A Routing Algorithm with Path Randomization for Enhanced Security and Balanced Energy Consumption

David C. Yuan
Texas Academy of Math and Science
University of Northern Texas
Denton, Texas 76203
davidyuan@my.unt.edu

Lei Chen
Department of Information Technology
Georgia Southern University
Statesboro, Georgia 30460
lchen@georgiasouthern.edu

***Abstract*: Routers are located at the core of communication networks such as the Internet and sensor networks. The routing algorithms deployed in routers have a profound impact on network security. The traditional shortest-path algorithm used in OSPF and other routing protocols has inherent vulnerability to certain types of attacks. These routing algorithms also influence the life span of the switches and routers with high sensitivity to energy consumption. In this paper, we address these challenges by developing a novel routing algorithm with a randomization process so that packets are sent through optimal yet less predictable paths. It is expected that this process will help increase the network defense against eavesdropping and jamming attacks. It is also expected to improve the energy consumption in sensor networks and similar ad hoc networks.**

Index Terms—network security; routing algorithm; energy consumption; path randomization; wireless network;

I. INTRODUCTION

Routers and similar switching devices are located at the core of modern communication networks such as the Internet and the sensor networks. They provide the vital switching functionality so that the user packets can travel through the networks along the proper paths and eventually arrive at the aimed destinations [1]. With the rapid expansion and security penetration of communication networks, an increasing amount of high-value traffic and mission-critical data is now traveling in these networks, thus making the routers and the network traffic the prime targets of malicious attacks.

One of the most popular conventional routing protocols is the Open-Shortest-Path-First (OSPF) protocol [2]. With this protocol, the routers periodically exchange Link State Advertisement (LSA) messages with one another and learn the current network connectivity information. Using these information, a shortest path algorithm such as Dijkstra's algorithm can determine the shortest or lowest-cost path from every source node to all possible destination nodes [2][3]. The data packets then travel along these paths to reach their respective destinations.

However, this type of protocol is vulnerable to several routing threat actions, including sniffing attacks and traffic analysis [4]. Additionally, a major risk lies in the path predictability. By deploying a sniffer to eavesdrop on the LSA messages exchanged, a hacker can learn the network topology and connectivity information. Consequently the hacker can use a shortest path algorithm such as Dijkstra's algorithm to determine the shortest or lowest cost paths from every source router to all possible destination routers, just as a router does. These are exactly the same paths traveled by the user data packets as determined by the routers. As long as the hacker can gain access to one of the links on these paths, it is then possible to eavesdrop, intercept, or jam the entire communications of the intended victims. If the network topology is relatively stable, the hacker does not even need to re-calculate the paths.

To defend against such type of attacks, among many others, a number of routing protocols have deployed built-in security measures. For example, OSPF deploys a simple password protection and cryptographic authentication [5][6]. However, a weak password is often easily defeated [9]. Routers and network applications may also use Transport Layer Security (TLS) or Internet Protocol Security (IPSec) to encrypt user packets [7][8]. While the encryption technologies provide better protection, they do require more processing capabilities from the routers and user devices; and even the most powerful encryptions can still be defeated by advances in mathematics and computation capabilities [10].

Another weakness of sending all packets on the shortest or lowest cost path is unbalanced energy consumption. If a router is a part of the shortest or lowest cost path, it must process more packets than the routers that are not on such paths. This router thus must have more computational power, and/or consume more energy. For wireless networks such as sensor networks and mobile ad hoc networks where energy supply may be highly constrained, additional energy consumption by a node can shorten its service life [11].

One solution to address these challenges is to make the routing paths less predictable. Instead of using the same

shortest or lowest cost path for all traffic between a source node and a destination node, it is possible to randomly choose among multiple paths to forward the packets, thus making it more difficult for an attacker to eavesdrop or intercept the entire communication session from a victim. This scheme also spreads the routing load to more routers, thus avoids draining more power from a smaller set of routers. In order to do so, the routers should be configured such that for every destination, instead of having only one next-hop entry per destination in the routing table, there should have multiple entries for each of the alternate paths to that destination.

