{match} is lower bounded by $\frac{\binom{k}{j}((j!)(i^{(k-j)}))}{m^k}$. Therefore, $p_{success}$ can be lower bounded by

$$\sum_{i=1}^{(\lceil \frac{h}{2} \rceil - 1)k} \Pr(C_i) \sum_{j=0}^{k} \binom{m-i}{j} \left(\frac{\binom{k}{j}((j!)(i^{(k-j)}))}{m^k} \right)^2 . \quad (36)$$

Thus, we have shown that the success-rate of the perfect impersonation attack on double-edge embedding method is bounded away from zero. □

The intuition behind the results in Propositions 2 and 3 is that the edge embedding method adds redundancy in conveying the path information to the destination, and as a result, the destination cannot be misguided to recover another path even if the attacker manages to impersonate the identity of another edge in the bloom filter. On the other hand, the double-edge embedding method adds no redundancy owing to which the attacker has non-zero probability of successfully executing the perfect impersonation attack by swapping a double-edge of the path with another double-edge in the network.

6.1 Experimental Results on Impersonation Attacks

In this section, we present experimental results to analyze the success- and failure-rate of impersonation attack on edge and double-edge embedding methods. To evaluate the effect of the attack, we assume that the value of k is already optimized to minimize the error-rate as a function of n, h and m (similar to the results presented in Section 5). To generate the results, we assume that one of the n nodes is compromised by an external attacker, and the compromised node, instead of using the identity of its edge (or the double-edge), uses a random identity to generate k index values in the bloom filter. Meanwhile, the other nodes on the path embed the identities of their edges (or double-edges) in a legitimate fashion as per the protocol. As a result, the destination witnesses one the following events when recovering the path information from the bloom filter: Type-one: No path (of hop-length as indicated by the hop counter) is recovered, Type-two: Only one path (of hop-length as indicated by the hop counter) is recovered that is other than the path traveled by the packet, and Type-three: More than one path (of hop-length as indicated by the hop counter) is recovered. Among the above three events, we are interested in computing the fraction of events of Type-two, wherein the destination successfully recovers only one path that is other than the path traveled by the packet. When m is sufficiently large the events in this case are due to perfect impersonation attack, wherein the recovered path differs from the actual path only at the position of the compromised node. However, when m is not large, then the events in this case are due to *relaxed* form of impersonation attack, wherein the recovered path differs from the actual path at more than one node. Such events are formed because the attacker's index values in the bloom filter coincide with that of a legitimate edge (or a double-edge) in the network, and simultaneously, other edges (or double-edges) are also lit in the bloom filter due to false positives, thereby contributing a single path to the destination. In short, events of Type-two capture the success-rate of the attacker in misguiding the destination to recover a different path albeit not knowing the identity of other nodes.

Other than computing the success-rate, we are also interested in computing the fraction of events of Type-one and -three, as they assist the destination to detect an impersonation attack. Note that without the impersonation attack, at least one path is recovered from the bloom filter, and moreover, the error-rate is *a priori* optimized to a small non-zero number. As a result, computing the fraction of union of events of Type-one and -three, helps the destination to quantify the failure-rate of the impersonation attack. In Fig. 25, we have plotted the success-rate and the failure-rate of the impersonation attack for both edge and double-edge embedding methods with various values of m, k and h. In particular, the values of m, k and h are as chosen in Fig. 21-23. The plots on the left-side of Fig. 25 show that with edge embedding, the success-rate is dominated by the events of relaxed form of impersonation attacks, whereas with double-edge embedding the success-rate is a combination of perfect as well as relaxed form of impersonation attacks. The plots also show that edge embedding is more resilient to impersonation attack than DE embedding, and this behavior in the order also corroborates with the results in Propositions 2 and 3. With respect to failure-rates, we show through the plots on the right-side of Fig. 25 that impersonation attacks can be detected at a high rate both in edge embedding and DE embedding when the bloom filter size is sufficiently large.

