# Secure Data Communications in Wireless Networks Using Multi-Path Avoidance Routing

Kazuya SakaiG>, *Membe r, IEEE* , Min-Te SunG , *Member, IEEE* , Wei-Shinn **KuG,** *Senior Member, IEEE* , Jie WuG>, *Fellow, IEEE* , and Ten H. Lai

***Abstract*-Due to software implementation failure and misuse of cryptography, data encryption can no longer be considered a safeguard from security attacks. As a result, adversaries with eavesdropping capability along a routing path can compromise data privacy. In addition, should an adversary be one of the intermediate relay nodes in a path, she can deny data forwarding to disconnect the end-to-end communications. One solution is to avoid message routing through certain insecure areas, such as malicious countries or likely-compromised nodes. To this end, an avoidance routing based on the single path has been proposed. However, this single-path-based protocd relies on the availability of a safe path, i.e., no adversary is in the proximity of the whole path, which is difficult to achieve and therefore limits the routing opportunity.To tackle this issue, we propose an avoidance routing framework, namely timer-based multi-path avoidance routing (TMPAR). In our approach, a source node first encodesa message into *k* different pieces, and each piece is sent via a different path. During its path discovery phase, a timer is used to efficiently discover a better set of paths. The destinationcan assemble the original message easily. Under the condition that no adversary obtains all the *k* pieces of the message, the proposed TMPAR can securely deliver a message to its destination in spite of eavesdropping. The extensive ns-2 simulation results demonstrate that our TMPAR achieves its design goals.**

*Index Terms*- **Network security, routing protocols, ad hoc networks, wireless networks.**

## I. INTRODUCTION

TO PROTECT data privacy in computer networks, cryptography is a common strategy. For example, regardless how strong a cryptographic protocol is, if the software implementation is not done carefully, it may contain loopholes [1], allowing adversaries to compromise the encrypted data. As an example, approximately 30% of 8,307 public key certificates of SSL servers randomly chosen from the Internet are vulnerable to the prime number factorization attack due to implementation failure in generating prime numbers [2]. Additionally, traffic analyses [3] empower adversaries to be able to exploit personal privacy using only the data eavesdropped during the routing process. Finally, as more powerful computer hardware becomes more accessible and inexpensive [4], the encrypted data can be recovered by adversaries within a reasonable time period without knowledge of the encryption key. The most famous example is the successful cryptanalysis of the Enigma machines by British Intelligence, which became a determining factor in the Axis's defeat in World War II [5]. These threats are of significant concern in a modem society and a clear indication that data encryption is no longer a perfect solution for network security. For example, suppose an opponent country invades the US homeland and randomly drops wireless sensors with eavesdropping capability from airplanes. Consequently, many of the communication sessions in the region of interest can be compromised.

Manuscript received January 14, 2019; revised June 26, 2019; accepted June 29, 2019. Date of publication July 22, 2019; date of current version October 9, 2019. This work was supported in part by the JSPS KAKENJD under Grant JP17K12675, in part. by the Taiwan MOST under Grant 108-2218-E-001-002 and Grant 107-2221-E-008-082-MY2, and in part by the U.S. National Science Foundation under Grant ACI-1642133(CICD. The associate editor coordinating the review of this paper and approving it for publication was M. Li. *(Corresponding author: Min-Te Sun)*

K. Sakai is with the Department of Electrical Engineeringand Computer Science, Tokyo Metropolitan University, Tokyo 191-0065, Japan (e-mail: ksakai@tmu.ac.jp).

M.-T. Sun is with the Department of Computer Science and Information Engineering, National Central University, Taoyuan 320, Taiwan (e-mail: msun@csie.ncu.edu.tw).

W.-S. Ku is with the Department of Computer Science and Software Engineering, Auburn University, Auburn, AL 36849 USA (e-mail: weishinn@auburnedu).

J. Wu is with the Department of Computer and Information Science, Temple University, Philadelphia, PA 19122 USA (e-mail:jiewu@templ.edu).

T. H. Lai is with the Department of Computer Science and Engineering, The Ohio State University, Columbus, OH 43210 USA (e-mail: lai@cse.ohio-state.edu).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier JO.1109/fWC.2019.2928801

To prevent adversaries from accessing or obtaining even the encrypted data, avoidance routing protocols [6] - [8] have been proposed, in which particular areas or nodes, e.g., routers in malicious nations and compromised routers, are avoided, such that a routing path does not contain insecure areas. The existing solutions [6]- [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 routing path between the source and the destination with no node on which an adversary can eavesdrop. Unfortunately, the opportunity for such a condition to be satisfied is very small especially when the wireless ad hoc network is considered where the number of possible paths between source and destination is often very limited due to fluctuated nodal density [10], power-saving topology control [11], [12], or other reasons.

In this paper, we propose an avoidance routing framework for wireless ad hoc networks by extending our preliminary work [13]. 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 $m_1$, $m_2, \dots, m_k$ where $m = m_1 \oplus m_2 \oplus \dots \oplus m_k$. Here, $\oplus$ is the XOR operation. Based on multi-path routing, the source node selects a set of paths, say $p_1, p_2, \dots$, and $p_k$, and sends each of the messages $m_1, m_2, \dots, m_k$, via each path. The destination

TABLE I

D EFINITION OF NOTATIONS

| Symbols | Definition |
|---|---|
| $n,$ | Node i |
| $P(n,,na)$ | A set of paths from $ns$ to $na$ |
| $p,$ E $P(n , ,na)$ | A path i between $n.$ and $na$ |
| $N(n ; )$ | The neighbor set of $n;$ |
| $N(p)$ | The union of neighbor sets of the nodes on path $p$, i.e., Uvn,EP$N$ $(n ; )$ |
| $A;$ | Adversary i |
| $AS$ | A set of adversary's IDs |
| $AS(p)$ | A set of adversaries along path $p$ |
| m | Message m |
| $Gen,.(.)$ | A random generator with the uniform distribution |
| $SGc$ | A set of graphs that satisfies the condition c |
| EB | The XOR operator |

node assembles pieces into the original message $m$. On the contrary, an adversary, which is assumed not to collude with others [14], 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 [7], [8] do not work. Instead, the delivery rate of our approach is significan tly 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 required by an ideal routing protocol with a perfect encryption scheme. Then, we analyze the exact network condition that the distance-vector-basedapproaches [7], [8] 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.
- Based on the preliminary work [13], we develop the timer-based k-path discovery protocol that identifies a better set of safe paths and reduces the message overhead compared with the original MPAR. The resulting protocol is named Timer-based **MPAR (TMPAR).**
- We conduct extensive simulations using ns-2 [15] to evaluate the proposed scheme under the collision environment. The results show that our protocol significantly improves performance in terms of message delivery rate.

The remainder of the paper is organized as follows: In Section II, 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 ill . The timer-based **MPAR** is developed using the framework in Section IV. Performance is quantitatively evaluated by ns-2 in Section V. Section **VI** reviews related works. Section **VII** concludes this paper.

## II. PROBLEM FORMULATION

### A. Notations and Definitions

A set of paths between two nodes, say $n_8$ and $nd$, is denoted by $P(n\ s,\ nd) = \{P1, P2, . . . , pk\}$. If $ns$ and $nd$ are not connected, then $P(n\ s,\ nd)$ is empty. Node $ni$ is n/s neighbor if and only if $ni$ is within the transmission range of $nJ$. The open neighbor set of node $ni$ is denoted by $N(ni)$, which contains all neighbors of $ni$ excluding $ni$ itself. In addition, we define a collection of neighbors of the nodes on path $p$ by $N(p) =$ UvrEp $N\ (ni-)$ An adversary is denoted as $.A$, 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. The proposed protocol does not require the unit disk model. As long as the link is bi-directional, the communication range of each node is not necessarily the same. The network topology is assumed to remain in a single communication session. The link cost is assumed to be uniform, since the primary objective of avoidance routing is discovering 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 [16].

We assume that each node knows about the existence of an adversary with a certain probability, if she is on its neighbor list. Finding adversaries in a local area is possible by anomaly detection [17], [18]. Thus, the probability of each nodedetecting adversaries is parameterized by the adversary detection rate. Note that recent research on the anomaly detection [18] for ad hoc networks shows that more than 80 percent of malicious nodes can be detected. 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 by flooding adversary disjoint route discovery requests.

In this paper, the collusion attack is considered, where a set of connected adversaries can collude to eavesdrop on data. As far as we know, no researc h 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 of the others, however, we can design a k-path route discovery protocol that guarantees that *k* paths have no common adversary; we elaborate on this in Section IV.

