Multi-Path-Based Avoidance Routing in Wireless Networks

Kazuya Sakai*, Min-Te Sun†, Wei-Shinn Ku‡, Jie Wu§, and Ten H Lai¶
*Department of Information and Communication Systems, Tokyo Metropolitan University,
6-6 Asahigaoka, Hino, Tokyo 191-0065, Japan.
†Department of Computer Science and Information Engineering, National Central University, Taoyuan 320, Taiwan.
‡Department of Computer Science and Software Engineering, Auburn University, Auburn, Alabama 36849–5347.
§Department of Computer and Information Sciences, Temple University, 1925 N. 12th St. Philadelphia, PA 19122.
¶Department of Computer Sciences and Engineering, The Ohio State University, 2015 Neil Ave., Columbus, Ohio 43120.
ksakai@tmu.ac.jp, msun@csie.ncu.edu.tw, weishinn@auburn.edu, jiewu@temple.edu, lai.1@osu.edu

***Abstract*—The speedy advancement in computer hardware has caused data encryption to no longer be a 100% safe solution for secure communications. To battle with adversaries, a countermeasure is to avoid message routing through certain insecure areas, e.g., malicious countries and nodes. To this end, avoidance routing has been proposed over the past few years. However, the existing avoidance protocols are single-path-based, which means that there must be a safe path such that no adversary is in the proximity of the whole path. This condition is difficult to satisfy. As a result, routing opportunities based on the existing avoidance schemes are limited. To tackle this issue, we propose an avoidance routing framework, namely Multi-Path Avoidance Routing (MPAR). In our approach, a source node first encodes a message into k different pieces, and each piece is sent via k different paths. The destination can assemble the original message easily, while an adversary cannot recover the original message unless she obtains all the pieces. We prove that the coding scheme achieves perfect secrecy against eavesdropping under the condition that an adversary has incomplete information regarding the message. The simulation results validate that the proposed MPAR protocol achieves its design goals.**

I. Introduction

Cryptography is commonly used to protect data privacy in computer networks. However, advances in computer hardware have made computing power more and more accessible and inexpensive [1]; data encryption is no longer a perfect solution for network security because computing server-encrypted data can still be recovered in a reasonable amount of time with the key being unknown [2]. The compromised encrypted data can result in destructive and devastating consequences. For example, the successful cryptanalysis of the Enigma machines by the British Intelligence became a determining factor in the Axis's defeat in World War II [3]. Moreover, the software implementation of cryptographic protocols, if not done carefully, may be seriously flawed [4]. Should such an implementation flaw exist, even a strong encryption algorithm, e.g., AES with 256-bit key and RSA, can be compromised. In fact, approximately 30% out of 8,307 public key certificates of SSL servers randomly chosen from the internet are vulnerable to prime number factorization due to the implementation failure in generating prime numbers [5]. Consequently, data eavesdropped during the routing process allow adversaries to be capable of traffic analyses [6] and violating personal privacy. These threats are of significant concern in modern society.

To address these issues, avoidance routing protocols [7]–[9] have been recently proposed. The key idea of this paradigm is to prevent adversaries from accessing or obtaining even the encrypted data. In avoidance routing, particular areas or nodes, e.g., routers in malicious nations and compromised routers, are deprioritized, such that a routing path never contains insecure areas. The existing solutions [7]–[9] are primarily designed for the Border Gateway Protocol (BGP) on the Internet or distance-vector networks, which avoid malicious and illegitimate nodes on which adversaries can eavesdrop. However, these approaches work under the assumption that there exists a safe path, i.e., a routing path with no node on which an adversary can eavesdrop, between the source and destination. Unfortunately, the opportunity for such a condition to be satisfied by routing is very low.

In this paper, we propose an avoidance routing framework for wireless ad hoc networks. The key idea of the proposed scheme is a combination of multi-path routing and the XOR coding. Consider that a source node, which wishes to send message m, computes m_1, m_2, ..., m_k where $m = m_1 \oplus m_2 \oplus ... \oplus m_k$. Here, $\oplus$ is the XOR operation. Based on multi-path routing, the source node selects a set of paths, say p_1, p_2, ..., and p_k, and sends messages m_1, m_2, ..., m_k, via each path. The destination node assembles pieces into the original message m. On the contrary, an adversary, which is assumed not to collude with others [10], cannot obtain m unless she eavesdrops on the traffic of all of the k paths.

At first glance, our approach incurs traffic overhead k times. However, we claim that our solution incurs more overhead only in the networks where the distance-vector avoidance protocols [8], [9] do not work. Instead, the delivery rate of our approach is significantly improved by taking advantage of the multi-path routing protocol. In the networks where the existing protocols securely deliver a message, the proposed protocol yields the same overhead as the existing approaches. Specifically, the contributions of this paper are as follows:

- We first derive the network condition that an ideal routing protocol with a perfect encryption scheme requires. Then, we analyze the exact network condition that the distance-vector-based approaches [8], [9] require in order to demonstrate the gap of the

performance upper bound between the ideal protocol and the existing solutions.

- We propose a framework for avoidance routing, namely Multi-Path Avoidance Routing (MPAR), that incorporates multi-path routing and the XOR coding. The proposed framework requires a much weaker condition to securely deliver a message compared with existing solutions.
- We develop a k-path route discovery protocol by using our MPAR framework, which discovers k paths with no common adversary.
- We provide security analyses of the MPAR framework. To be specific, we prove that the encoding scheme used in the framework achieves perfect secrecy, as long as an adversary has incomplete information regarding a message.
- We conduct extensive simulations to evaluate the proposed scheme. The results show that our protocol significantly improves performance in terms of message delivery rate.

The remainder of the paper is organized as follows: Section II reviews related works. In Section III, the avoidance routing is formulated and network conditions required by an ideal routing protocol and existing solutions are analyzed. We propose the MPAR framework for securing data communications in Section IV and develop a k-path route discovery protocol by using the framework in Section V. The security of the MPAR framework is analyzed in Section VI, and the performance is quantitatively evaluated by computer simulations in Section VII. Section VIII concludes this paper.

II. Related Work

A. Ad Hoc Routing Protocols

Dynamic source routing (DSR) [11] and ad hoc on-demand routing (AODV) [12] are the most well-known ad hoc routing protocols, which compute the shortest path from a source to a destination. While the shortest path routing generally brings high performance, it lacks important considerations in resource-limited ad hoc networks. To this end, non-shortest path-based routing protocols, such as load-balancing [13], energy-aware [14], reliability [15], and congestion avoidance, have been proposed in the past. These protocols will *avoid* bottleneck area/nodes based on the different design goals. On the contrary, our *avoidance* concept differs greatly; this is because the avoidance routing, which we refer to in this research, is a countermeasure to security attacks.

B. Multi-Path Routing Protocols

Multi-path routing protocols, such as [16]–[18], send data via a set of node/link disjoint paths [19] from a source to a destination. Finding multiple paths with fewer number of route discovery packets are studied in [20], [21]. There are two advantages to multi-path routing. One is that it improves throughput and/or message delivery rate by parallelizing message transmissions over multiple paths [16]; the other is fault tolerance [17], [18]. Even if one of the paths goes down due to congestion, failure of link/node, or out-of-battery of mobile hosts, the other paths, which can be considered as backups, are available for message delivery. While the avoidance routing that we consider in this research is multi-path-based, its purpose and design principles differ from those of existing protocols. That is, we employ the multi-path approach to protect data from security attacks.

