A MULTIPATH ROUTING APPROACH FOR SECURE DATA DELIVERY

Wenjing Lou and Yuguang Fang

Wireless Networks Laboratory (WINET)
Department of Electrical and Computer Engineering
University of Florida
Gainesville, FL 32611

Email: wjlou@ufl.edu , fang@ece.ufl.edu

ABSTRACT

In this paper, we propose a novel approach to enhance data confidentiality when transmitting across the insecure networks. The idea is to take advantage of the distributed nature of networks such as Internet or wireless networks and combine the secret sharing scheme and multipath routing. With a (T,N) secret sharing scheme, the secure message is divided into N shares such that from any T or more shares, we can easily recover the message, while from any T-1 or less shares, it is computationally impossible to recover the message. Then using multipath routing algorithm, the shares are delivered across the network via N different paths, where no T or more paths can share a single node. The destination node reconstructs the original message upon receiving T or more shares. However, any intermediate node does not intercept T shares necessary for the message recovery. In this paper, we present the basic idea, then, we describe a distributed multipath routing algorithm to find the desired N different paths. The algorithm takes path independence, path quantity, as well as path cost into consideration. With comparably low complexity, the algorithm is able to find, between any source-destination pair, sets of node disjoint paths. The algorithm is compared with another disjoint path finding algorithm and the result shows that our algorithm has better performance in terms of number of paths found.

I. INTRODUCTION

In any modern network, there is a need for security. However, the current Internet, without integrating with security mechanisms originally, has a number of security problems and lacks effective protection of confidentiality and integrity of the data transferred over the network below the application layer. The internetworking communication will be exposed to all kinds of attacks in such an open hostile environment. With the emerging of applications such as e-commence/m-commerce, the need for network security services that can provide secure communication in public networks has been more and more significant.

The common approach of providing secure communication across unsecured channel is basically to apply data encryption/decryption on the information transmitted over the networks. Encryption algorithms widely used for this purpose include Data Encryption Standard (DES), which is a 64-bit block cipher and widely used to encrypt the transferred data, and RSA, a popular public-key algorithm widely used for session key exchange and distribution. Both of the algorithms are computationally intensive while RSA is more. In addition, encryption/decryption algorithms need to work in combination with a good authentication mechanism and a good key management scheme. The confidentiality provided by pure data encryption is not absolute in the sense that with today's super computer, it is possible to break any encryption algorithm when enough encrypted information are collected. Most routing algorithms used today favor the stable path, i.e., the session from a source node to a destination node tends to use the same path for quite a long time. If the path is breached in, a large number of messages will be intercepted, which can greatly facilitate the unauthorized decryption of the messages. Till now there is no absolute security in the network.

In this paper, we propose a novel approach to enhance the confidentiality service in public networks. The basic idea is simple: if we purposefully deliver message shares in distributed fashion to minimize the potential captures of message shares, the security can be enhanced. Our approach to achieve this effect relies on the following two principles. First, we exploit the secret sharing principle. A (*T,N*) threshold secret sharing scheme allows us to divide a confidential message into *N* shares and requires the knowledge of at least *T* out of *N* shares to reconstruct the original message. Here the *N* shares is not the simple

This work was supported in part by the National Science Foundation Faculty Early Career Award ANI 0093241.