## C. Adversary Model

In our model, adversaries are assumed to have unbounded com putational power. While an encryption scheme protects data against adversaries with bounded computational resources, e.g., power, time, and memory storage, this is not 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 $A$ is node n;'s neighbor, i.e., $A \in N(n)$, then $A$ can eavesdrop on n;'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. In the multiple paths scenario, if any of the k paths from $n_s$ to destination $n_d$ contains an adversary as an intermediate node, she can deny forwarding to prevent $n_d$ from assembling the original message.*

Attack 2 tells us that a routing path (or any path *in k* paths) 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 perfec t enc ryption scheme shall defend encrypted data from eavesdropping, i.e., Attack 1. However, even with an ideal encryption, a routing protocol is not able to defend against 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 co ndition that even an ideal routing protocol, and thus any routing protocol, requires secure delivery of a message to the destination agai nst Attacks 1 and 2, as foUows:

*Condition 1 (The Bounded Condition): Given source $n_s$ and destination $n_d$, no adversary (or no 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 secure 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 by contradiction. Assume that Condition 2 does not hold, i.e., there exists cut vertices by a set of adversaries which disco nnect the source and destination nodes. Then, there must exist a routing protocol that securely delivers a message under Attacks 1 and 2. No adversary can be used as an intermediate node, and the removal of al1 the adversaries will disconnect the source and destination nodes when Condition 2 does not hold. However, no routing protocol can deliver a message if two nodes are disconnected in a graph. This is a contradiction. Therefore, the above claim must be true. This concludes the ■

Figure 1 depicts a network of 7 nodes which include adversary $A_1$, and $A_1$ is also a cut vertex. The removal of $A_1$ divides the network into two disjoint components. One contains $n_s$, n1, and n2; the other contains $n_d$, n3, and n4. Since any routing path from $n_s$ to $n_d$ must traverse $A_1$, no ro uting protocol can securely deliver a message between $n_s$ and $n_d$,

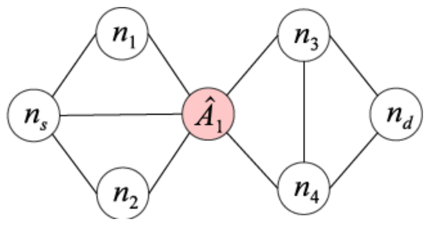

Fig. I . A cut vertex.

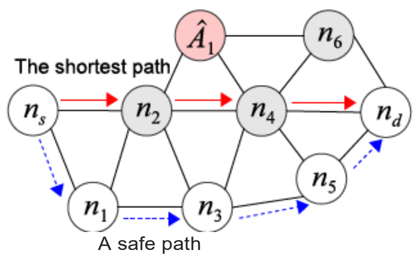

Fig. 2. A safe path.

## *F. Avoiding Eavesdroppers*

1n reality, an ideal routing protocol wi!h a perfect encryption scheme does not exist. Adversary *A* can intercept data transmitted via *p* when *A* E *N(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* $n_8$ *and destination nd, a path, denoted by p, is said to be safe against eavesdropping, if A N (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 $n_8$ and *nd* is *ns* -> n2 -> *n4* -> *nd.* However, this path is not safe because *Ai* can eavesdrop on the transmission of n2 and *n4.* On the contrary, the path $n_8$ -> *ni* -> $n_3$ -> $n_5$ -> *nd* is safe, since *Ai* is not in the proximity of any node on the path.

The existing avoidance protocols [7], [8] 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* $n_8$ *and destination nd, there is at least one safe path ( Definition 1) between* $n_8$ *and nd.*

It is clear that Condition 2 is much more strict than the bounded condition provided by Condition I. Unfortunately, all of the single-path routing protocols with a polynomial encryption scheme, including [7], [8], 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 by contradiction. Assume that Condition 2 does not hold, i.e., at least one safe path does not exist between the source and destination nodes. Then, there must exist a single-path routing protocol that securely delivers messages under Attacks 1 and 2. Since adversaries have unbounded computational power, all the traffic transmitted from their neighbors can be compromised. Thus, the nodes which have at least one adversary in their neighbors cannot be used as intermediate nodes. The removal of the adversaries, as well as their neighbors, will divide a network into two disjoint components, when Condition 2 does not hold. However, no routing protocol can deliver messages, if two nodes are disconnected in a graph. This is a contradiction, and the above claim must be true. This completes the proof. ■

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. A method that relaxes Condition 2 is critical in designing a practical avoidance routing protocol.

## III. MULTI-PATH AVOIDANCE ROUTING FRAMEWORK

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

### *A. The MPAR Framework*

The proposed MPAR framework incorporates the idea of multi-path routing and the XOR coding. The pseudo code of the **MPAR** framework is presented in Algorithm 1. First, the **MPAR** framework tries to find a safe path 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* 2'. 2), source node $n_8$ randomly generates bit strings, mi, m2, . . . , and *mk-i,* by a random number generator with the uniform distribution, denoted by G en u(lml 1) . Here , the input, i.e., lmll, is the length of a bit string that the generator returns, and therefore, lm l = 1ml for 1 :S i :S *k* - I. Then , *ns* computes *mk* by taking *m* EB mi EB m2 EB ... EB *m k - i* - Finally , each message *mi* is sent to the destination via a different path *p;,* where 1 :::; i :::; *k.*

When the destination node receives all the pieces, **it** assembles *m* from *m;* (1 :S i :S *k)* by taking the XOR operation. Note that *mk* is a random string to a third party since the messages, mi, m2, ... , ***mk-i,*** **are** randomly generated. Therefore, an adversary cannot recover *m* unless she obtains all of mi, m 2, ... , and *m k* . Those readers interested in the proof of this property are referred to [13] for details.

### *B. The Condition of The MPAR Framework*

1n 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 ns and destination nd, a set of paths P(ns,nd) = {pi,*

**Algorithm 1** *M PAR(n ₈, nd,* m)

```
1: /* Node n₈ executes the following: */
2: /* The route discovery phase */
3: P(n ₈ , nd) +- RouteDiscovery(nd)-
4: /* The message forwarding phase */
5: if there is a safe path pin P (n ₈ , nd) then
6:     /* The single-path mode */
7:     n₈ sends m via p.
8: else if there are adversary disjoint paths in P(n ₈, nd) then
9:     /* The multi-path routing mode */
10:    k +- IP(ns, nd) I
11:    n₈ randomly generates mi for 1 :::; i :::; k - 1 by
       Gen,,(lml l) .
12:    n₈ computes mk = m EB m1 EB m2 EB . . . EB mk-1 ·
13:    n₈ sends mi via p; for each i (1 :::; i :::; k).
14: else
15:    /* There is neither a safe path nor adversary disjoint paths
       */
16:    n₈ discards m.
```

p2 , . . . ,*pk} is said to be adversary disjoint paths if and only if there is no common adversary A for all of the k paths, i.e., A¢ N7p;).*

When there is no safe path between *n ₈* and *nd,* the protocol enters the k-path routing mode. The condition in which the *k*-path mode securely delivers a message from a source to a destination is given as follows:

*Condition 3: Given source n₈ and destination nd, there is at least one set of adversary disjoint paths (Definition 2) between n ₈ and nd*

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 bold for successful message delivery.

*Theorem 3: Given source n₈ and destination nd, the MPAR framework can securely deliver a message from n ₈ to nd if and only if either Condition 2 or Condition 3 is met.*

*Proof'* We will prove the forward direction by contradiction. Assume that the **MPAR** framework can securely deliver a message from *n ₈* to *nd,* but neither Condition 2 nor Condition 3 bolds. When Condition 2 does not bold, the removal of all the adversaries disconnects *n ₈* and *nd* in the graph. Thus, no routing protocol can deliver a message from *n₈* to *nd* . When Condition 3 does not hold, the removal of one of the adversaries, say *A,* in the network and her neighbors *N(.A)* will divide the network into two disjoint components, i.e., *n ₈* and *nd* are disconnected in the graph. *This* is because there will be at least one path without the common adversary *A,* if *n ₈* and *nd* are not disconnected. However, when *n ₈* and *nd* are disconnec ted in the graph, no routing protocol can deliver a message between them. This is a contradiction. Thus, the forward direction of the above claim must be true.

Next, we will prove the backward direction by contradiction. Assume that either Condition 2 or Condition 3 is met, but the **MPAR** framework cannot securely deliver a message from *n ₈* to *nd,* When Condition 2 holds, there must exist at least one path along which nodes have no adversary as their neighbor. However, according to Algorithm 1, the MPAR framework shall succeed by forwarding a message via one of the safe paths. When Condition 3 holds, there must exist at least one set of adversary disjoint paths. This implies that no adversary obtains all the pieces of a message. Thus, the MPAR framework shall succeed by forwarding encoded messages via the set of adversary disjoint paths. This is a contradiction. Therefore, the backward direction of the above claim must be true.