C. Secure Multi-Path Routing Protocols

Secure multi-path routing protocols have been explored to protect data from eavesdroppers. The idea of securing data is based on (t, k)-threshold [22], in which a party can obtain the plaintext from ciphertext if she has t out of k secrets. SPREAD [23] applies the (t, k)-threshold scheme to secure multi-path routing by spreading k secrets into k different paths. However, the route discovery process in SPREAD relies on a modified Dijkstra algorithm [24], and thus is not appropriate due to the lack of its distributed nature. Another approach is to use secure network coding [25], [26]. However, these works are primarily designed for broadcast, and how to find safe paths is not discussed.

The theoretical aspects of secure communications in large scale wireless networks have been studied in terms of the per-node secure throughput. To be specific, the secrecy capacity in a static network with eavesdroppers of known and unknown locations are presented in [27] and [28], respectively. The work [29] further explores the secure throughput in the case with the multi-path mode and proposes a routing algorithm with corporative jamming. However, jamming requires an additional operational cost for transmission scheduling and incurs energy consumption; therefore, such an approach is out of our scope.

D. Avoidance Routing Protocols

The work most relevant to this study is that regarding area/nodes avoidance routing [7]–[9]. The work in [7] proposes an avoidance request algorithm in which a source node specifies the security properties, such as geographical locations, router types, and ISPs. This can be incorporated into BGP routers used on the Internet. The other works [8], [9] are designed for distance-vector networks. Virtual Positioning Source Routing (VPSR) [8] utilizes the beacon-vector-based routing paradigm, in which a subset of nodes in a network are used as reference points, and each node maintains the distance to reference nodes as its virtual coordinate. The research in [9] extends the beacon-based protocol so that not only reference nodes, but also non-reference nodes, can be the intermediate nodes on a routing path, and then the path selection algorithms, called the greedy and restricted area avoidance (Greedy-AA and Restricted-AA) protocols, are applied. The problems of this approach are that there must be a safe path between two nodes, and the network is assumed to be dense with high connectivity.

III. Problem Formulation

A. Notations and Definitions

A set of paths between two nodes, say n_s and n_d, is denoted by $P(n_s, n_d) = \{p_1, p_2, ..., p_k\}$. If n_s and n_d are not connected, then $P(n_s, n_d)$ is empty. Node n_i is n_j's neighbor if and only if n_i is within the transmission range of n_j. The open neighbor set of node n_i is denoted by $Nbr(n_i)$, which

TABLE I: Definition of notations.

Symbols	Definition
n_i	Node i
$P(n_s, n_d)$	A set of paths from n_s to n_d
$p_i \in P(n_s, n_d)$	A path i between n_s and n_d
$Nbr(n_i)$	The neighbor set of n_i
$Nbr(p)$	The union of neighbor sets of the nodes on path p, $\bigcup_{\forall n_i \in p} Nbr(n_i)$
$\hat{\mathcal{A}}_i$	Adversary i
$AS(p)$	A set of adversaries along path p
m	Message m
$Gen_u(.)$	A random generator with the uniform distribution
SG_c	A set of graphs that satisfies the condition c
$\oplus$	The XOR operator

contains all neighbors of n_i excluding n_i itself. In addition, we define a collection of neighbors of the nodes on path p by $Nbr(p) = \bigcup_{\forall n_i \in p} Nbr(n_i)$. An adversary is denoted as $\hat{\mathcal{A}}$, and a set of adversaries on a path, say p, is denoted as $AS(p)$, which contains the adversaries in the neighbor lists of all the nodes on path p. The notation used in this paper is listed in Table I.

B. Assumptions

In this paper, an undirected graph is used to represent an ad hoc network. For simplicity, each node has the same transmission range, and thus the link is bi-directional. The link cost is assumed to be uniform, since the primary objective of avoidance routing is how to discover safe paths, while excluding paths with high throughput. The global view of a network graph is assumed to be unavailable, but each node has its neighbor list within the transmission range by periodic local information exchanges (i.e., beacons) defined by the IEEE 802 standards [30].

We assume that each node knows about the existence of an adversary, if she is in its neighbor list. Finding adversaries in a local area is possible by anomaly detection [31]. On the contrary, a source node, which wishes to send a message, does not know where adversaries are located when it initiates a route discovery process. Therefore, a routing path that avoids an insecure area must be discovered in a distributed fashion.

In this paper, the collusion attack is considered, where a set of connected adversaries can collude to eavesdrop data. As far as we know, no research on how to determine to which group or community an adversary belongs has been conducted. Without the information about a group of adversaries colluding together, there is no deterministic way to find a set of paths without common adversaries. When each adversary is independent from the others, however, we can design a k-path route discovery protocol that guarantees that k paths have no common adversary; this is elaborated on in Section V.

C. Adversary Model

In our model, adversaries are assumed to have unbounded computational power. While an encryption scheme protects

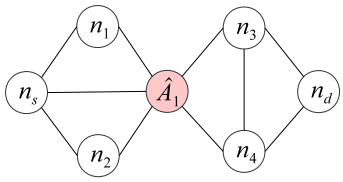

Fig. 1: A cut vertex.

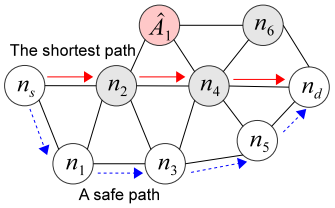

Fig. 2: A safe path.

data against adversaries with bounded computational resources, e.g., power, time, and memory storage, it is not necessarily always the case. For instance, a nation may spend a large amount of resources to break encrypted data, one example being British Intelligence during World War II. In addition, not only data privacy and integrity, but traffic analysis is also of concern. Thus, eavesdropping adversaries can become potential threats to end users and network administrators.

In this paper, we claim that avoiding insecure areas (or nodes) is the primary countermeasure against potential adversaries. The first step for adversaries to break security is to eavesdrop/block traffic. Specifically, the following attacks are considered.

Attack 1 (Eavesdropping) *If adversary $\hat{\mathcal{A}}$ is node n_i's neighbor, i.e., $\hat{\mathcal{A}} \in Nbr(n_i)$, then $\hat{\mathcal{A}}$ can eavesdrop on n_i's data transmission. Once the adversary obtains the transmitted data, she is assumed to be capable of breaking data privacy within a reasonable amount of time, unless the encryption scheme is of perfect secrecy.*

Attack 2 (Denying Service) *If a path from source n_s to destination n_d contains an adversary as an intermediate node, she can not only obtain the content of the message but can also deny forwarding the message.*

Attack 2 tells us that a routing path should never contain an adversary as an intermediate node. In addition, Attack 1 implies that a routing path should avoid insecure areas, or equivalently, nodes that have an adversary in their neighborhood.

D. Perfect and Polynomial Secrecy

An encryption scheme is of *perfect secrecy* if and only if no adversary with unbounded computational power can compromise encrypted data with a probability better than random guessing. By this definition, a perfect encryption scheme shall defend encrypted data from eavesdropping, i.e., Attack 1. However, even with an ideal encryption, a routing protocol may not accommodate Attack 2, since it is impossible to deliver a message to the destination if an adversary on a path drops the message.

On the contrary, an encryption scheme is of *polynomial secrecy* if and only if no adversary with a polynomial amount of computational power can break encrypted data with a probability non-negligibly greater than random guessing. In the worst case, an adversary which obtains encrypted data may spend a huge amount of computational and human resources to compromise it. Therefore, in this research, the polynomial encryption scheme is assumed to be insecure against both Attacks 1 and 2.