Some past and recent research conducted in this area include [12], [13], [14], and [15]. In [12] and [13], Luo, Liu and Fang proposed algorithms to find multiple disjoint paths at the expense of additional control messages. They assumed no topology changes during the path finding procedure. In [14], Kuo, Pang and Chan focused on Routing Information Protocol (RIP) for wired networks [16]. They proposed a path randomization algorithm that has small path-similarity sharing the minimal number of common links between source-destination nodes. The most recent work can be found in [15], where Pagan, Hession and Yuan proposed a path randomization algorithm for routing protocols such as OSPF. In their solution, instead of finding a single shortest or lowest cost path between a source node and a destination node, they ran multiple rounds of Dijkstra's algorithm. In each round, all the links belonging to the shortest or lowest cost paths found in the early rounds were excluded from being considered, thus resulted in multiple link-disjoint paths connecting the same source and destination nodes. These alternate paths were then used to create the next-hop entries in the routing table. This solution does not require any changes to LSA message content or format, and is fully compatible with existing OSPF protocol. But because the alternate paths were found running multiple round of Dijkstra's algorithm, these paths may not be optimal in term of their total length or cost. In addition, this solution may not find the disjoint paths in certain topology even if such paths exist [17].

In this paper, we propose a new routing algorithm to address the weaknesses of the solution in [15]. Instead of running multiple rounds of Dijkstra's algorithm, we propose to use both Dijkstra's algorithm and Suurballe's algorithm to find the alternate paths. Suuballe's algorithm is the optimal algorithm to find two link-disjoint paths between two end nodes [18]. It always finds the disjoint paths as long as they exist, regardless of the topology. These alternate paths are then used to populate the routing table. When a data packet arrives at a router, instead of always sending the packet by the same path as in traditional OSPF protocol, the router now randomly chooses one of the next hops for the destination in the routing table to forward the packet. Because each of the packets from a user now take a randomly chosen path at every router, it becomes much more difficult for a hacker to eavesdrop, intercept, or jam the user's entire communication session. This added layer of security is implemented at the routing layer, therefore is fully transparent to the users and network applications. It also better spreads the routing load to more routers, resulting better balance of energy consumption in a senor network or an ad hoc mobile network. The proposed algorithm is fully backward compatible with existing OSPF protocol.

The paper is organized as follows. In Section II, we describe and analyze the proposed algorithm. In Section III, we perform computer simulations to compare the algorithm with Dijkstra's algorithm and the solution in [15]. In Section IV, we conclude the paper.

II. Improved Random Paths Routing (IRPR) Algorithm

Traditional routing protocols such as OSPF use shortest or lowest cost path algorithms to determine a single optimal path connecting each source node and each destination node. One of such algorithms is Dijkstra's algorithm. The input to this algorithm is the complete network topology information, which may be obtained through LSA message exchanges with other routers. The network topology includes all the nodes, the links, and the link costs. The algorithm starts with the source node and checks all its neighboring nodes. The closest neighbor is marked. Then the source node's remaining neighbors, as well as the neighbors of the marked node(s) are checked one by one. The closest node to the source that has not been marked is now marked. This procedure continues until all connected nodes are marked. For a network with $|N|$ nodes and $|L|$ links, the running time can be as low as $O(|L|+ |N|\log|N|)$ [19].

This is a very simple and efficient algorithm to find the shortest or lowest cost paths from the starting node to every other node in a network. The security risk is that an algorithm like this is well-known. An attacker can first obtain the network topology information, and then use the algorithm to compute the exact path by which data packets will be traveling. As discussed in Section I, network traffic can be better protected if the paths taken by the packets are less predictable.