Both the directions are proven, and these conclude the proof. ■

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 SGc2 be the set of all the network graphs that satisfy Condition 2, and let SGc3 be the set of all the network graphs that satisfy Condition 3. The MPAR framework succeeds in S Gc2 U SGc 3, while a single-path-based approach succeeds only in SGc2.

The proof is by contradiction. Assume the **MPAR** framework requires a stronger condition than any single-path protocol. Then, S Gc2 U SGc3 c S Gc2 must bold. However, the set of graphs S Gc3\ SGc2 is not empty, since a counter example is shown in Figure 3. In Figure 3, there are two paths between *n ₈* and *nd*. Neither of the paths is safe, but they are adversary disjoint paths. Thus, SGc2 U SGc3 c S Gc2 never holds. This is a contradiction. Therefore, the above claim is true. *This* completes the proof. ■

## *C. 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 the single-path protocol works, the single-path mode of MPAR, which is basically a single-path protocol, should work equally well as shown in lines 6 and 7 in Algorithm 1. If the single-path protocol does not work, our MPAR still has a chance to securely deliver a message by the multi-path mode. We derive Theorem 5 as follows.

*Theorem 5: The **MPAR** framework introduces additional message overhead, only when the single-path protocol does not work.*

*Proof'* The proof is by contradiction. Assume that the **MPAR** framework introduces message overhead, but the single-path protocol succeeds. According to Algorithm 1, the MPAR framework starts with the single-path mode and routes messages via single path whenever a safe path is discovered. This implies that the MPAR framework does

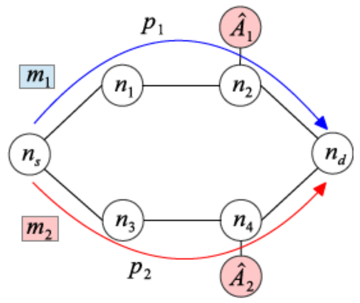

Fig. 3. Adversary disjoint paths.

not introduce additional message overhead, whenever the single-path protocol works. This is a contradiction. Therefore, the above claim must be true. This concludes the proof. ■

## IV. TIMER-BASED MULTI-PATH AVOIDANCE ROUTING

In this section, we introduce our timer-based k-path route discovery protocol, which is the key component of the MPAR framework. To be specific, this is an implementation of *RouteDiscovery(.)* in Algorithm 1. The **MPAR** framework incorporating the proposed timer-based path discovery is called Timer-based **MPAR (TMPAR).**

### *A. Protocol Overview*

The pseudo code of the timer-based k-path route discovery protocol is presented in Algorithm 2. Unlike the preliminary work [13], the proposed scheme uses a defer timer to discover a set of adversary disjoint paths with fewer adversaries. In addition, optimization by avoiding unnecessary route discovery is incorporated to reduce the control overhead.

Similar to the route request phase of DSR [19] and AODV [20], 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 process is repeated until $k$ adversary disjoint paths are discovered, or the amount of flooding exceeds $kmax$. By doing this, a safe path or $k$ adversary disjoint paths are discovered, if they exist.

### *B. Single Path Discovery Mode*

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 $p_1, p_2, \ldots, p_k$, 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 $k \leq kmax$.

**Algorithm 2** kPathRouteDiscovery($n_s$, $n_d$, $kmax$)

```
1: /* Node $n_s$ executes the following: */
2: $n_s$ broadcasts $RREQ_1 := (n_s, n_d)$.
3: On receiving $RPLY_1$ via $p_1$.
4: if $AS(p_1) := RPLY_1.AS$ is empty then
5:     return $p_1$. /* $p_1$ is a safe path. */
6: if $|AS(p_1)| \geq 1$ or timeout is detected then
7:     if $k > |AS|$ and $k \geq 3$ then $n_s$ terminates the discovery
       phase.
8:     /* $n_s$ tries to find adversary disjoint paths */
9:     $k \leftarrow k + 1$.
10:    divide $AS(p_1)$ into $k-1$ pieces, say $AS_i$ for $1 \leq i \leq$
       $k-1$.
11:    $n_s$ broadcasts $(k-1)$ requests, $RREQ_i$
       $(n_s, n_d, AS_{i-1})$, for $2 \leq i \leq k$.
12:    $n_s$ sets a timer.
13:    On receiving all $RPLY_i$ via $p_i$ for $2 \leq i \leq k$.
14:        return adversary disjoint paths $(p_1, p_2, \ldots, p_k)$.
15: else if $k$ exceeds $kmax$ then
16:        return an empty set. /* No path is found */
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 $n_i \neq A$ and $n_i$ has no routing entry for $(1, n_i, n_d)$ then
22:     $n_i$ creates an entry $(1, n_i, n_d, n_j, \text{null})$.
23:     $n_i$ sets a defer timer and broadcasts $RREQ_1$ when
        expires.
24: else
25:     $n_i$ drops $RREQ_1$.
26: On receiving $RPLY_1$ 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 $\forall A \in N(n_i)$ to $RPLY_1.AS$.
29:     $n_i$ sends $RPLY_1$ to the predecessor.
30: On receiving $RREQ_k$ from $n_j$ for $p_k$.
31: if $n_i \neq A$, $n_i$ has no routing entry for $(k, n_i, n_d)$, and
    $N(n_i) \cap RREQ_k.AS$ is empty then
32:     $n_i$ creates an entry $(k, n_i, n_d, n_j, \text{null})$.
33:     $n_i$ sets a defer timer and broadcasts $RREQ_k$ when
        expires.
34: else
35:     $n_i$ drops $RREQ_k$
36: On receiving $RPLY_k$ 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 $RPLY_k$ to the predecessor.
39:
40: /* Node $n_d$ executes the following: */
41: On receiving $RREQ_k$ from $n_i$.
42: if this is the first time to receive $RREQ_k$ and there is no
    adversary in $N(n_d)$ then
43:     $n_d$ sends $RPLY_k$ to $n_i$.
```

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., $n_8$ and $nd$ . Consider the case that an intermediate node, *ni,* receives the request packet from node *ni*. On receiving $R\ RE\ Q_1$, *ni* 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 *kmax* (the ID assigned by *RREQk)-* At this time, *n;* sets the path ID to be 1, the source ID to be $n_8$, the destination ID to be *nd,* and the predecessor ID to be *nj,* respectively. The descendant ID, which is set during the reply phase, is kept null at this moment. If there exists the corresponding entry, *ni* just discards $R\ RE\ Q_1$. In addition, if *ni* 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.

Note that the proposed k-path discovery protocol uses a defer timer in order to select a better path, as shown in lines 23 and 33 in Algorithm 2. We will elaborate on this technique in Section IV-E.

When the destination node, *n,1,* receives $RREQ_1$ the first time, it replies with a reply packet, denoted by $RPLY_1$. The reply packet contains the source ID, the destination ID, and a set of adversary IDs. The reply packet is routed along the predecessor ID at each intermediate node. Consider the case that an intermediate node *ni* receives the reply packet from *ni* . On receiving $RP\ LY_1$, *ni* stores *ni* as the descendant ID. If *ni* has adversaries in its neighbors, the adversary IDs are added to the set of adversary IDs, which is denoted by *AS* in $RP\ LY_1$. Then, Tii sends $RPLY_1$ to the predecessor node.

If there exists a path that satisfies Condition 1, $n_8$ will receive $RPLY_1$. Let p1 be the path found in the first request phase. Now, $n_8$ has a list of adversaries on $p_1$.

If *RP LY.AS* is empty, p1 is a safe path.Thus, by Condition 2, *k Pat hRout eDi scovery(n* $_8$ *, nd)* returns a single path Pl, and $n_8$ simply sends *m* via p1. When $n_8$ does not receive a reply from *nd,* any path from $n_8$ to *nd* 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, *k Pat hRout eDi scovery(n* $_8$ *, nd)* returns an empty set.