E. The Bounded Condition

Different routing schemes require different network conditions. Consider an ideal routing protocol with perfect encryption, i.e., an encryption scheme achieves perfect secrecy where an unbounded adversary cannot break encrypted data with a probability no better than random guessing. Although the encryption scheme is perfect, the ideal routing protocol may fail. For example, should an adversary be on the path, she simply denies forwarding the message, resulting in a delivery failure.

Intuitively, any routing protocol requires that there must be a path between the source and destination such that the path contains no adversary as an intermediate node. We derive the bounded condition that even an ideal routing protocol, and thus any routing protocol, requires the secure delivery of a message to the destination against Attacks 1 and 2, as follows:

Condition 1 (The bounded condition) *Given source n_s and destination n_d, an adversary (or a set of adversaries) consists of a cut vertex (or cut vertices), whose removal would disconnect n_s and n_d.*

We will prove that all routing protocols require Condition 1 by Theorem 1.

Theorem 1 *Any routing protocol requires Condition 1 to securely deliver a message from a source to a destination against Attacks 1 and 2.*

Proof: The proof is trivial, and thus is omitted. ∎

Figure 1 depicts a network of 7 nodes which include adversary $\hat{\mathcal{A}}_1$, and $\hat{\mathcal{A}}_1$ is also a cut vertex. The removal of $\hat{\mathcal{A}}_1$ divides the network into two disjoint components. One contains n_s, n_1, and n_2; the other contains n_d, n_3, and n_4. Since any routing path from n_s to n_d must traverse $\hat{\mathcal{A}}_1$, no routing protocol can securely deliver a message between n_s and n_d.

F. Avoiding Eavesdroppers

In reality, an ideal routing protocol with a perfect encryption scheme does not exist. Adversary $\hat{\mathcal{A}}$ can intercept data transmitted via p when $\hat{\mathcal{A}} \in Nbr(p)$. Even if data is encrypted, an adversary may spend a large amount of computational resources to compromise encrypted data once she intercepts data by eavesdropping. Hence, the primary countermeasure is to avoid insecure areas. A *safe* path against eavesdropping is defined as follows:

Definition 1 (A Safe Path Against Eavesdropping) *Given source n_s and destination n_d, a path, denoted by p, is said to be safe against eavesdropping, if $\hat{\mathcal{A}} \notin Nbr(p)$ holds.*

Figure 2 depicts a snapshot of a network, where a red circle represents an adversary and gray circles represent the neighbors of the adversary. As observed from the figure, the shortest path between n_s and n_d is $n_s \rightarrow n_2 \rightarrow n_4 \rightarrow n_d$. However, this path is not safe because $\hat{\mathcal{A}}_1$ can eavesdrop on the transmission of n_2 and n_4. On the contrary, the path $n_s \rightarrow n_1 \rightarrow n_3 \rightarrow n_5 \rightarrow n_d$ is safe, since $\hat{\mathcal{A}}_1$ is not in the proximity of any node on the path.

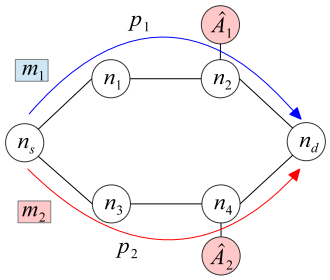

Fig. 3: The idea of MPAR.

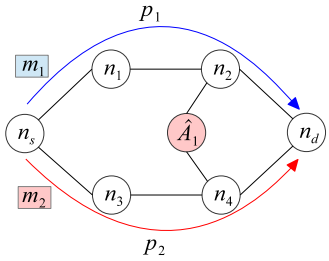

Fig. 4: A common adversary.

The existing avoidance protocols [8], [9] for distance-vector networks assume that there exists a safe path, and therefore, to securely deliver a message, these protocols require Condition 2.

Condition 2 *Given source n_s and destination n_d, there is at least one safe path (Definition 1) between n_s and n_d.*

It is clear that Condition 2 is much more strict than the bounded condition provided by Condition 1. Unfortunately, all of the single-path routing protocols with a polynomial encryption scheme, including [8], [9], require this strong condition, which is proven by Theorem 2.

Theorem 2 *Any single-path routing protocol with a polynomial encryption scheme requires Condition 2 to securely deliver a message from a source to a destination against Attacks 1 and 2.*

Proof: The proof is trivial, and thus is omitted.

The availability of safe paths depends on the network condition. Hence, a protocol that requires a weaker condition has more routing opportunities than does one that requires a strong condition. How to relax Condition 2 is critical in designing a practical avoidance routing protocol.

IV. Multi-Path Avoidance Routing

In this section, we propose a framework for avoidance routing, namely Multi-Path Avoidance Routing (MPAR).

A. The Basic Idea

The proposed MPAR framework incorporates the idea of multi-path routing and the XOR coding. Figure 3 shows a network of 8 nodes which includes 2 adversaries. There are two paths, i.e., p_1 and p_2, from n_s to n_d. Clearly, both of the paths are not safe, since $\hat{\mathcal{A}}_1 \in Nbr(p_1)$ and $\hat{\mathcal{A}}_2 \in Nbr(p_2)$.

The idea of the proposed protocol works as follows: to deliver a message, say m, source node n_s first generates a random bit string m_1 ($|m| = |m_1|$), and then computes m_2 such that $m = m_1 \oplus m_2$. Note that m_2 seems random to other nodes since m_1 is randomly generated. Thus, this coding scheme is as secure as the Vernam's one-time pad [32]. After the XOR coding, node n_s sends m_1 via p_1 and m_2 via p_2, respectively. During the delivery process, adversary $\hat{\mathcal{A}}_1$ intercepts m_1 by eavesdropping on n_2's transmission, and $\hat{\mathcal{A}}_2$ intercepts m_2 by eavesdropping on n_4's transmission. However, neither $\hat{\mathcal{A}}_1$ nor $\hat{\mathcal{A}}_2$ is able to successfully decode m with a probability better than random guessing without obtaining both m_1 and m_2. On the contrary, destination n_d

will receive both m_1 and m_2, and therefore is able to assemble m by computing $m_1 \oplus m_2$.

However, this two-path routing does not work in the network shown in Figure 4, where $\hat{\mathcal{A}}_1 \in Nbr(n_2) \cap Nbr(n_4)$. In this case, the adversary can intercept both m_1 and m_2, and assembles them to obtain the original message, m.

To tackle this problem, we can generalize the aforementioned idea into k-path routing. Assume there are k paths, p_1, p_2, ..., p_k, between a source and a destination. The source node computes $m = m_1 \oplus m_2 \oplus ... \oplus m_k$, and then sends m_i via path p_i ($1 \leq i \leq k$). The destination node obtains m by computing XOR from the received k messages. On the contrary, an adversary cannot obtain m unless she intercepts all of the k messages.

B. The MPAR Framework

The skeleton of the MPAR framework is presented in Algorithm 1. First, a route discovery process discovers multiple paths. Since MPAR is a framework, any routing discovery algorithm can be incorporated. As a demonstration, we will present an adversary disjoint j-path route discovery protocol in Section V.

Lines 6 to 16 in the skeleton describe how to select and send a message to its destination. If there is a safe path p in $P(n_s, n_d)$, that satisfies the safe path condition against eavesdropping (Definition 1), then message m is sent via p.