fragmentation of a message, but a mathematical transformation such that each individual share does bear information about the message, but does not carry any meaningful partial plain text of the original message. Even with infinite computing time and power, if the number of shares available is less than *T*, the cheater can do no better than guessing [1]. Second, we exploit multipath routing. While the current Internet is based on the single shortest path routing, multipath routing has attracted much research attention recently for its potential in aggregating bandwidth, minimizing delay, alleviating network congestion, improving fault tolerance (reliability), etc. [2][3][4][5]. By taking advantage of the connectivity redundancy of the network, multipath routing can give the source node a choice at any given time of multiple paths to a particular destination. Therefore, the shares may be routed to a destination via a number of disjoint multiple paths depending on the level of security level, reliability level, and congestion level. From network point of view, if a whole message follows a single path to its destination, a hacker has a chance to intercept all the information about that message by compromising any one of the intermediate nodes. However, if we apply the (*T,N*) secret sharing scheme to the message, divide it into *N* shares, and send the *N* shares over *N* independent paths, the hacker has to compromise at least *T* different nodes on *T* independent paths to obtain enough information necessary for the message reconstruct (Here two paths are defined as independent if they have only the source node and the destination node in common [6]. They are also called node disjoint paths). This will greatly improve the security of the message transmitted. Moreover, this approach tolerates the disclosure of *T-1* shares. As ensured by the secret sharing scheme, even with super computers, when the number of shares compromised is less then *T*, the hacker cannot recover the message, neither any partial content of the message. What he can do is no better than guessing (the brute-force attack). Therefore, the integration of the secure sharing idea and the multipath routing technique results in a novel approach for enhancing communication confidentiality in public networks.

This paper is organized as follows. In section II, we describe the threshold secret sharing scheme and how it is applied to the message to be transmitted. In section III, we present a distributed multipath routing algorithm to find the desired multiple independent paths. The simulation results about the algorithm are reported in section IV. Conclusions are drawn in the last section.

II. THRESHOLD SECRET SHARING SYSTEM

Secret sharing principle has been well developed as a means for increasing the confidence in the proper functioning of the information based systems. Secret sharing scheme is popularly used in secret key management [1],[8]. In this section, we apply this principle to design a data delivery scheme with enhanced security. Suppose that we have a system secret *K* and we divide it into *N* pieces, S_1, S_2, ..., S_N, called shares or shadows. Each of *N* participants of the system, P_1, P_2, ..., P_N, hold one share of the secret respectively. Any less than *T* participants cannot learn anything about the system secret, while with an effective algorithm, any *T* out of *N* participants can reconstruct the system secret *K*. This is called a *(T,N)* threshold secret sharing scheme [1]. The simplest example of this principle is the well-known two-man control rule that the Unite States enforces for critical military actions. Each of the two men knows a private piece of information, only when combined with the piece known to the other man does it suffice to allow access to a weapons system. However, each piece of the information individually provides its holder no greater chance of access or ability to use the weapon than what an outsider who knows nothing at all about the secret controlling information would be.

Secret sharing schemes consist of two algorithms. The first is called the *dealer*, which generates and distributes shares among the participants. The second is called the *combiner*, which collects shares from the participants and recomputes the secret. It produces the secret *K* from any *T* correct shares. The combiner fails to recompute the secret if the number of the correct shares is less than *T*.

Consider a source node intends to send data to a destination node securely over the distributed insecure networks. In this application, the dealer process is implemented at the source node while the combining is done at the destination node. For illustration purpose, we use the Shamir's Lagrange interpolating polynomial scheme as an example [8]. The dealer obtains the *i*th participant' share by evaluating a polynomial of degree (*T*-1)

$$f(x) = (a_0 + a_1 x + ... + a_{T-1} x^{T-1}) \bmod p$$

at $x=i$:

$$P_i \rightarrow S_i = f(i)$$

Here *p* is a prime number greater than any of the coefficients and must be made available to both dealer and combiner [8]. To save network bandwidth, we can make an improvement on Shamir's scheme by assigning secrets to all coefficients instead of one coefficient commonly used in key management. So the coefficients a_0, a_1, a_2, …, a_{T-1} are all secrets here. Similar to all block cipher programs, the dealer process works on an *L*-bit block basis. The message is first chopped into *L*-bit blocks. A group of *T* blocks, which correspond to the *T* secrets a_0, a_1,

$a_2, \ldots, a_{T-1}$, are sent to the T inputs of the dealer process at one time. The output of the dealer process is a group of N $(L+1)$-bit blocks, where the ith output corresponds to the value of $f(i)$. Each block can form a single IP packet or a number of blocks from the same output port can be concatenated to form a longer IP packet. The decision of all the blocks in one group should be the same. Identification information needs to be added to the packets so that they can be identified at the receiver end. Figure 1 shows the T-input N-output dealer structure -- T L-bit blocks of message goes in one end of the algorithm and N $(L+1)$-bit blocks of secret shares come out the other end.