## C. Multi-Path Discovery Mode

When p1 contains at least one adversary, $n_8$ will try to find another path. The *k-th* request phase can be generalized as follows. A *k-th* request packet, denoted by *RREQk,* include the source ID, the destination ID, and a list of adversaries, i.e., *(n* $_8$ *, nd,* AS(p1)). Similar to the first request packet, $n_8$ broadcasts *RREQk-* When an intermediate node, say *ni,* receives *RREQk* from *nj, ni* first computes *N (n i )* n *N(p;).* If the intersection is not empty, *ni* has at least one common adversary with some neighbors on p1, *pz, ... ,* or *Pk-I,* and therefore, *n;* drops *RREQk-* Otherwise, *ni* can be an intermediate node of *Pk,* and it creates a new routing entry with the same format as the first request packet. If there exists an adversary disjoint path from *PI, pz, ... ,* and *Pk-I, RREQk* will reach *nd* . 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 lines 13 to 14 in Algorithm 2, source node $n_8$ will receive the route reply *RPLYk* from *nd* . Thu s, *k Pat hRout eDi scover y(n* $_8$ *, nd)* will return a set of paths, Pl, *pz, ... ,* and *Pk·* Otherwise, there are no adversary disjoint paths, and an empty set will be returned since the route discov- ery fails. The failure of k-path discovery request is detected by a timeout event. Since intermediate nodes drop request packets if there is at least one common adversary in *AS (p1 ), RPLYk* is returned to $n_8$ only when there exist adversary disjoint paths.

If a timeout event is detected, $n_8$ starts sending *(k +* 1)-path discovery request packets. In general, the set of adversaries along the first path *AS(pi)* is divided into *(k -* 1) pieces, say *ASi* for 1 i *k-1*. Then, (k-1) request packets, *RREQi* for 2 i *k* with each containing *ASi* for 1 *j k* - 1, are flooded. If $n_8$ receives *RPLYk* from *nd,* then *k* adversary disjoint paths are set up during the reply phase.

In addition, when there is no chance to discover a set of adversary disjoint paths, $n_8$ terminates the route discovery phase to eliminate unnecessary control overhead, as indicated in line 7 in Algorithm 2. We will elaborate on this in Section IV-F.

## D. Message Forwarding Phase

Based on the result of the route discovery, i.e., $p = \{p_1,$ $pz, \dots ,Pk\}$, or an empty set, source node $n_8$ sends *m* by either the single-path mode or the k-path mode, or refrains from message transmission. When a safe path is found, $n_8$ simply sends *m* via *p* as described from lines 6 to 8 in Algorithm l.

When *PI, pz, ... ,* and *Pk,* which are adversary disjoint, are returned, $n_8$ enters the multi-path mode presented from lines 9 to 13 in Algorithm l. A set of bit strings, say m1, m2, ... , and mk-l, arerandomly generated by Gen,..(lm l), and then *mk* is computed by taking m EB m1 Ef)m 2 EB ... EBm k- 1· Note that *mk* seems to be random, becausem1, m 2, ... , and *mk* are random strings. Then, $n_8$ sends *mk* via *Pk·* Since *PI , pz, ... ,* and *Pk,* are adversary disjoint paths, no adversary can obtain all the m1, m 2, ... , and *mk,* and hence cannot recover *m* even if she has access to one of them. On the contrary, *nd* can assemble *m* by taking m1 EB m2 EB ... *Ef)m k.*

If there is no available path, the route discovery returns an empty set, and $n_8$ discards mas shown from lines 14 to 16 in Algorithm 1.

## E. Greedy Node Selection

Our preliminary work [13] tends to return the shorter path, but not the path with fewer adversaries. In order to prioritize the nodes without adversaries in their proximity, we use a defer timer which is widely used as greedy selection with no additional control packet [2 1].

Let *Nad v* be the set of adversaries in the neighbor list of a node. We define the defer timer, denoted by *Td,* which is initiali zed as Equation 1.

$$Td = Rand() \bmod Tmax + |N_{adv}| \cdot Tmax \qquad (1)$$

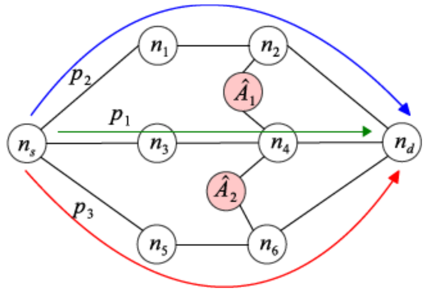

Fig. 4. Example of the k-pathdiscovery.

Here, *Rand()* is a random number generator that returns an integer value and *T max* is the maximal jitter period. Note that *Tmax* is set to be 4 msec in ns-2 [15]. The value of *Td* increases in proportion to the number of adversaries |Nadv|-

At every request packet forwarding process, a node starts the timer. When the timer expires, the node broadcasts a request packet. With this defer timer, the nodes with fewer adversaries as their neighbors will forward request packets earlier than the ones with more adversaries. Therefore, the nodes with fewer adversaries have a higher probability to be selected as the intermediate nodes of the first path. As a result, the TMPAR requires fewer adversary disjoint paths than does the original MPAR, leading to the lower message transmission cost and control overhead.

*Example:* We will show an example of k-path discovery with Figure 4. As we will explain later, the nodes with fewer adversaries as their neighbor have higher probability to be selected as intermediate nodes. Consider that path p2, which contains a set of adversaries *AS(p2 )* = {.Ai} , is discovered in the first route request phase. In the 2-path discovery phase, *RREQ2* with *RREQ 2.AS* := *AS (p2)* is flooded. Since none of the nodes along path *p3* has the common adversary as *AS(p2 ),* path *p3* will be identified by the 2-path discovery phase. By doing so, adversary disjoint paths, p1 and p2 , are discovered.

## *F. Reducing Unnecessary Discovery Phases*

Io **MPAR,** the k-phase discovery phase is repeated until the value of *k* reaches *kmax* However, in the case where there is no possibility that adversary disjoint paths can be found, the protocol should terminate sending further request packets to reduce unnecessary control overhead.

For example, consider that the first path is returned with a set of adversaries, say *AS* = { A.}. Since the first path contains one adversary, the 2-path discovery phase seeks to find a path that avoids *A* in the proximity of any intermediate node. If no such a path can be found, there is no possibility that the k-path discovery mode will discover *k* adversary disjoint paths for *k* 2 3. Thus, sending further route discovery packets is a waste of network resources. By generalizing this observation, we derive Theorem 6 to reduce unnecessary discovery processes.

*Theorem 6: For k* 2 3, *when k adversary disjoint paths cannot be found for k* > IASI, *it is impossible to discover (k* + 1) *adversary disjoint paths.*

*Proof* The proof is by contradiction. Assume that the k-path discovery phase fails, but the (k + 1)-path discovery phase identifies adversary disjoint paths for *k* > IASI. The first request packet contains IASI adversaries in *RREQ* $_1$*.AS.* When *k* > IASI, the other request packets have exactly one adversary in their adversary set, i.e., *IRREQ ; .AS I* = 1 for 2 i *k.* Note that if there were a request packet with *IRREQ i .AS I* = 0, the k-path discovery phase would have already succeeded. If the k-path discovery fails with *IRRE Qi .ASI* = 1 for some i (2 i *k),* all the possible paths between $n_8$ and *nd* have the common adversary in *RREQ;.AS.* Thus, the *(k* + 1)-path discovery phase will not be able to identify adversary disjoint paths. This is a contradiction. Therefore, the above claim must be true. This completes the proof. ■

Therefore, in the advanced MPAR, the k-path discovery phase continues only when *k* *kmax* and *k* > IASI *fork* 23, as indicated at line 7 in Algorithm 2.

## V. PERFORMANCE EVALUATION

In this section, computer simulations with ns-2 [15] are conducted to evaluate the proposed scheme. Along with our TMPAR with the timer-based k-path route discovery protocol, the ideal protocol, Greedy-AA [8], and the original **MPAR** are implemented in the same way as described in our preliminary work [13].

### *A. Simulation Configurations*

A network is generated by randomly placing nodes in an 800 x 800 square meter region. The number of nodes ranges from 100 to 400, and the effective communication range under the two-ray ground propagation model is set to be 100 meters. One to ten percent of nodes are set to be adversaries. Although a node may overhear signal at a distance more than 100 meters, each node maintains a set of nodes located within effective range (i.e., 100 meters) as neighbors. Thus, the average number of neighbors of each node ranges from 4.9 to 19.6. The adversary detection rate ranges from 0.8 to 1.0. Unless specified, each node is assumed to know the existence of adversaries in its proximity. The IEEE 802.11 is used as the MAC protocol, and the transmission rate is set to be 11 Mbps. The interface queue length at the media access is set to be 50, and the drop tail is applied. At the source node, the constant bit rate (CBR) generates a 128-byte message every 0.25 seconds (i.e., four messages per second) and sends them with UDP datagrams. The total number of packets generated by the source node is 100. In addition to CBR, junk traffic is generated at every node. Each junk packet is of 128-byte, and the number of junk packets that each node receives ranges from 10 to 50 per second.