The authors of [15] proposed the Disjoint Path Routing with Random Selections (DPRRS) algorithm. In this solution, Dijkstra's algorithm is run multiple times, one for each alternate path from a source node to a destination node. The first step is to run Dijkstra's algorithm with the complete network topology. When the shortest path is generated and saved, the link cost of all the links in that path is set to infinity. This modification effectively excludes all links of the existing shortest path(s) from the alternate paths being generated later. The Dijkstra's algorithm is run again to find another path which is link disjoint from the first one.

This process repeats until the desired number of disjoint paths is generated, or until a new disjoint path can no longer be found. For a network with $|N|$ nodes and $|L|$ links, the running time is $O(kL + kN\log N)$ for k alternate paths.

However we see two major weaknesses in this algorithm. First, the total length or cost of the k alternate paths generated may not be the minimum. For instance, when k = 2, it has been proven that Suurballe's algorithm always generate the minimum total length or cost whereas the two-step DPRRS algorithm in this case does not. The second weakness is, for certain network topology, the DPRRS algorithm may not find the disjoint paths, even if they exist. For instance, when k = 2, the DPRRS algorithm cannot find the two disjoint paths *s*-*e*-*b*-*d* and *s*-*a*-*f*-*d* between node *s* and node *d* as shown in Fig. 1, while. Suurballe's algorithm can.

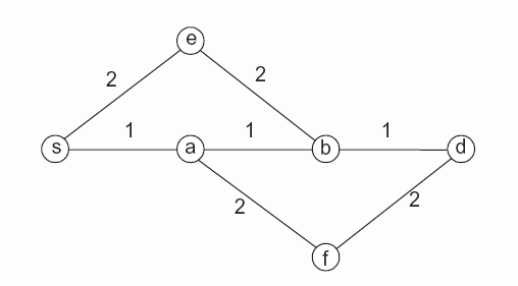

Fig.1. A network with two disjoint paths [17]. The numbers indicate link costs.

The details of our proposed algorithm are given below:

Name:	Improved Random Paths Routing (IRPR) Algorithm
Inputs:	Network $G(N, L)$ where N is the set of nodes and L is the set of links with a cost c_l associated with each link, assuming all links are bidirectional; Node *s*: source node; Node *d*: destination node; *k*: the number of link-disjoint paths connecting *s* and *d* to be found and $k > 1$.
Output:	Up to *k* link-disjoint paths connecting *s* and *d*

Pseudo code:	$p = k$;
	while $p \neq 0$, repeat: //loop begins
	if $p = 1$,
	execute Dijkstra's Algorithm with $G(N, L)$ and *s*, *d* as input;
	if succeeds,
	save the returned path;
	$p := 0$;
	else
	break the loop;
	else
	execute Suurballe's Algorithm with $G(N, L)$ and *s*, *d* as input;
	if succeeds,
	save the two returned paths, and
	set the cost of all the links in the two disjoint paths to infinity;
	$p := p - 2$;
	else
	break the loop;
	//end of loop
	return k–p and all the saved paths;

The major component of this algorithm is Suurballe's algorithm. If the algorithm runs successfully, it executes $\lfloor k/2 \rfloor$ rounds of Suurballe's algorithm and only $k\%2$ round of Dijkstra's algorithm ($\lfloor\ \rfloor$ represents floor operation and % represents modulus operation), and returns k link disjoint paths. Because the running time for Suurballe's algorithm and Dijkstra's algorithm are $O(|N|^2\log|N|)$ and $O(|L|+|N|\log|N|)$ respectively, the total running time for the IRPR algorithm is at most $O(|L| + k|N|^2\log|N|)$, which is polynomial to k, $|L|$ and $|N|$.

Compared to the DPRRS algorithm proposed in [15], our IRPR algorithm has longer running time, due to the complexity of Suurballe's algorithm. However, the IRPR algorithm has a higher probability of finding disjoint paths, and the disjoint paths it finds potentially have shorter total length or lower total cost. These are important improvements because being able to find more disjoint paths leads to more paths for a router to randomly choose from when forwarding user packets, thus improves security and achieves more balanced energy consumption among the routers. Shorter total length or lower total cost of the disjoint paths also have the advantage of requiring less network resource allocation, and users packets may experience shorter delays.