7 Summary and Conclusions

To conclude, we have proposed new provenance embedding algorithms while facilitating low-latency routing of packets over a multi-hop network. We have derived bounds on the error-rates of the proposed techniques in order to select an appropriate number of hash functions for a given number of hops, bloom filter size, and the total number of nodes. We have demonstrated the latency benefits of the double-edge embedding ideas on a test bed of XBee devices, and have also presented a security analysis to asses their vulnerabilities against impersonation attacks. We observe that the proposed technique does not provide 50% reduction in latency although at most half the nodes skip the provenance embedding process. This behavior is attributed to the fact that majority of the delay is contributed by executing the crypto-primitives at each node. Therefore, for future research, we intend to explore the idea of skipping crypto-primitves at some nodes to further reduce the latency, and study the associated trade-offs in security features.

Acknowledgments

This work was supported by the Indigenous 5G Test Bed project from the Department of Telecommunications, Ministry of Communications, New Delhi, India.

References

[1] N. A. Johansson, Y. E. Wang, E. Eriksson and M. Hessler, "Radio Access for Ultra-Reliable and Low-Latency 5G Communications," in the Proc. of *2015 IEEE International Conference on Communication Workshop (ICCW)*, London, 2015, pp. 1184–1189.

[2] K. C. Dey, A. Rayamajhi, M. Chowdhury, P. Bhavsar, and J. Martin, "Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) Communication in a Heterogeneous Wireless Network–performance Evaluation," *in Transportation Research Part C: Emerging Technologies*, vol.68, pp. 168-184, 2016.

[3] A. Ramachandran, K. Bhandankar, M. B. Tariq, and N. Feamster, "Packets with Provenance," *Georgia Tech CSS Technical Report*, GT-CS-08-02, 2008.

[4] D. Sy and L. Bao, "Captra: Coordinated Packet Traceback," *in ACM/IEEE 5th International Conference on Information Processing in Sensor Networks*, pp. 152-159, 2006.

[5] S. R. Hussain, C. Wang, S. Sultana, and E. Bertino, "Secure Data Provenance Compression using Arithmetic Coding in Wireless Sensor Networks," *in IEEE 33rd International Performance Computing and Communications Conference (IPCCC)*, pp. 1-10, 2014.

[6] C. Wang, S. R. Hussain, and E. Bertino, "Dictionary Based Secure Provenance Compression for Wireless Sensor Networks," *in IEEE Trans. on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 405-418, 2016.

[7] H. S. Lim, Y. S. Moon, and E. Bertino, "Provenance-based Trustworthiness Assessment in Sensor Networks," *in Proceedings of the ACM 7th International Workshop on Data Management for Sensor Networks*, pp. 2-7, 2010.

[8] M. Keller, J. Beutel, and L. Thiele, "How was Your Journey?: Uncovering Routing Dynamics in Deployed Sensor Networks with Multi-hop Network Tomography," *SenSys*, pp. 15-28, 2012.

[9] Z. Liu, Z. Li, M. Li, W. Xing, and D. Lu, "Path Reconstruction in Dynamic Wireless Sensor Networks using Compressive Sensing," *in IEEE/ACM Trans. on Networking*, vol. 24, no. 4, pp. 1948-1960, 2016.

[10] X. Lu, D. Dong, X. Liao, and S. Li, "Pathzip: Packet Path Tracing in Wireless Sensor Networks," *in IEEE 9th International Conference on Mobile Ad-Hoc and Sensor Systems (MASS 2012)*, pp. 380-388, 2012.

[11] J. Xu, X. Zhou, and F. Yang, "Traceback in Wireless Sensor Networks with Packet Marking and Logging," *in Frontiers of Computer Science*, Springer, no. 03, pp. 308-315, 2011.

[12] S. Sultana, E. Bertino, and M. Shehab, "A Provenance based Mechanism to Identify Malicious Packet Dropping Adversaries in Sensor Networks," in *International Conference on Distributed Computing Systems Workshops*, Minneapolis, USA, June 2011, pp. 332–338.

[13] B. Shebaro, S. Sultana, S. Reddy Gopavaram, and E. Bertino, "Demonstrating a Lightweight Data Provenance for Sensor Networks," *in ACM Conference on Computer and Communications Security*, pp. 1022-1024, 2012.