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 bop.

## *B. The ns-2 Implementation of MPAR and TMPAR*

In our ns-2 implementation, three kinds of packets, route request (RREQ), route reply (RPLY) and data (DATA), are defined in MPAR and TMPAR. Among them, RREQ and RPLY are called control packets to discover a safe path or a set of adversary disjoint paths. In these packets, the packet type (RREQ or RPLY of 1-byte), a path mode (1-byte), a path ID (1-byte), a source node ID (4-byte), a destination ID (4-byte), and a set of adversaries' IDs (4-byte each) are defined. In addition, to these pieces of information, a DATA packet contains a sequence number and the data field to store bytes from the transport layer. Note that an **IP** header (20-byte) is included in RREQ and RPLY packets, and both an IP header and UDP header (20-byte) are contained in a DATA packet. In the original **IP** header , there are 4-byte padding fields to make the size of an IP header a multiple of 32-bit. Therefore, the packet types (i.e., RREQ, RPLY, and DATA), path mode, and path ID are embedded into the padding field. In addition, an IP header contains the source and destination IDs, and thus, we avoid double counts for these fields. Therefore, our implementation increases the header size of an **IP** datagram by $4 \times |AS|$ bytes, where $|AS|$ is the size of the adversary set **in RREQ** or **RPLY.**

Each node maintains a routing table in their routing module. Each entry of a routing table contains the path mode, a path ID, a source ID, a destination **ID,** the predecessor node ID, and the descendant node ID. Each entry is uniquely identified by a 4-tuple (a path mode, a path ID, a source node ID, and a destination node ID). In addition, the open neighbor set within the effective communication range is maintained as a set of integers in the routing module.

## *C. Performance Metric*

The performance metrics, the message delivery rate, the delay, hop stretch, the amount of traffic, and the number of control packets are used as follows:

- The message delivery rate is defined as the ratio between the number of messages received by the destination divided by the total number of messages that the source node sent out. The ideal protocol succeeds when Condition 1 is met, and this provides the upper bound of the delivery rate. Greedy-AA will find a safe path only when Condition 2 is met. The proposed MPAR with the k-path route discovery protocol will successfully find a safe path or a set of adversary disjoint paths when either Condition 2 or Condition 3 holds. That is, there exists a safe path or a set of adversary disjoint paths. Note that our simulation is conducted in a practical setting, where the IEEE 802.11 is employed as a MAC protocol. Therefore, due to packet collisions, routing may fail even if these conditions are met.
- The delay is defined as the time required to deliver a message from the source to the destination. Since each CBR traffic generates 100 messages, the delay is computed for individual messages and then the average delay is obtained. For MPAR and TMPAR, a message from the CBR module might be encoded into up to $k_{max}$ coded messages. Let $t_s$ be the time that CBR traffic generates message $m$. Assume that the source node sends $m_1, m_2, \ldots,$ and $m_k$ with the k-path mode, and the destination receives each of them at $t_1, t_2, \ldots,$ and $t_k$, respectively. In this case, $\max_{1 \le i \le k}\{t_i\} - t_s$ is set as the delay of $m$. Note that the path discovery time is not included in the message delay.
- The hop stretch is defined as the ratio between the actual number of hops and the lowest number of bops 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. For a randomly generated graph and a pair of source and destination nodes, the shortest number of hops is obtained by the breadth first search with a global view, and then, the hop stretch is computed from the resulting paths by each avoidance protocol. For our **MPAR** and TMPAR protocols, there may be $k$ paths, say $P_1, p_2, \ldots,$ and $P_k$, and hence the number of hops is defined as the largest one, i.e., $max\{|p_1|, |P_2|, \ldots, |P_k|\}$
- The amount of traffic is defined as the amount of CBR traffic transmitted among nodes. That is, the size of MAC frames of sending events in an ns-2 trace file is counted. In MPAR and **TMPAR,** each transmission of a MAC frame containing a CBR packet is of 216-byte including UDP, IP, MPAR/fMPAR, MAC headers, and PHY preamble. This metric is somewhat related to the number of hops, because the number of message transmissions increases as the number of hops increases. The difference is that the CBR traffic is computed from all the transmission events in the MAC layer in a ns-2 trace file. There could be multiple transmissions of the same message due to the ACK mechanism of IEEE 802.11. In MPAR and TMPAR, the sum of the number of message transmissions via all the $k$ paths is considered as the amount of traffic.
- The control overhead is defined as the number of route request packets and reply packets in the route discovery phase. In on-demand single-path-based protocols, the number of request packets will be the number of legitimate nodes in a network; the number of reply packets equals the number of hops between source and destination nodes. In our multi-path-based protocols including MPAR and **TMPAR,** the flooding of request packets is performed several times to discover $k$ paths. Note that the size of the control packets in a routing protocol is generally small. Thus, we consider the number of request and reply packets as the control overhead, rather than the amount of control packet traffic. This is because the introduction of more control packets will increase the chance of packet collisions in a MAC protocol.

## *D. Simulation Results*

Figure 5 shows the delivery rate with respect to the number of neighbors. The percentage of adversaries is set

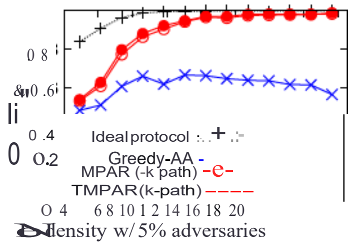

Fig. 5. The delivery rate.

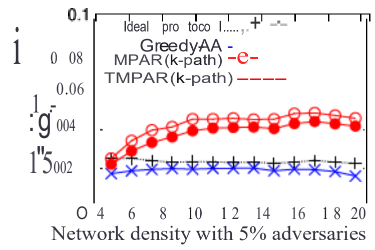

Fig. 7. The end-to-end delay.

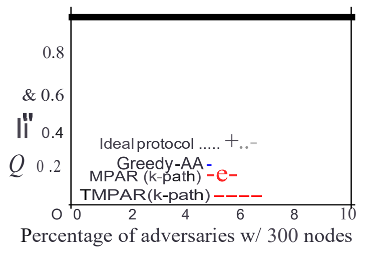

Fig. 6. The delivery rate.

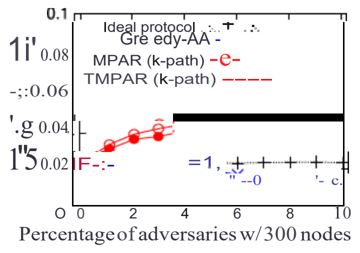

Fig. 8. The end-to-end delay.

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/fMPAR significantly improves the performance compared with Greedy-AA, and the delivery rate is very close to the ideal protocol when the number of neighbors is larger than 15. Note that MPAR and TMPAR result in mostly the same delivery rate in the case that adversaries do not collude. This indicates that MPAR eventually finds a set of adversary disjoint paths as TMPAR does. However, as we will show later, TMPAR provides higher delivery rates under the collusion attacks than MPAR and effectively reduces the cost of message transmission as well as that of path discovery.

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/fMPAR** signi ficantly improves the routing opportunity.

Figure 7 illustrates the end-to-end delay with respect to the number of neighbors. Both MPAR and **TMPAR** increase the delay as the network density increases . When the network density is low, the probability of a safe path or a set of adversarydisjoint paths is small. As a result, route discovery succeeds only when the source and destination nodes are close to each other. This implies that the end-to-end delay is small if the number of neighbors is small. Compared with the ideal protocol and Greedy-AA, the proposed **MPAR** and **TMPAR** pro tocols cause longer delay. This is mostly due to the processing delay which shall occur in the XOR coding process in order to encode an original message into several pieces, each of which takes 4 milliseconds.

Figure 8 presents the end-to-end delay with respect to the percentage of adversaries. As shown in the figure, the delay of MPAR and TMPAR increases as the percentage of adversaries increases. Intuitively, the probability of a single safe path being found becomes smaller, when the number of adversaries increases. In other words, more encoding operations are involved in the routing process. Greedy-AA incurs the smallest delay. Considering the extremely low delivery rate of Greedy-AA in Figure 6, Greedy-AA cannot securely deliver a message unless the source and destination nodes are very close to each other.

Figure 9 depicts the bop 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 1.6. TMPAR slightly reduces the hop stretch by selecting the nodes with fewer adversaries in the first path discovery. Considering the improvement of the delivery rate shown in Figure 5, this overhead is acceptable. Note that the ideal protocol does not always result in the shortest path due to the randomness in the physical and MAC layers, e.g., propagation, contentions in media access, and the queue delay.