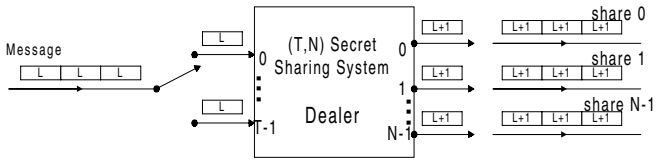


Figure 1 The (T,N) Secret Sharing System – Dealer Structure

At the network layer, the multipath routing protocol will assign each of the N packets in one group a different path to the destination. To maximize the security, the path assignment could be scrambled. Source routing can be used to route the packets via the specified paths across the network.

A buffer at the destination node is needed to temporally store the received packets. Here we assume the capacity of the buffer is not an issue so that re-sequencing of the packets could be done (Further implementation issues such as synchronization will be investigated in the future). For each group of blocks, as long as T blocks of them have been received, the combiner can reconstruct the original blocks by solving a set of linear equations over a finite field. For example, let the received T blocks be $f(i_0)$, $f(i_1), \ldots, f(i_{T-1})$, let

$$A = \begin{bmatrix} a_0 & a_1 & \cdots & a_{T-1} \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ i_0 & i_1 & \cdots & i_{T-1} \\ \vdots & \vdots & & \vdots \\ i_0^{T-1} & i_1^{T-1} & \cdots & i_{T-1}^{T-1} \end{bmatrix}$$

$$F = \begin{bmatrix} f(i_0) & f(i_1) & \cdots & f(i_{T-1}) \end{bmatrix}$$

then, the original blocks, $a_0, a_1, a_2, \ldots, a_{T-1}$, can be determined by the linear equations in matrix form

$$AB = F$$

This equation has a unique solution over the finite field $GF(p)$. This can also be explained via Lagrange interpolating polynomial approach. Given T points in the two-dimensional plane, there is a unique solution of a polynomial of degree $(T-1)$ which transverses these T points. Therefore, the reconstruction always succeeds if the combiner has at least T different shares, but fails if the number of shares is less than T [8]. The reconstructed blocks should be concatenated and passed to the higher layer.

III. MULTIPATH ROUTING

In order to distribute shares to multiple paths, we need to design efficient algorithms for finding multiple paths with minimum overlaps. The routing protocols used in today's Internet are destination-based single shortest path algorithms. In between a single source-destination pair, normally the same shortest path will be used. The existing Internet routing protocols provide very limited multiple paths routing capability. Only when there exist multiple paths and are of the same (or varies within certain range of) cost, the packets will be forwarded via multiple paths to the same destination, and this is mainly done for load balancing, congestion control or reliability.

How to find the multiple paths with the desired property is the key implementation issue in our approach. The solution lies on the so-called multipath routing algorithms and protocols. Notice that in the current Internet implementation, a router will have at least two interfaces/connections and usually it has more. Multipath routing aims to take advantage of the connectivity redundancies of the underlying physical networks by providing multiple paths between source-destination pairs [2]. A closely related topic to multipath routing is the long studied k-shortest path problem. Generally speaking, the k-shortest paths problem is to list the k paths connecting a given source-destination pair in the digraph with minimum total cost [9][10]. However, the paths found by the k-shortest path algorithms tend to share many links. The lack of independence limits the effectiveness of providing the security for the message shares in our proposed scheme. Algorithms that overcome the problem of path independence are ones that find disjoint paths between nodes. Ogier, et al, proposed a distributed algorithm for finding two disjoint paths of minimum total cost from each node to a destination [11]. Sidhu, et al, proposed a distributed algorithm that finds multiple disjoint paths to a destination [12]. In our case, path quantity and path

independence is the major consideration. A multipath routing algorithm is needed to find maximal number of paths between a single source-destination pair, which also maximizes the independence of the paths. In the next section, we propose a new multipath routing algorithm, which is specifically efficient for our multipath secure routing scheme.

A. Multipath Routing Algorithm

In this section, we describe a distributed multipath routing algorithm to find node disjoint paths in a network.