[14] S. Sultana, G. Ghinita, E. Bertino, and M. Shehab, "A Lightweight Secure Scheme for Detecting Provenance Forgery and Packet Drop Attacks in Wireless Sensor Networks," *in IEEE Transactions on Dependable and Secure Computing*, no. 3, pp. 256-269, 2015.

[15] S. M. I. Alam and S. Fahmy, "A Practical Approach for Provenance Transmission in Wireless Sensor Networks," *Ad Hoc Networks*, no. 16, pp. 28–45, 2014

[16] M. Klonowski and A. M. Piotrowska, "Light-weight and secure aggregation protocols based on bloom filters," *Computers and Security*, no. 72, pp. 107–121, 2018.

[17] S. Suhail, M. Abdellatif, S. R. Pandey, A. Khan, and C. S. Hong, "Provenance-enabled packet path tracing in the RPL-based internet of things," available online at *arXiv:1811.06143*, 2018.

[18] C. Wang, W. Zheng, and E. Bertino, "Provenance for Wireless Sensor Networks: A Survey," *in Data Science and Engineering*, Springer, vol. 01, no. 3, pp. 189-200, 2016.

[19] J. Naous, M. Walfish, A. Nicolosi, D. Mazie'res, M. Miller, and A. Seehra, "Verifying and enforcing network paths with ICING," in the Proc. of *ACM CoNEXT*, 2011.

[20] T. H.-J. Kim, C. Basescu, L. Jia, S. B. Lee, Y.-C. Hu, and A. Perrig, "Lightweight source authentication and path validation," in the Proc. of *ACM SIGCOMM*, 2014.

[21] H. Wang, C. Qian, Y. Yu, H. Yang, and S. S. Lam, "Practical network-wide packet behavior identification by ap classifier," in the Proc. of *ACM CoNEXT*, 2015.

[22] E. Shi and A. Perrig, "Designing secure sensor networks," in *IEEE Wireless Communications*, vol. 11, no. 6, pp. 38–43, Dec. 2004.

[23] F. Liu, X. Cheng and D. Chen, "Insider attacker detection in wireless sensor networks," *IEEE INFOCOM 2007*, Anchorage, AK, 2007, pp. 1937-1945.

[24] M. Imran, H. Hlavacs, F. A. Khan, S. Jabeen, F. G. Khan, S. Shah, and M. Alharbi, "Aggregated provenance and its implications in clouds," in *Future Generation Computer Systems*, vol. 81, pp. 348–358, April 2018.

[25] M. Raya and J. P. Hubaux, "Securing vehicular ad hoc networks," *Journal of Computer Security*, vol. 15, no. 01, pp. 39–68, 2007

[26] V. Kumar, H. Li, J. M. Park, K. Bian, and Y. Yang, "Group Signatures with Probabilistic Revocation: A Computationally-Scalable Approach for Providing Privacy-Preserving Authentication," in the Proc. of *ACM SIGSAC Conference on Computer and Communications Security*, pp. 1334–1345, Oct. 2015

J. Harshan is an Assistant Professor in the Department of Electrical Engineering, IIT Delhi. His research interests include wireless networks, security, and coding theory.

Amogh Vithalkar received the B.Tech. degree in Electronics and Communication from IIIT Delhi, India. He is currently a project assistant at IIT Delhi, working on security for connected devices in 5G. His research interests include wireless networks, security, radar signal processing, and antenna design.

Naman Jhunjhunwala is pursuing B.Tech. degree in the Department of Mathematics. His research interests include machine learning, algorithms and probability.

Manthan Kabra is pursuing B.Tech. degree in the Department of Mathematics, IIT Delhi. His research interests include applied probability, statistics, machine learning and discrete mathematics.

Praful Manav is pursuing B.Tech. degree in the Department of Electrical Engineering, IIT Delhi. His interests include signal processing, communication systems, applied probability and control systems.

Yih-Chun Hu is an Associate Professor in the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, USA. His research interests include network security and wireless networks.