Figure 10 plots the hop stretch with respect to the percentage of adversaries. The number of nodes is again set to be 300. The hop stretch of **MPAR/fMPAR** 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 bops 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, from Figures 6 and 10, it can be seen that our **MPAR/fMPAR** can still securely deliver a message with a greater number of hops.

Figure 11 illuminates the amount of traffic with respect to the number of neighbors. As can be seen in the figure, **MPAR** inc urs more traffic than the other protocols. However,

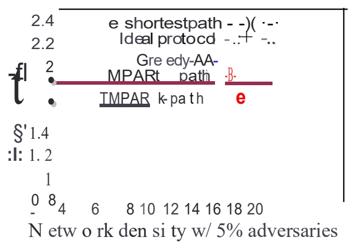

Fig. 9. The hop stretch.

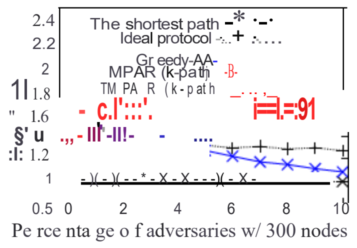

Fig. 10. The hop stretch.

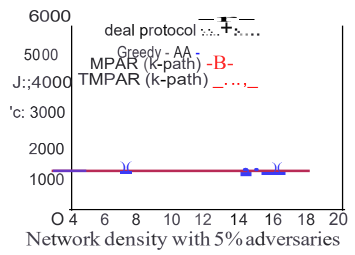

Fig. 11. The amount of traffic.

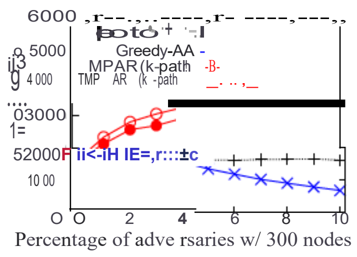

Fig. 12. The amount of traffic.

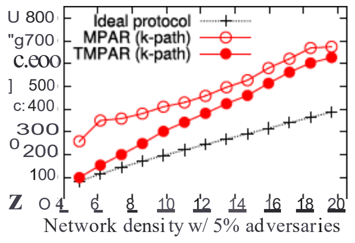

Fig. 13. The control overhead.

we would like to emphasize that the larger amount of traffic does not necessarily mean that MPAR and TMPAR are 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 III-C, TMPAR does not introduce extra traffic in a network where Greedy-AA securely delivers a message. When a safe path doesnot exist between a source and a destination, TMPAR introduces additional traffic overhead to discover adversary disjoint paths for the k-path routing mode for securely delivering a message.

Figure 12 gives the amount of traffic with respect to the percentage of adversaries. Apparently, the amount of traffic resulting from our MPAR/TMPAR 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.

Figure 13 provides the control overhead for different protocols with respect to the number of neighbors. Note that Greedy-AA is a proactive protocol and its result is not plotted since it is not fair to compare the control overhead of proactive and on-demand protocols. In addition, a proactive protocol requires $O(n^2)$, where $n$ is the number of nodes in a network, for each node to collect routing information. As shown in the figure, the control overhead increases as the network scale increases. This is because the entire network is flooded by RREQ packets. Since the MPAR and TMPAR protocols require a number of flooding events, the message overhead is much larger than the ideal protocol. By terminating the $k$-path route discovery when there is no possibility of finding adversary disjoint paths, TMPAR results in lower control overhead than MPAR.

Figure 14 demonstrates the control overhead for different protocols with respect to the percentage of adversaries. The number of control packets required by the ideal protocol is the number of nodes minus the number of adversaries, and thus, the control overhead slightly decreases as the percentage of adversaries increases. On the other hand, the greater the number of adversaries in a network, the greater the number of paths MPAR and TMPAR require to fulfill the adversary disjoint property in order to securely deliver messages. As a result, MPAR and TMPAR incur higher control overhead when the percentage of adversaries is high.

## *E. Impact of Collusion Attacks*

Figure 15 shows the delivery rate of 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 noteworthy that the delivery rate decreases when the number of neighbors is greater than or equal to 14. This is because more neighbors indicate that more adversaries are connected to each other to collude. With the use of a defer timer to select the nodes with fewer adversaries, TMPAR has a slightly better delivery rate compared with MPAR. From Figures 5 and 15, MPAR and TMPAR do not show any significant performance degradation due to collusion attacks.

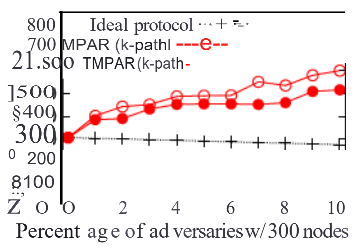

Fig. 14. The control overhead.

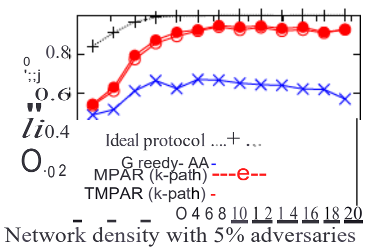

Fig. 15. The delivery rate under collusion attacks.

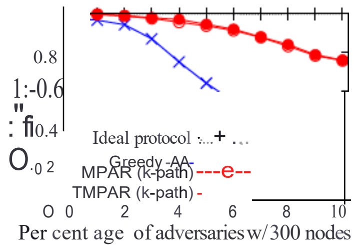

Fig. 16. The delivery rate under collusion attacks.

Figure 16 illustrates the delivery rate of collusion attacks with respect to the percentage of adversaries. Compared with Figure 6, the delivery rate of **MPAR/f MPAR** decreases by up to 13%. However, the proposed solution still maintains a higher delive ry rate than that of Greedy-AA. Thus, we claim that **MPAR** can accommodate a certain degree of collusion attacks.

## F. Impact of Adversary Detection Rate

Figures 17 and 18 present the delivery rate with respect to the successful adversary detection rate with no collusion attack and with collusion attack. When the adversary detection rate is 1.0, the delivery rate of TMPAR is higher than that of MPAR. This is because the probability that TMPAR discovers a single safe path is higher than that MPAR does. On the other hand, when the adversary detection rate is smaller than 1.0, some undetected adversaries may clandestinely listen to the channel. Since there is no way to explicitly avoid insecure areas monitored by undetected adversaries, the routing paths should be split as much as possible for higher delivery rate. This observation is clearly shown in Figures 17 and 18. As the adversary detection rate decreases, the delivery rate of MPAR is higher than that of **TMPAR.** Howeve r, regardless of the adversary detection rate, both MPAR and **TMPAR** en joy much better delivery rate than the single-path Greedy-AA.

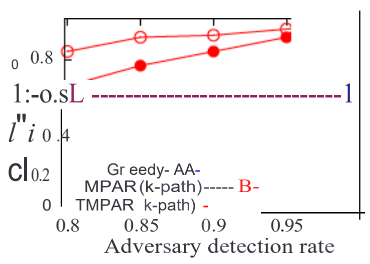

Fig. 17. The delivery rate with different successful adversary detection rate.

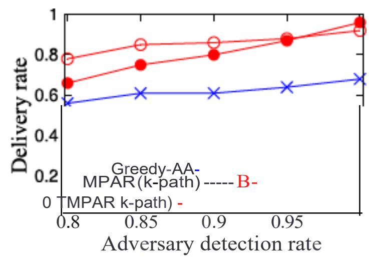

Fig. 18. The delivery rate with different successful adversary detection rate under the collusion attack.

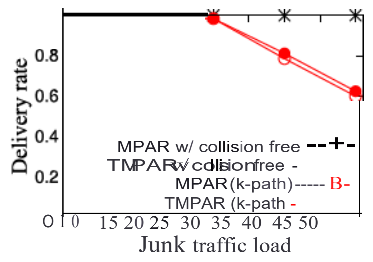

Fig. 19. The delivery rate under junk traffic.

## G. Impact of Traffic Load

Figure 19 demonstra tes the delivery rate with respect to the number of junk packets received at each node. Here, the number of junk packets that each node receives is set to be 10 to 50 per second, and the size of a junk packet is set to be 128-byte. Clearly, adding more junk packets causes a higher probability of collisions. In addition, the message generated by CBR might be dropped at the queue when the junk packets increase. Note that the interface queue length in ns-2 is set to be 50. As a result, the delivery rate decreases as the number of junk packets increases, especially when the traffic load is more than 40 junk packets per second. As shown in the figure, **MPAR** and **TMPAR** under a collision-free environment achieves the 100% delivery, since the network is dense enough to discover a safe path and/or a set of adversary disjoint paths. On the other hand, in a practical setting with junk traffic, both **MPAR** and **TMPAR** still maintain reasonable delivery rate as long as the junk traffic load is less than 30 per second.