If no safe path can be found, the protocol enters the multi-path mode with the XOR coding. Given k paths (where $k \geq 2$), source node n_s randomly generates bit strings, m_1, m_2, ..., and m_{k-1}, by a random generator with the uniform distribution, denoted by $Gen_u(|m|)$. Here, the input, i.e., $|m|$, is the length of a bit string that the generator returns, and therefore, $|m_i| = |m|$ for $1 \leq i \leq k-1$. Then, n_s computes m_k by taking $m \oplus m_1 \oplus m_2 \oplus ... \oplus m_{k-1}$. Finally, each message m_i is sent to the destination via each path p_i, where $1 \leq i \leq k$.

When the destination node receives all the pieces, it assembles m from m_i ($1 \leq i \leq k$) by taking the XOR operation. Note that m_k seems to be a random string to a third party since the messages, m_1, m_2, ..., m_{k-1}, are randomly generated. Therefore, an adversary cannot recover m unless she obtains all of m_1, m_2, ..., and m_k.

C. The Condition of The MPAR Framework

In this subsection, we will show that the MPAR framework requires a much weaker condition than the existing avoidance protocols. First, we introduce the concept of adversary disjoint paths by Definition 2.

Definition 2 (Adversary Disjoint Paths) *Given source n_s and destination n_d, a set of paths $P(n_s, n_d) = \{ p_1, p_2, ..., p_k\}$ are said to be adversary disjoint paths if and only if there is no common adversary $\hat{\mathcal{A}}$ for all of the k paths, i.e., $\hat{\mathcal{A}} \notin \bigcap_{i=1}^{k} Nbr(p_i)$.*

When there is no safe path between n_s and n_d, the protocol enters the k-path routing mode as shown from lines 9 to 13 in Algorithm 1. The condition in which the k-path mode securely delivers a message from a source to a destination is given as follows:

Algorithm 1 MPAR(n_s, n_d, m)

```
1:  /* Node n_s executes the following: */
2:  /* The route discovery phase */
3:  P(n_s, n_d) ← RouteDiscovery(n_d).
4:
5:  /* The message forwarding phase */
6:  if there is a safe path p in P(n_s, n_d) then
7:      /* The single-path mode */
8:      n_s sends m via p.
9:  else if there are adversary disjoint paths in P(n_s, n_d) then
10:     /* The multi-path routing mode */
11:     k ← |P(n_s, n_d)|
12:     n_s randomly generates m_i for 1 ≤ i ≤ k − 1 by
        Gen_u(|m|).
13:     n_s computes m_k = m ⊕ m_1 ⊕ m_2 ⊕ ... ⊕ m_{k−1}.
14:     n_s sends m_i via p_i for each i (1 ≤ i ≤ k).
15: else
16:     /* There is neither a safe path nor adversary disjoint
        paths */
17:     n_s discards m.
```

Condition 3 *Given source n_s and destination n_d, there is at least one set of adversary disjoint paths (Definition 2) between n_s and n_d.*

The if-then statement at line 6 is exactly the same as that in Condition 2, and line 9 is the case of Condition 3. Now, we can derive the network condition that the proposed MPAR must hold for successful message delivery.

Theorem 3 *Given source n_s and destination n_d, the MPAR framework can securely deliver a message from n_s to n_d if and only if either Condition 2 or Condition 3 is met.*

Proof: The proof is trivial, and thus is omitted.

We claim that the condition required by our MPAR framework is much weaker than that required by any single-path-based approach with a polynomial encryption scheme (hence, we hereby simply say single-path protocol). To prove our claim, we provide the following theorem.

Theorem 4 *The MPAR framework requires a weaker condition than any single-path protocol with a polynomial encryption scheme.*

Proof: First, the MPAR framework requires either Condition 2 or 3, while a single-path-based approach requires Condition 2. Let SG_{c2} be the set of all the network graphs that satisfy Condition 2, and let SG_{c3} be the set of all the network graphs that satisfy Condition 3. The MPAR framework succeeds in $SG_{c2} \cup SG_{c3}$, while a single-path-based approach succeeds only in SG_{c2}.

The proof is by contradiction. Assume the MPAR framework requires a stronger condition than any single-path protocol. Then, $SG_{c2} \cup SG_{c3} \subset SG_{c2}$ must hold. However, the set of graphs $SG_{c3} \backslash SG_{c2}$ is not empty, since a counter example is shown in Figure 3. In Figure 3, there are two paths between n_s and n_d. Neither of the paths is safe, but they are adversary disjoint paths. Thus, $SG_{c2} \cup SG_{c3} \subset SG_{c2}$ never holds. This is a contradiction. Therefore, the above claim is true. This completes the proof. ∎

D. Performance of The MPAR Framework

At first glance, our MPAR framework increases traffic overhead by k times, since k messages are sent via k different paths. However, the MPAR framework does not sacrifice the performance in the networks where a single-path protocol works. This is because, if there exists a safe path, MPAR simply sends the message via the safe path as shown in lines 6 and 7 in Algorithm 1. If no safe path is found, the single-path protocol does not work, but our MPAR still has a chance to securely deliver a message by the multi-path mode. Therefore, the MPAR framework introduces additional overhead only when the single-path protocol does not work.

V. k-Path Route Discovery

In this section, we develop a k-path route discovery protocol by using the MPAR framework. To be specific, this is an implementation of $RouteDiscovery(.)$ in Algorithm 1.

A. Protocol Overview

The pseudo code of the k-path route discovery protocol is presented in Algorithm 2. Similar to the route request phase of DSR [11] and AODV [12], a source node, say n_s, floods the network with route requests. Then a destination, say n_d, replies with the list of adversaries. By reversing from n_d, intermediate nodes set up a routing entry. Note that intermediate nodes may have adversaries in their neighbors, but no adversary is used as a predecessor or a successor on the path during the route reply phase. After n_s receives the first reply from n_d, it again floods the network with the second request packets containing the set of adversaries obtained in the first request phase. This repeats until k adversary disjoint paths are discovered, or the amount of flooding exceeds k_{max}. By doing this, two adversary disjoint paths are discovered, if they exist.

B. Route Discovery Phase

A source node, say n_s, employs Algorithm 2 to discover routing paths. This function will return either single path p_1, a set of k paths p_1, p_2, ..., p_k, or an empty set. During the route discovery phase, a request is first broadcast over the network in search of a safe path. In the case that a safe path is not found, a second request will then flood the network looking for an adversary disjoint path. This process continues until k adversary disjoint paths are found. Otherwise, the route discovery terminates and returns an empty set, i.e., no k adversary disjoint paths can be found, where $k \leq k_{max}$.

In the first request phase, node n_s broadcasts a request packet, denoted by $RREQ_1$, which contains the source and destination IDs, i.e., n_s and n_d. Consider the case that an intermediate node, n_i, receives the request packet from node n_j. On receiving $RREQ_1$, n_i creates a routing entry in its table for this flow. A routing entry has a path ID, source ID, destination ID, predecessor ID, and descendant ID. Each entry is uniquely identified by a tuple (a path ID, a source ID, and a destination ID), where the path ID ranges from 1 to k_{max} (the ID assigned by $RREQ_k$). At this time, n_i sets the path ID to be 1, the source ID to be n_s, the destination ID to be n_d, and the predecessor ID to be n_j, respectively. The descendant ID, which is set during the reply phase, is kept null at this moment. If there exists the corresponding entry, n_i just discards $RREQ_1$. In addition, if n_i receives a request packet from an adversary, it simply drops the request. Note that we assume a node knows the existence of an adversary only when it is a neighbor of the adversary.