III. Computer Simulations and Analysis

In this section we conduct simulations to compare the performance of our IRPR algorithm with the DPRRS algorithm and the traditional Dijkstra's algorithm. Using LEDA programs, a C++ class library for efficient data types and algorithms, we generated network graphs with network sizes of 10, 20 and 40 nodes and an average nodal degree (i.e., the number of links ending in a node) of 2.8 for testing [15] [20]. We run the IRPR algorithm to generate 2, 3, and 4 link-disjoint paths for every pair of source-destination nodes,

and compare the results with those of DPRRS listed in [15]. Based on the desired number of disjoint paths, we label each variation of the IRPR algorithm as IRPR-2, IRPR-3 and IRPR-4, similar to how the DPRRS algorithm are labeled in [15] as DPRRS-2, DPPRRS-3, and DPRRS-4.

We are most interested in two matrices, the success rate of finding the desired number of disjoint paths, and the total length or cost of the disjoint paths being generated. For the path length or cost, we agree with [15] on that the most significant network resource consumptions and delays occur at the routers; therefore we set all link cost to 1 when we ran the algorithms, which made the path length equal to the hop count. A hop count is the number of routers on a path connecting the source node and the destination node.

For all source-destination node pairs, we ran the three variations of the IRPR algorithm, and compare its success rates and the average hop-counts with those of the DPRRS algorithm and the Dijkstra's algorithm.

We observed that the success rates for the IRPR algorithm and the DPRRS algorithms were nearly the same in all the simulations we performed. This seemingly surprising result may indicate for networks of nodal degree of 2.8 or higher, there exist sufficient numbers of alternate paths between a source node and a destination node; hence both Suurballe's algorithm and two-step Dijkstra's algorithm are equally likely to find these paths.

For the average hop counts of the disjoint paths, we first compare the results of IRPR-2, IRPR-3, IRPR-4 and Dijkstra's algorithm. The comparisons are depicted in Fig 2. As expected, the single shortest paths generated by Dijkstra's algorithm have the lowest average hop counts. For the IRPR algorithm, the more disjoint paths it must generate, the higher the average hop counts become. This is because the additional paths must take longer routes in order to be disjoint from each other. It can also be observed that the more nodes a network has, the longer the paths becomes, simply because larger networks provide more connectivity; therefore there are increased path variations and the average distances between the nodes become larger.

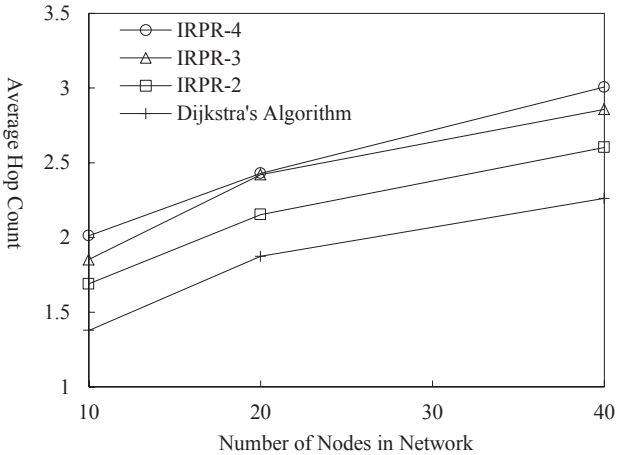

Fig 2. Comparison of the average hop counts using variations of the IRPR algorithm and Dijkstra's Algorithm

Next we compare the average hop counts for the same number of disjoint paths generated by the IRPR algorithm and the DPRRS algorithm. The results are depicted in Fig 3, Fig 4, and Fig 5. It is very clear that the IRPR algorithm consistently and significantly outperforms the DPRRS algorithm for all network sizes and all numbers of disjoint paths generated. To be more specific, for 2 disjoint paths, IRPR outperforms DPRRS by 11.62% to 15.56%; for 3 disjoint paths, IRPR outperforms DPRRS by 14.97% to 19.04%; and for 4 disjoint paths, IRPR outperforms DPRRS by 19.20% to 22.99%. These results match our expectations discussed in Section II. It also appears that the more disjoint paths are to be generated, the more advantages the IRPR algorithm become over DPRRS.