## H. The Statistics

Figures 20 and 21 present the number of adversary disjoint paths found by **MPAR** and **TMPAR,** respective ly, with respect to the number of neighbors. For example, when the network density is 14.7 in Figure 20, MPAR requires single safe path 28.7% of the time, 2-path 67.3% of the time, 3-path 1.5%

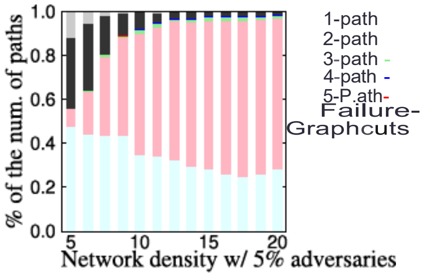

Fig. 20. The ratio of number of paths found by MPAR.

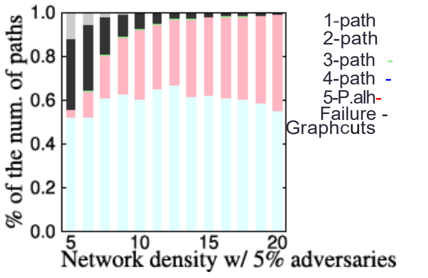

Fig. 21. The ratio of number of paths found by TMPAR.

of the time, 4-path 0.6% of the time, and 5-path 0.2% of the time for secure message delivery. A graph cut by adversaries is not identified when the density is 14.7, and **MPAR** fails 1.7% of the time, i.e., 17 out of 1000 network realizations because there is neither a safe path nor adversary disjoint paths. As can be seen in these fig ures, 1 or 2 paths are required most of the time. This is because it is very rare that 3 adversary disjoint paths are found in 2-dimensional networks.

In Figure 20, the ratio of 1-path decreases as the number of neighbors increases. This is because **MPAR** does not prioritize the nodes using adversaries in their neighbor as intermediate nodes_ As a result, MPAR tends to discover a shorter path for the first discovery phase, and then, the k-path *(k* ::::: 2 ) discovery phase tries to find paths that avoid the adversaries contained in the first path. This trend is not observed in **TMPAR,** which is shown in Figure 21, as **TMPAR** prioritizes the nodes using the number of adversaries in the proximity by the defer timer. To be specific, with the network density 14_7, **TMPAR** securely delivers a message by 1-path 66.7% of the time, 2-path 30.3% of the time, and 3-path 0.2% of the time. Although the delivery rate of TMPAR is the same as MPAR, TMPAR requires a smaller number of paths. Thatindicates that TMPAR incurs lower control overhead to discover adversary disjoint paths as well as lower transmission cost to encode a message.

## VI. RELATED WORK

### *A_ Multi-Path Routing Protocols*

Multi-path routing protocols, such as [22]- [25], send data via a set of node/link disjoint paths [26] from a source to a destination. Finding multiple paths with fewer route discovery packets is studied in [27], [28]. The algorithms in [25] try to find the optimal disjoint paths, where the number of common nodes and links are minimized_ 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 [22], [29]; the other is fault tolerance [23], [24]. 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.

### *B. 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 [30], in which a party can obtain the plaintext from ciphertext if she has $t$ out of $k$ secrets. SPREAD [31] 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 [32], and thus is not appropriate due to the lack of its distributed nature. Another approach is to use secure network coding [33], [34]. However, these works are primarily designed for broadcast, and finding 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 capac- ity in a static network with eavesdroppers of known and unknown locations is presented in [35] and [36], respectively. The work [37] further explores the secure throughput in the case of the multi-path mode and proposes a routing algorithm with cooperative jamming. However, jamming requires an additional operational cost for transmission scheduling and incurs energy consumption; therefore, such an approach is out of our scope.

The idea of combining the XOR coding and multi-path is used in [38]-[40]. However, their route discovery phases identify a physically distanced set of paths, but are not designed to avoid insecure areas and/or malicious nodes.

### C. *Avoidance Routing Protocols*

The work most relevant to this study is that regarding area/nodes avoidance routing [6]- [9], [41]. The work in [6] 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. Alibi routing [9] proves that a packet did not travel insecure areas by alibi, i.e., a packet is located at some routers which are physically apart from the insecure regions. Routing Around Nation-States **(RANS)** protocol [41] prevents packets from traveling routers in opponent nations by forwarding packets around friendly nations. Other works [7], [8] are designed for distance-vector networks. Virtual Positioning Source Routing **(VPSR)** [7] 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 (8) 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.

Recently, the idea of avoidance routing is applied to delay tolerant networks, and contact avoidance routing (CAR) (42) is proposed to securely deliver a message to its destination by avoiding a contact with malicious nodes. However, the model of opportunistic networks is different from the one in ad hoc networks. Thus, they are out of the scope of this paper. In addition, the presence of adversaries can be detected by the nodal behaviors [43).

## VII. 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 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 $m = m_1$ EB $m_2$ EB . . . EB $mk$, and each $mi$ is sent via a different path. By doing this, an adversary cannot have access to $m$ unless she obtains all pieces of the message $mi$ (1 :S; i :S; $k$), while a legitimate receiver can assemble $m$ by the XOR operation. Based on our preliminary work, we develop the timer-based 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 $mi$ for some i (1 :S; i :S; $k$), and that adversaries do not collude with each other. The simulation results using ns-2 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.

This work can be extended into multiple directions. First, **MPAR** can be further improved in a more realistic scenario, where the adversary' s locations are unknown, by selecting a set of physically distanced paths. Second, the idea of avoidance routing with the XOR coding can be applied to securely penetrating a barrier consisting of a set of strategically deployed adversaries. Finally, the proof of avoidance, such as the one in [9], for our **MPAR** design is of significant interest.

## REFERENCES

[I] P. Gutmann, "Lessons learned in implementing and deploying crypto software," in *Proc. Usenix Security Symp.,* 2002, pp. 315- 325.

[2] R. Nojima, T. Kurokawa, and S. Moriai, "XPIA, X.509 certificate public-key investigation and analysis system," N/CT *News,* vol. 435, pp. 5-6, Dec. 2013.

[3] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson, "Users get routed: Traffic correlation on Tor by realistic adversaries," in *Proc. CCS.* 2013, pp. 337-348.

[4] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics,* vol. 38, no. 8, 1965.

[SJ S. Singh, *The Code Book: The Evolution of Secrecy From Mary, Queen of Scots to Quantum Cryptography.* New York, NY, USA: Doubleday, 1999.

[6] E. Kline and P. Reiher, "Securing data through avoidance routing," in *Proc. NSPW,* 2009, pp. 115-124.

[7] H. Zlatokrilov and H. Levy, "Navigation in distance vector spaces and its use for node avoidance routing," in *Proc. INFOCOM,* 2007, pp. 1253-1261.

[8] H. Zlatokrilov and H. Levy, "Area avoidance routing in distance-vector networks," in *Proc. INFOCOM,* 2008, pp. 475-483.

[9] D. Levin *et al.,* "Alibi routing", in *Proc. SJGCOMM,* 2015, pp. 611-624.

[10] X. Liu, L. Xiao, A. Kreling, and Y. Liu, "Optimizing overlay topology by reducing cut vertices," in *Proc. NOSSDAV,* 2006, Art. no. 17.

[11] Y. He, H. Ren, Y. Liu, and B. Yang, "On the reliability of large-scale distributed systems- A topological view," *Comput. Netw.,* vol. 53, no. 12, pp. 2140-2152, 2009.

[12] G. Xie, K. Ota, M. Dong, F. Pan, and A. Liu, "Energy-efficient routing for mobile data collectors in wireless sensor networks with obstacles," *Peer-Peer Netw. Appl.,* vol. 10, no. 3, pp. 472--483 2016.

[13] K. Sakai, M.-T. Sun, W.-S. Ku, J. Wu, and T. H. Lai, "Multi-path based avoidance routing in wireless networks," in *Proc. ICDCS,* 2015, pp. 706-715.

[14] P. C. Pinto, J. Barros, and M. Z. Win, "Wireless physical-layer security: The case of colluding eavesdroppers," in *Proc. JS IF,* 2009, pp. 2442-2446.

[ IS] (2011). *The Network Simulator (NS-2).* [Online]. Available: http://www isi.edu/nsnam/ns/

[16] (1999). *IEEE 802.ll b Standard.* [Online]. Available: http://standards. ieee.org/getieee802/download/802.11b-1999.pdf

[17] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey", *ACM Comput. Surv.,* vol. 41, no. 3, p. I 5, 2009.