When the destination node, n_d, receives $RREQ_1$ the first time, it replies with a reply packet, denoted by $RREP_1$. The reply packet contains the source ID, the destination ID, and a set of adversary IDs. The reply packet is routed along the

Algorithm 2 kPathRouteDiscovery(n_s, n_d, k_{max})

1: /* Node n_s executes the following: */
2: n_s broadcasts $RREQ_1 := (n_s, n_d)$.
3: **On receiving** $RREP_1$ **via** p_1**.**
4: **if** $AS(p_1)$ is empty **then**
5: /* p_1 is a safe path. */
6: return p_1.
7: **else**
8: **for** $k = 2$ to k_{max} **do**
9: /* n_s tries to find an adversary disjoint path */
10: n_s broadcasts $RREQ_k := (n_s, n_d, \sum_{r=1}^{k-1} AS(p_r))$.
11: **On receiving** $RREP_k$ **via** p_k**.**
12: **if** $\bigcap_{r=1}^{k-1} AS(p_r)$ is empty **then**
13: /* p_1, p_2, ..., p_k are adversary disjoint paths. */
14: return (p_1, p_2, ..., p_k).
15: /* No path is found */
16: return an empty set.
17:
18: /* Intermediate node n_i executes the following: */
19: /* Note that a routing entry is defined as (path ID, source ID, destination ID, predecessor ID, descendant ID) */
20: **On receiving** $RREQ_1$ **from** n_j **for** p_1**.**
21: **if** $n_j \neq \hat{\mathcal{A}}$ and n_i has no routing entry for (1, n_i, n_d) **then**
22: n_i creates an entry (1, n_i, n_d, n_j, null).
23: n_i broadcasts $RREQ_1$.
24: **else**
25: n_i drops $RREQ_1$.
26: **On receiving** $RREP_1$ **from** n_j **for** p_1**.**
27: n_i sets the descendant ID to be n_j in the corresponding entry.
28: n_i adds $\forall \hat{\mathcal{A}} \in Nbr(n_i)$ to $RREP_1.AS(p_1)$.
29: n_i sends $RREP_1$ to the predecessor.
30: **On receiving** $RREQ_k$ **from** n_j **for** p_k**.**
31: **if** $n_j \neq \hat{\mathcal{A}}$, n_i has no routing entry for (k, n_i, n_d), and $Nbr(n_i) \cap \sum_{r=1}^{k-1} RREQ_k.AS(p_r)$ is empty. **then**
32: n_i creates an entry (k, n_i, n_d, n_j, null).
33: n_i broadcasts $RREQ_k$.
34: **else**
35: n_i drops $RREQ_k$.
36: **On receiving** $RREP_k$ **from** n_j **for** p_k**.**
37: n_i sets the descendant ID to be n_j in the corresponding entry.
38: n_i sends $RREP_k$ to the predecessor.
39:
40: /* Node n_d executes the following: */
41: **On receiving** $RREQ_k$ **from** n_j**.**
42: **if** this is the first time to receive $RREQ_k$ and there is no adversary in $Nbr(n_d)$ **then**
43: n_d sends $RREP_k$ to n_j.

predecessor ID at each intermediate node. Consider the case that an intermediate node n_i receives the reply packet from n_j. On receiving $RREP_1$, n_i stores n_j as the descendant ID. If n_i has adversaries in its neighbors, the adversary IDs are added to the set of adversary IDs, which is denoted by $AS(p_1)$ in $RREP_1$. Then, n_i sends $RREP_1$ to the predecessor node.

If there exists a path that satisfies Condition 1, n_s will receive $RREP_1$. Let p_1 be the path found in the first request phase. Now, n_s has a list of adversaries on p_1. If $AS(p_1)$ is empty, p_1 is a safe path. Thus, by Condition 2, $kPathRouteDiscovery(n_s, n_d)$ returns a single path p_1, and n_s simply sends m via p_1. When n_s does not receive a reply from n_d, any path from n_s to n_d contains at least one adversary, i.e., Condition 1 does not hold. In this case, even an ideal routing protocol with a perfect encryption scheme cannot route a packet to the destination, and therefore, $kPathRouteDiscovery(n_s, n_d)$ returns an empty set.

Otherwise, p_1 contains at least one adversary, and hence n_s will try to find another path. The k-th request phase can be generalized as follows. A k-th request packet, denoted by $RREQ_k$, includes the source ID, the destination ID, and a list of adversaries, i.e., (n_s, n_d, $\bigcap_{r=1}^{k-1} SA(p_r)$). Similar to the first request packet, n_s broadcasts $RREQ_k$. When an intermediate node, say n_i, receives $RREQ_k$ from n_j, n_i first computes $Nbr(n_i) \cap Nbr(p_i)$. If the intersection is not empty, n_i has at least one common adversary with some neighbors on p_1, p_2, ..., or p_{k-1}, and therefore, n_i drops $RREQ_k$. Otherwise, n_i can be an intermediate node of p_k, and it creates a new routing entry with the same format as the first request packet. If there exists an adversary disjoint path from p_1, p_2, ..., and p_{k-1}, $RREQ_k$ will reach n_d. Otherwise, the k-th request packets will be discarded in the middle of this process. The routing table is created for the k-th path during the second reply process in the same fashion as the first reply process.

As described from Line 8 to 14 in Algorithm 2, source node n_s will receive the route reply $RREP_k$ from n_d. If the set of paths obtained by the discovery process have no common adversary, these k paths are adversary disjoint. Thus, $kPathRouteDiscovery(n_s, n_d)$ will return a set of paths, p_1, p_2, ..., and p_k. Otherwise, there are no adversary disjoint paths, and an empty set will be returned since the route discovery fails.

Remark Note that the proposed k-path route discovery protocol does not guarantee to obtain the minimum number of adversary disjoint paths. For instance, assume the k-path route discovery finds that p_1 has adversary $\hat{\mathcal{A}}_1$, and p_2 has adversaries $\hat{\mathcal{A}}_1$ and $\hat{\mathcal{A}}_2$, it will continue its path discovery. Again assume it finds that the third path p_3 has adversary $\hat{\mathcal{A}}_2$. As a result the set of paths $\{p_1, p_2, p_3\}$ will be considered adversary disjoint paths, which will all be used to transmit the message. However, it is clear that the subset of paths $\{p_1, p_3\}$ already achieve the adversary disjoint property. However, such cases rarely occur, since MPAR is most likely to deliver packets via two paths or fail to find a set of safe paths, which will be shown by the simulations in Section VII.

C. Message Forwarding Phase

Based on the result of the route discovery, i.e., $p = \{p_1, p_2, ..., p_k\}$), or an empty set, source node n_s sends m by either the single-path mode or the k-path mode, or refrains from message transmission. When a safe path is found, n_s simply sends m via p as described from lines 6 to 8 in Algorithm 1.

When p_1, p_2, ..., and p_k, which are adversary disjoint, are returned, n_s enters to the multi-path mode presented from lines 9 to 13 in Algorithm 1. A set of bit strings, say m_1, m_2, ..., and m_{k-1}, are randomly generated by $Gen_u(|m|)$, and then m_k is computed by taking $m \oplus m_1 \oplus m_2 \oplus ... \oplus m_{k-1}$. Note that m_k seems to be random, because m_1, m_2, ..., and m_k are random strings. Then, n_s sends m_k via p_k. Since p_1, p_2, ..., and p_k, are adversary disjoint paths, no adversary can obtain all the m_1, m_2, ..., and m_k, and hence cannot recover m even if she has the access to one of them. On the contrary, n_d can assemble m by taking $m_1 \oplus m_2 \oplus ... \oplus m_k$.