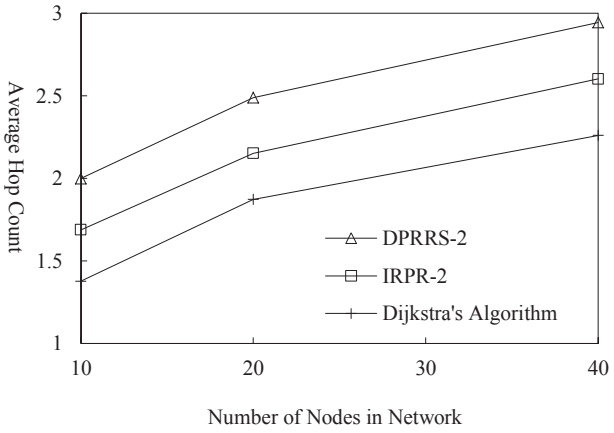

Fig 3. Comparison of the average hop counts of two disjoint paths using the IRPR algorithm and the DPRRS algorithm

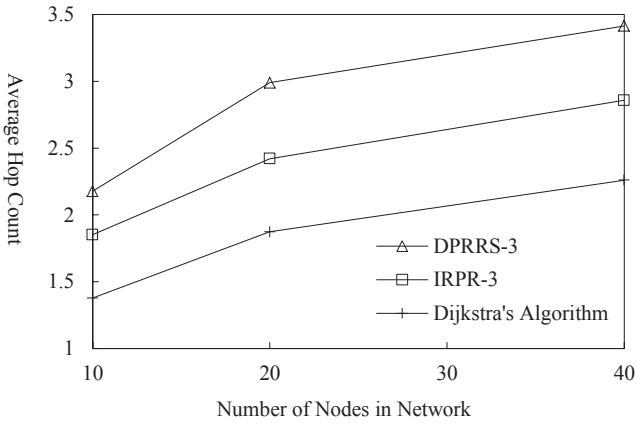

Fig 4. Comparison of the average hop counts of 3 disjoint paths using the IRPR algorithm and the DPRRS algorithm

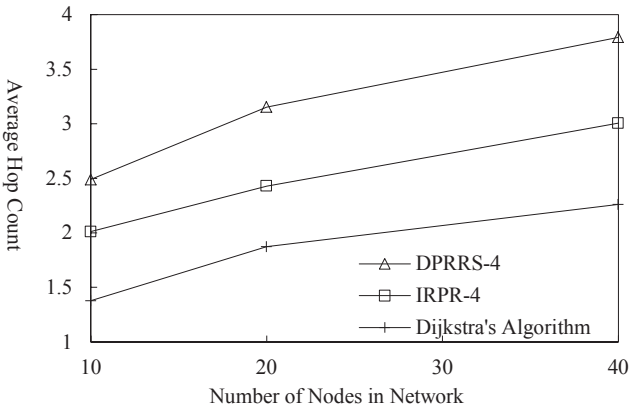

Fig 5. Comparison of the average hop counts of 4 disjoint paths using the IRPR algorithm and the DPRRS algorithm

IV. CONCLUSION

Traditional routing protocols such as OSPF use shortest path or lowest cost routing algorithms to determine the optimal path for a given source node and a destination node, which may lead to various security vulnerabilities, and unbalanced energy consumptions by the routers. In this paper, we proposed a new routing algorithm, IRPR that generates multiple disjoint paths for the routers to randomly select from. The algorithm's complexity is polynomial and can be easily implemented. Computer simulations confirmed that the algorithm consistently and significantly outperformed the existing solutions in term of path hop counts. For future study, we will investigate more advanced solutions, especially when more disjoint paths are desired and when node disjoint paths are desired.