[18] B. Gao, T. Maekawa, D. Amagata, and T. Hara, "Environmen-tadaptive malicious node detectjon in MANETs with ensemble learning," in *Proc. ICDCS,* 2018, pp. 556-566.

[19] D. B. Johnson, "Routing in ad hoc networks of mobile hosts," in *Proc. WMCSA,* 1994, pp. 158-163.

*[20]* C. Perkins, E. Belding-Royer, and S. Das, *Ad -Hoc On-Demand Distance Vector (AODV) Routing,* document IETF RCF 3561, 2003.

[21] K. Sakai, S. C.-H. Huang, W.-S. Ku, M.-T. Sun, and X. Cheng, "Timer -based CDS construction in wireless ad hoc networks," *IEEE Trans. Mobile Comput.,* vol. 10, no. 10, pp. 1388-1402, Oct. 2011.

[22] A. Tsirigos and Z. J. Haas, "Analysis of multipath routing- Part 1: The effect on the packet delivery ratio," *IEEE Trans. Wireless Commun.,* vol. 3, no. I, pp. 138-146, Jan. 2004.

[23] P. P. C. Lee, V. Mfara, and D. Rubenstein, "Distributed algorithms for secure multipath routing in attack-resistant networks," *IEEE/ACM Trans. Netw.,* vol. 15, no. 6, pp. 1490--1501, Dec. 2007.

[24] X. Zhang, X. Dong, J. Wu, X. Li, and N. Xiong, "Fault -aware How co ntrol and multi-path routing in wireless sensor networks," in *Proc. ICDCS Workshops,* 2013, pp. 27- 32.

[25] J. Yallouz,0 . Rottenstreich, P. Babarczi, A. Mendelson, and A. Orda, "Optimal link-disjoint node-'somewhat disjoint' paths," in *Proc. /CNP,* 2016, pp. 1-10.

[26] Y. Ge , G. Wang, and J. Wu, "Node-disjoint multipath routing with group mobility in MANETs," in *Proc. NAS,* 2010, pp. 181- 189.

[27] S.-J. Lee and M. Gerla, "Split multipath routing with maximally disjoint paths in ad hoc networks," in *Proc. ICC,* 2001, pp. 3201-3205.

[28] 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.

[29] S. Kompella, S. Mao, Y. T. Hou, and H. D. Sherali, "Cross -layer optimized multipath routing for video communications in wireless networks," *IEEE J. Se/. Areas Commwt,* vol. 25, no. 4, pp. 831-840, May 2007.

[30] A. Shamir, "How to share a secret," *Commu11. ACM,* vol. 22, no. 11, pp. 612-613, Nov. 1979.

[31] W. Lou, W. Liu, Y. Zhang, and Y. Fang, "SPREAD: Improving network security by multipath routing in mobile ad hoc networks," *Wireless Netw.,* vol. 15, no. 3, pp. 279- 294, 2009.

*[32]* R. Bhandari, *Survivable Networks: Algorit/uns for Diverse Routing.* Norwel,l MA, USA: Kluwer, 1998.

[33] N. Cai and R. W. Yeung, "Secure network coding on a wiretap network," *IEEE Trans. Inf Theory,* vol. 57, no. 1, pp. 424-435, Jan. 2011.

[34] K. Jain, "Security based on network topology against the wiretapping attack," *IEEE Wireless Commun.,* vol. 11, no. 1, pp. 68- 71, Feb. 2004.

[35] 0. 0. Koyluoglu, C. E. Koksal, and H. El Gama!, "On secrecy capacity scaling in wireless networks," *IEEE Trans. Inf. Theory,* vol. 58, no. 5, pp. 3000--3015, May 2012.

[36] S. Vasudevan, D. Goeckel, and D. F. Tows ley, "Security -capacity trade - off in large wireless networks using keyless secrecy," in *Proc. Mobihoc,* 2010, pp. 21-30.

[37] <;. <;apar, D. Goeckel, B. Liu, and D. Towsley, "Secret communication in large wireless networks without eavesdropper location information," in *Proc. INFOCOM,* 2012, pp. 1152- 1160.

[38] G. Xia, Z. Huang, and Z. Wang, "Secure data transmission on multiple paths in mobile ad hoc networks," in *Proc. WASA,* vol. 4138, 2006, pp. 424-434.

[39] A. A. Ahmed and N. F. Pisa!, "Secure real-time routing protocol with load distribution in wireless sensor networks," *Security Commun. Netw.,* vol. 4, no. 8, pp. 839-869, Aug. 2011.

[40] C. Wang, S. Cai, and R. Li, "AODVsec: A multipath routing protocol in ad-hoc networks for improving security," in *Proc. ICM/NS,* 2009, pp. 401-404.

[41] A. Edmundson, R. Ensafi, N. Feamster, and J. Rexford, "A first look into transnational routing detours," in *Proc. SJGCOMM,* 2016, pp. 567- 568.

[42] T. Osuki, K. Sakai, and S. Fukumoto, "Contact avoidance routing in delay tolerant networks," in *Proc. INFOCOM,* 2017, pp. 1- 9.

[43] Y. Liu, M . Dong, K. Ota, and A. Liu, "ActiveTrust: Secure and trustable routing in wireless sensor networks," *IEEE Trans. Inf. Forensics Security,* vol. 11, no. 9, pp. 2013- 2027, Sep. 2016.

**Kazuya Sakai** (S'09- M' 14) received the Ph.D. degree in computer science and engineering from the Ohio State University in 2013. He is currently an Associate Professor with the Department of Electrical Engineering and Computer Science, Tokyo Metropolitan University. His research interests include wireless and mobile computing, information and network security, and distributed algorithms. He was a recipient of the IEEE Computer Society Japan Chapter Young Author Award 2016. He is a member of the ACM.

Min-Te Sun (S'99- M' 02) received the B.S. degree in mathematics from National Taiwan University in 1991, the M.S. degree in computer science from Indiana University in 1995, and the Ph.D. degree in computer and information science from the Ohio State University in 2002. Since 2008, he has been with the Department of Computer Science and Information Engineering, National Central University, Taiwan, where he is currently a Full Professor. His research interests include distributed algorithm design and wireless network protocol development.

Wei-S hinn **Ku** (S'02 - M'07-S M' 12) received the M.S. degree in computer science, the M.S. degree in electrical engineering, and the Ph.D. degree in computer science from the University of Southern California (USC) in 2003, 2006, and 2007, respectively. He is currently a Program Director with the National Science Foundation and a Professor with the Department of Computer Science and Software Engineering, Auburn University. He has published more than 100 research papers in refereed international journals and conference proceedings. His research interests include databases, data science, mobile computing, and cybersecurity. He is a member of the ACM SIGSPATIAL.

**Jie Wu** (F'09) was a Program Director at the National Science Foundation and a Distinguished Professor at Florida Atlantic University. He is currently an Associate Vice Provost for international affairs with Temple University. He also serves as the Directorof the Center for Networked Computing and a Laura H. Carnell Professor. He served as the Chair of Computer and Information Sciences from 2009 to 2016. He regularly publishes in scholarly journals, conference proceedings, and books. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. He serves on several editorial boards, including the IEEE TRANSACTIONS ON SERVICE COMPUTING and the *Joumal of Para.Ile[ and Distributed Computin g.* He was a recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award. He was a General Co hair of theIEEE MASS 2006, the IEEE IPDPS 2008, the IEEE ICDCS 2013, ACM MobiHoc 2014, ICPP 2016, and the IEEE CNS 2016, and an Program Co hair of the IEEE INFOCOM 2011 and CCF CNCC 2013. He was an IEEE Computer Society Distinguished Visitor, an ACM Distinguished Speaker, and the Chair of the IEEE Technical Committee on Distributed Processing (TCDP). He is a CCF Distinguished Speaker.

**Ten H. Lai** is currently a Professor of computer science and engineering with the Ohio State University. He is interested in applying Zen to teach and research. He served as the Program Chair of ICPP 1998, the General Chair of ICPP 2000, the Program Co-Chair of ICDCS 2004, a General Chair of ICDCS 2005, and recently, the General Co-Chair of ICPP 2007. He is/was an Editor of the IEEE TRANSACTIONS ON PARALLELAND DISTR IBUTED SYSTEMS, *ACM/Spri11ger Wireless Networks,* the *Academia Sinica's Journal of Information Science and E11gi11eering,* the *Intematio11al Journal of Sensor Networks,* and the */11temational Journal of Ad Hoc a11d Ubiquitous Computing.*