If there is no available path, the route discovery returns an empty set, and n_s discards m as shown from lines 14 to 16 in Algorithm 1.

VI. Security Analyses

A. Security of The MPAR Framework

It is known that the Vernam's one-time pad [32] achieves the perfect secrecy under two conditions: 1) the key length is the same as the message size; 2) the key is used only once. Our encoding scheme behaves just like the one-time pad, and thus achieves perfect secrecy as long as an adversary does not obtain all of the messages.

Recall that for a given message, m, a source node generates a random bit string, m_j ($1 \leq j \leq k-1$), by a generator, $Gen_u(.)$, where $|m_1| = |m|$. Then, $m_k = m \oplus m_1 \oplus m_2$, ..., $\oplus m_{k-1}$ is computed. Let M, K, and C be the domain of a message, a key, and a cipher, respectively. If an adversary does not have m_i for some $1 \leq i \leq k$, m_i works as a key and $m \oplus m_1 \oplus m_2 \oplus ... m_{i-1} \oplus m_{i+1} \oplus ... \oplus m_k$ works as a cipher. Since $m_j = \{0,1\}^{|m|}$ for any $1 \leq j \leq k$ and $|m| = |m_j|$, $|M| = |K| = |C|$ holds. The first condition of the perfect secrecy of the Vernam's one-time pad is met. The second condition is also the case, as different m_i for ($1 \leq i \leq k-1$) is generated and is used for different messages m.

To prove the security of the proposed scheme, we rely on Shannon's theorem [33] as follows:

Theorem 5 (Shannon's Theorem) *An encryption scheme over the message space M for which $|M| = |K| = |C|$ is perfectly secret, if and only if:*

- *Every key $k \in K$ is chosen with equal probability $1/|K|$ by a random generator.*
- *For every $m \in M$ and every $c \in C$, there exists a unique key $k \in K$ such that the encryption scheme outputs c.* ∎

By Shannon's theorem, we formally prove the perfect secrecy of our encryption scheme against eavesdropping (Attack 1) as long as no adversary has all the m_1, m_2, ..., m_k. Note that the adversary cannot obtain all the messages if k paths are adversary disjoint paths. In Theorem 6, we assume that an adversary does not have m_i ($1 \leq i \leq k$), and thus m_i works as a key.

Theorem 6 *Let $m := m_1 \oplus m_2 \oplus ... \oplus m_k$ be messages to be sent out. Given M, K, and C, the encryption scheme of the MPAR framework with the k-path mode achieves the perfect secrecy against eavesdropping as long as no adversary has m_i for $1 \leq i \leq k$.*

Proof: We prove the above claim by Shannon's Theorem. Denoting $m^i = m \oplus m_1 \oplus m_2 \oplus ...m_{i-1} \oplus m_{i+1} \oplus ... \oplus m_k$, m_i works as a key and m^i works as a cipher. The length of m^i is the same as m and m_i for any $1 \leq i \leq k$, and hence, $|M| = |K| = |C|$ holds. Since any of m_1, m_2, ..., and m_{k-1} are randomly generated with the uniform distribution, $Pr[key = m_i] = 1/|K|$ holds. For every $m \in M$ and $m^i \in C$, there exists a unique m_i such that $m_i = m \oplus m^i$. Therefore, by Shannon's Theorem, the above claim must be true. This completes the proof. ∎

VII. Performance Evaluation

In this section, computer simulations are conducted to evaluate the proposed scheme. Along with our MPAR with the k-path route discovery protocol, the ideal protocol and Greedy-AA [9] are implemented as follows:

- The ideal protocol with a perfect encryption scheme is assumed to deliver messages successfully when there is a path that contains no adversary as an intermediate node, which is provided by Condition 1. Although such a protocol does not exist, it represents the upper bound of the performance that any avoidance routing can possibly achieve.
- Greedy-AA [9] computes the shortest safe path that satisfies Condition 2. The cost of Greedy-AA is large, but the lowest number of hops among safe paths is guaranteed. However, for successful message delivery, there must exist a safe path between a source and a destination such that no node on the path has adversaries as its neighbors.

A. Simulation Configurations

A network is generated by randomly placing nodes in an 800 x 800 unit square region. The number of nodes ranges from 100 to 400, and the communication range is set to be 100 units. Thus, the average number of neighbors of each node ranges from 4.9 to 19.6. One to ten percent of nodes are set to be adversaries. For each network realization, a source and a destination are randomly selected. For each setting, 1,000 simulations are conducted.

Both the independent adversary and collusion attacks scenarios are considered. In the first scenario, adversaries never collude. In the second scenario, a set of connected adversaries are assumed to collude together by assembling eavesdropped data. We assume that each node can detect adversaries in its neighbors by anomaly detection. However, it is not distinguishable if a set of adversaries collude, since connected adversaries are distanced by more than one hop.

B. Performance Metric

As performance metrics, the message delivery rate, hop stretch, and the amount of traffic are used as follows:

- The message delivery rate is defined as the probability that a safe path can be found by each avoidance protocol. The ideal protocol succeeds when Condition 1 is met, and this provides the upper bound of the delivery rate. Greedy-AA succeeds only when Condition 2 is met. The proposed MPAR with the two-path route discovery protocol will succeed when either Condition 2 or Condition 3 holds. That is, there exists a safe path or a pair of adversary disjoint paths.
- The hop stretch is defined as the ratio between the actual number of hops and the lowest number of hops that a protocol incurs. Note that the lowest number of hops is defined as the minimum number of hops between a source and a destination excluding adversaries from the network, which can be achieved by the ideal protocol. For our MPAR protocol, there may be k paths, say p_1, p_2, ..., and p_k, and hence the number of hops is defined as the larger one of the two, i.e., $max\{|p_1|, |p_2|, ..., |p_k|\}$.
- The amount of traffic is defined as the number of message transmissions. This metric is somewhat related to the number of hops, because the number of message transmissions increases as the number of hops increases. For the proposed MPAR, the sum of the number of message transmissions via all the k paths is considered as the amount of traffic.

C. Simulation Results

Figure 5 shows the delivery rate with respect to the number of neighbors. The percentage of adversaries is set to be 5%. The delivery rate increases as the number of neighbors increases. This is because having more neighbors means that there are more paths, and as a result, more routing opportunities exist. As can be seen in the figure, our MPAR significantly improves the performance compared with Greedy-AA, and the delivery rate is very close to the upper bound when the number of neighbors is larger than 15.

Figure 6 demonstrates the delivery rate with respect to the percentage of adversaries. The number of nodes is set to be 300. It is natural that the delivery rate decreases in proportion to the percentage of adversaries. With our scheme, the delivery rate gradually decreases compared with Greedy-AA, and thus we can say that the proposed MPAR significantly improves the routing opportunity.

Figure 7 illustrates the hop stretch with respect to the number of neighbors. The percentage of adversaries is again set to be 5%. The hop stretch of the MPAR framework is slightly longer than that of Greedy-AA, but the difference is not significant. Overall, the hop stretch is upper bounded by approximately 1.27. Considering the improvement of the delivery rate shown in Figure 5, this overhead is acceptable.