REFERENCES

[1] Douglas Comer, *Internetworking with TCP/IP*, 6th Edition. Pearson.
[2] J. T. Moy, *OSPF: Anatomy of an Internet Routing Protocol*, 1st Edition. Addison-Wesley Professional.
[3] N. Jasika, N. Alispahic, A. Elma, K. Ilvana, "Dijkstra's Shortest Path Algorithm Serial and Parallel Execution Performance Analysis." *Proceedings, IEEE 35th International Convention MIPRO*, 2012. Pages: 1811-1815.
[4] A. Barbir, S. Murphy, and Y. Yang. "Generic Threats to Routing Protocols." *https://www.ietf.org/rfc/rfc4593.txt*
[5] Vetter, Brain, Feiyi Wang, and S. Felix Wu. "An Experimental Study of Insider Attacks for OSPF Routing Protocol," *Proceedings, IEEE International Conference on Network Protocols*, 1997. Pages: 293-300.
[6] Fang Ying-lan, Han Bing, and Li Ye-bai. "Research and Implementation of Key Technology Based on Internet Encryption and Authentication." *Proceedings, IEEE International Conference on Networking and Digital Society*, 2009. Pages: 179-182.
[7] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol," *https://tools.ietf.org/html/rfc5246*.
[8] S. Kent and K. Seo, "Security Architecture for the Internet Protocol," *https://tools.ietf.org/html/rfc4301*.
[9] Kamal, S., and B. Isaac. "Analysis of Network Communication Attacks," Proceedings, *IEEE 5th Student Conference on Research and Development*, 2007. Pages: 1-6.
[10] Charlie Kaufman and Radia Perlman. *Network Security: Private Communication in a Public World*, 2nd Edition. Prentice Hall.
[11] Yi-Bing, Li Hai-Bo, Li Zhong-Cheng, and Dutkiewicz, E. "A New Method of Selecting Stable Paths in Mobile Ad Hoc Networks." *Proceedings, IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, 2006. Volume: 2. Pages: 38 – 45.
[12] Wenjing Lou, and Yuguang Fang. "A Multipath Routing Approach for Secure Data Delivery." *Proceedings, IEEE Military Communications Conference, MILCOM* 2001. Volume: 2. Pages: 1467 – 1473.
[13] Wenjing Lou, Wei Liu, and Yuguang Fang. "Spread: Improving Network Security by Multipath Routing." *Proceedings, IEEE Military Communications Conference, MILCOM* 2003. Volume: 2. Pages: 808 - 813.
[14] Chin-Fu Kuo, Ai-Chun Pang, and Sheng-Kun Chan. "Dynamic Routing with Security Considerations". *IEEE Transactions on Parallel and Distributed Systems*. January 2009. Pages: 48-58.
[15] Mario Pagan, Audrey Hession, and Shengli Yuan. "A security-enhanced routing algorithm with path randomization." *Proceedings, IEEE International Conference on Computing, Networking and Communications* (*ICNC*) 2015. Pages: 1137 – 1141.
[16] G. Melkin. "RIP Version 2." *https://tools.ietf.org/html/rfc2453*
[17] Shengli Yuan and Jason P. Jue, "Dynamic Lightpath Protection in WDM Mesh Networks under Wavelength-Continuity and Risk-Disjoint Constraints," *Computer Networks Journal* (*Elsevier*), vol. 48, no. 2, pp. 91-112, June 2005.
[18] J. W. Suurballe, R.E. Tarjan, "A quick method for finding shortest pairs of disjoint paths." *Networks*. Vol. 14. 1984. Pages: 325-336.
[19] T. Cormen, C. Leiserson, R. Rivest, *Introduction to Algorithms*, McGraw Hill, 1997
[20] Algorithmic Solutions Software GmbH, *http://www.algorithmic-solutions.com/leda/index. htm*