Figure 8 presents the hop stretch with respect to the percentage of adversaries. The number of nodes are again set to be 300. The hop stretch of MPAR increases as the percentage of adversaries increases, while that of Greedy-AA decreases when the percentage of adversaries is larger than 4%. This is because the number of hops can be computed only when a message is delivered. Message deliveries over Greedy-AA are most likely to fail if the distance between source and destination nodes is long. As a result, Greedy-AA results in a smaller hop stretch as only short paths are counted. However,

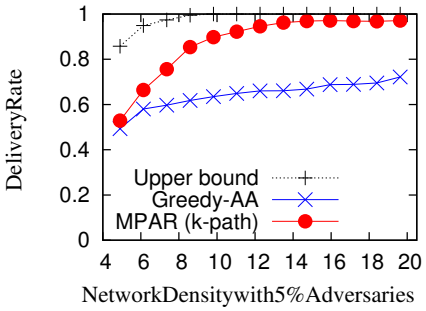

Fig. 5: The delivery rate.

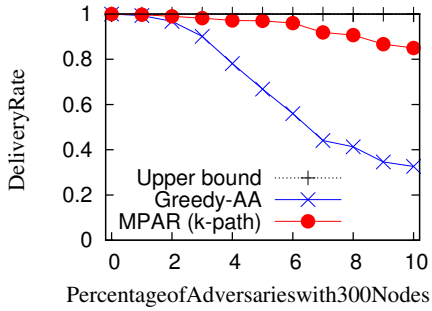

Fig. 6: The delivery rate.

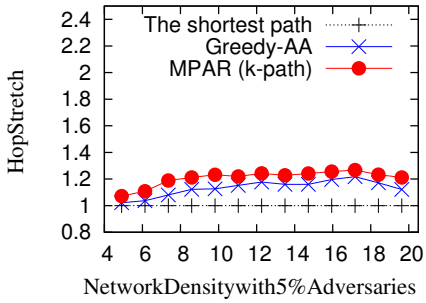

Fig. 7: The hop stretch.

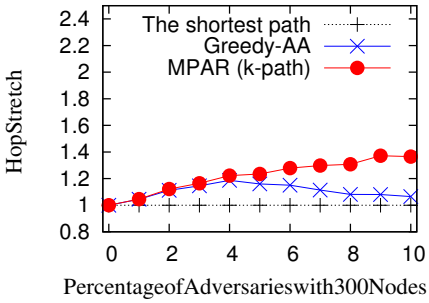

Fig. 8: The hop stretch.

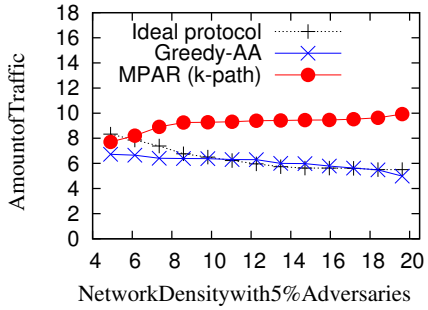

Fig. 9: The amount of traffic.

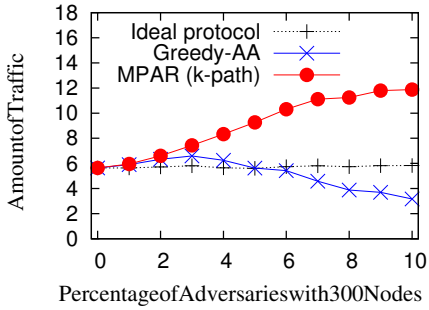

Fig. 10: The amount of traffic.

from Figures 6 and 8, it can be seen that our MPAR can still securely deliver a message with a greater number of hops.

Figure 9 depicts the amount of traffic with respect to the number of neighbors. As can be seen in the figure, MPAR incurs more traffic than the other protocols. However, we would like to emphasize that the larger amount of traffic does not necessarily mean that MPAR is poor. Putting Figures 5 and 6 together, it can be seen that Greedy-AA has a lower delivery rate, which contributes to a smaller amount of traffic since the message delivery is frequently terminated in the middle of a routing process. In addition, as we claim in Section IV-D, the MPAR framework does not introduce extra traffic in a network where Greedy-AA securely delivers a message. When a safe path does not exist between a source and a destination, MPAR introduces additional traffic overhead to discover adversary disjoint paths for the k-path routing mode in securely delivering a message.

Figure 10 plots the amount of traffic with respect to the percentage of adversaries. Apparently, the amount of traffic resulting from our MPAR increases as the percentage of adversaries increases. This is because having more adversaries in the network means that we have to rely more on the multi-path mode. As a result, the amount of traffic increases due to the redundant messages. On the other hand, the 70% routing process by Greedy-AA terminates message forwarding in networks with 10% adversaries, as Figure 6 indicates. This is why Greedy-AA introduces a smaller amount of traffic than does the ideal protocol.

D. Impact of Collusion Attacks

Figure 11 shows the delivery rate under collusion attacks with respect to the number of neighbors. In the case that adversaries collude together, the delivery rate is lower than the case that no adversaries collude. In addition, it is interesting that the delivery rate decreases when the number of neighbors is greater or equal to 14. This is because more neighbors indicate that more adversaries are connected to each other to collude. From Figures 5 and 11, MPAR does not show any significant performance degradation due to collusion attacks.

Figure 12 illustrates the delivery rate under collusion attacks with respect to the percentage of adversaries. Compared with Figure 6, the delivery rate of MPAR decreases by up to 13%. However, the proposed solution still maintains a higher delivery rate than that of Greedy-AA. Thus, we claim that MPAR can accommodate a certain degree of collusion attacks.

E. The Statistics

Figure 13 presents the cumulative distribution function (CDF) of the number of adversary disjoint paths for different numbers of nodes in a network. In the figure, the CDF does not reach 1, because the MPAR protocol cannot produce adversary disjoint paths in the networks where neither Condition 2 nor Condition 3 hold. In the case of the low network density (100 nodes), MPAR fails to find adversary disjoint paths and consequently results in low performance. However, when the number of nodes is greater than or equal to 200, the CDF is close to 1 by using more than one path. In addition, the figure indicates that two adversary disjoint paths are sufficient for securely delivering data most of the time.

VIII. Conclusion

In this paper, we investigate the avoidance routing problem. First, we formulate the network condition required by an ideal avoidance routing protocol with a perfect encryption scheme and any single-path routing protocol. Since the existing solutions simply avoid insecure areas, the routing opportunity thus is very limited. To tackle this issue, we propose a framework of Multi-Path Area Avoidance (MPAR), in which a message, m, is encoded into k pieces by $m = m_1 \oplus m_2 \oplus ... \oplus m_k$ and each m_i is sent via a different path. By doing this, an

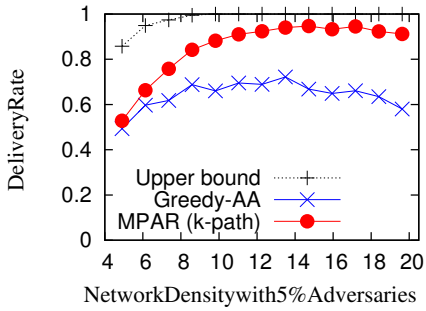

Fig. 11: The delivery rate under collusion attacks.

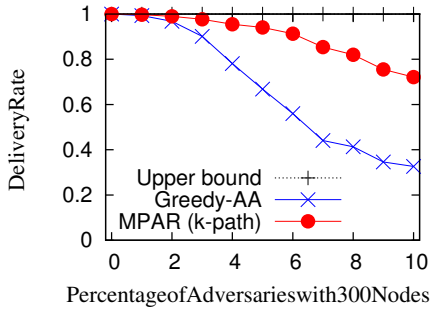

Fig. 12: The delivery rate under collusion attacks.

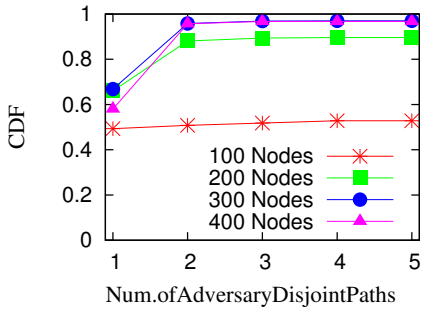

Fig. 13: The CDF of the number of adversary disjoint paths.

adversary cannot have access to m unless she obtains all pieces of the message m_i $(1 \leq i \leq k)$, while a legitimate receiver can assemble m by the XOR operation. Based on this framework, we develop the k-path route discovery protocol which discovers k adversary disjoint paths. In addition, the encoding scheme in the proposed solution achieves perfect secrecy under the condition that an adversary does not have m_i for some i $(1 \leq i \leq k)$, and that adversaries do not collude with each other. The simulation results show that our approach significantly improves the message delivery rate over the existing solutions even in the collusion attacks scenario. We believe that our MPAR framework serves as the foundation of critical communication environments, in which adversaries may spend a huge amount of computational and human resources to break encrypted data.

REFERENCES

[1] G. E. Moore, "Readings in Computer Architecture," 2000, ch. Cramming More Components Onto Integrated Circuits, pp. 56–59.

[2] D. Boneh, C. Dunworth, and R. J. Lipton, "Breaking DES Using a Molecular Computer," in *DIMACS Workshop on DNA Computing*, 1995.

[3] S. Singh, *The Code Book: The Evolution of Secrecy from Mary, Queen of Scots, to Quantum Cryptography*. Doubleday, 1999.

[4] P. Gutman, "Lessons Learned in Implementing and Deploying Crypto Software," in *Usenix Security Symposium*, 2002.

[5] R. Nojima, T. Kurokawa, and S. Moriai, "XPIA, X.509 Certificate Public-key Investigation and Analysis System," *NICT News, December 2013*, no. 435, pp. 5–6, 2013.

[6] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson, "Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries," in *CCS*, 2013, pp. 337–348.

[7] E. Kline and P. Reiher, "Securing Data Through Avoidance Routing," in *New Security Paradigms Workshop*, 2009, pp. 115–124.

[8] H. Zlatokrilov and H. Levy, "Navigation in Distance Vector Spaces and Its Use for Node Avoidance Routing," in *Infocom*, 2007, pp. 1253–1261.

[9] ——, "Area Avoidance Routing in Distance-Vector Networks," in *Infocom*, 2008, pp. 475–483.

[10] P. C. Pinto, J. ao Barros, and M. Z. Win, "Wireless Physical-Layer Security: The Case of Colluding Eavesdroppers," in *ISIT*, 2009, pp. 2442–2446.

[11] D. B. Johnson, "Routing in Ad Hoc Networks of Mobile Hosts," in *WMCSA*, 1994.

[12] C. E. Perkins and E. M. Royer, "Ad-hoc On-Demand Distance Vector (AODV) Routing," in *IETF RCF 3561*, 2003.

[13] R. Cohen and G. Nakibly, "On the computational complexity and effectiveness of n-hub shortest-path routing," *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, pp. 691–704, 2008.

[14] M. Agarwal, L. Gao, J. H. Cho, and J. Wu, "Energy Efficient Broadcast in Wireless Ad Hoc Networks with Hitch-Hiking," *MONET*, vol. 10, no. 6, pp. 897–910, 2005.

[15] J. Wu, F. Gao, Z. Li, and Y. Min, "Optimal and Reliable Communication in Hypercubes Using Extended Safety Vectors," *IEEE Trans. on Rel.*, vol. 54, no. 3, pp. 402–411, 2005.

[16] A. Tsirigos and Z. J. Haas, "Analysis of Multipath Routing-Part I: The Effect on The Packet Delivery Ratio." *IEEE Trans. on Wireless Commun.*, vol. 3, no. 1, pp. 138–146, 2004.

[17] P. P. C. Lee, V. Misra, and D. Rubenstein, "Distributed Algorithms for Secure Multipath Routing in Attack-resistant Networks," *IEEE/ACM Trans. Netw.*, vol. 15, no. 6, pp. 1490–1501, 2007.

[18] X. Zhang, X. Dong, J. Wu, X. Li, and N. Xiong, "Fault-Aware Flow Control and Multi-Path Routing in Wireless Sensor Networks," in *ICDCS Workshops*, 2013, pp. 27–32.

[19] Y. Ge, G. Wang, and J. Wu, "Node-Disjoint Multipath Routing with Group Mobility in MANETs," in *NAS*, 2010, pp. 181–189.

[20] S.-J. Lee and M. Gerla, "Split Multipath Routing with Maximally Disjoint Paths in Ad Hoc Networks," in *ICC*, 2001, pp. 3201–3205.

[21] S. K. Das, A. Mukherjee, S. Bandyopadhyay, D. Saha, and K. Paul, "An Adaptive Framework for QoS Routing Through Multiple Paths in Ad Hoc Wireless Networks," *J. Parallel Distrib. Comput.*, vol. 63, no. 2, pp. 141–153, 2003.

[22] A. Shamir, "How to Share a Secret," *Commun. of ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[23] W. Lou, W. Liu, Y. Zhang, and Y. Fang, "SPREAD: Improving Network Security by Multipath Routing in Mobile Ad Hoc Networks," *Wireless Networks*, vol. 15, no. 3, pp. 279–294, 2009.

[24] R. Bhandari, *Survivable Networks: Algorithms for Diverse Routing*. Kluwer Academic Publishers, 1998.

[25] N. Cai and R. W. Yeung, "Secure Network Coding on a Wiretap Network," *IEEE Trans. Inf. Theory*, vol. 57, no. 1, pp. 424–435.

[26] K. Jain, "Security Based on Network Topology Against The Wiretapping Attack," *IEEE Trans. on Wireless Commun.*, vol. 11, no. 1, pp. 68–71, 2004.

[27] O. O. Koyluoglu, C. E. Koksal, and H. E. Gamal, "On Secrecy Capacity Scaling in Wireless Networks," *IEEE Trans. Inf. Theory*, vol. 58, no. 5, pp. 3000–3015, 2012.

[28] S. Vasudevan, D. Goeckel, and D. F. Towsley, "Security-Capacity Trade-off in Large Wireless Networks Using Keyless Secrecy," in *Mobihoc*, 2010, pp. 21–30.

[29] C. Capar, D. Goeckel, B. Liu, and D. Towsley, "Secret Communication in Large Wireless Networks Without Eavesdropper Location Information," in *Infocom*, 2012, pp. 1152–1160.

[30] "IEEE 802.11b Standard," http://standards.ieee.org/getieee802/download/802.11b-1999.pdf.

[31] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," *ACM Computing Survey.*, vol. 41, no. 3, pp. 15:1–15:58, Jul. 2009.

[32] G. S. Vernam, "Cipher Printing Telegraph Systems for Secret Wire and Radio Telegraphic Communications," *Journal of American Institute for Electrical Engineers*, vol. 45, pp. 109–115, 1926.

[33] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. Chapman and Hall/CRC, 2008.