A.1 Notations

To facilitate the presentation of our algorithm, we first give some notation. A network is modeled as an undirected graph, *G=(V,E)*, with a finite set of nodes, *V*, and a finite set of links, *E*. A link in *E*, connecting a pair of nodes, *x* and *y*, in *V*, is denoted by *(x,y).* The cost associated with the link is represented by *c(x,y).* We assume that the cost *c(x,y)* is a positive number which is same in both directions.

There is an arbitrary node*, t*, in *V,* called the destination node. A path *p* from *s* to *t* is a sequence *(s,i,j,…,k,t)* of nodes of *G* such that, each link of *(s,i),(i,j),…,(k,t)* is in *E*. The cost of the path, denoted as *c(p)*, is defined as the sum of its link costs. The path identifier of a path *p=(s,x,…,y,t)* is defined as *pid(p)=y*. A shortest path tree, *SPT*, rooted at destination *t,* is a subgraph of *G* such that the length of every path from each node *x* to *t* is minimized. Links in the tree are called *tree* links while the links not in the tree are called *nontree* links. Let *SPT(x,t)* denote the unique path from *x* to *t* in *SPT*. If *y* is on the path *SPT(x,t), y* is defined to be *downtree* of *x*, and *x* is *uptree* of *y*. For a node *x*, if there exist a link *(x,y), y* is called a *neighbor* of *x*. Let *nbr(x)* denote the set of neighbors of *x*. If *y∈nbr(x)* and *y* is uptree of *x*, *y* is a *uptree neighbor* of *x*. Let *upnbr(x)* denote the set of uptree neighbors of *x*. If *y∈nbr(x)* and *y* is downtree of *x*, *y* is *parent* of *x*. Let *parent(x)* denote the parent of *x*. If *y∈nbr(x)* and *(x,y)* is a nontree link, *y* is called a *horizontal* neighbor of *x*.

A constant parameter *COST_BOUND* is set to avoid too high cost path. A *(s,t*) path with cost greater than *COST_BOUND* c(SPT(s,t))* will not be accepted.

A.2 Distributed Algorithm

The distributed multipath routing algorithm described here is able to find for each node *x* a set of disjoint path to destination node *t*. Initially, each node *x* in graph *G* has an empty path set *Px*, which is used to store the disjoint paths from *x* to *t*, and an empty path set *Cx*, which is used to store candidate paths also from *x* to *t* but not disjoint from paths in *Px*. The elements in *Px* and *Cx* have the same structure *{pid,cst,path*}, where *path* is the node sequence of the path; *pid* is the path identifier; and *cst* is the cost of the path. Here we define *Px* and *Cx* as ordered sets, with all the elements ordered by *cst* in increasing order. The order of the elements is always kept when elements are added or removed.

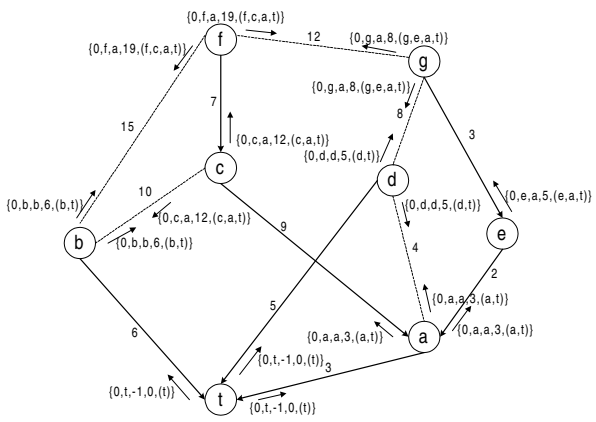
We assume that each node has the knowledge of its parent, uptree neighbors, horizontal neighbors and the costs to them. We also assume that no topological changes occur during the path finding procedure.

Two types of messages are used to advertise the path information. The general form of the message is *msg{mtype,nid,pid,cst,path},* where *mtype* is the message type, either 0 or 1; *nid* is the identifier of the node sending the message; *path* is node sequence of the path to be advertised; *pid* is the path identifier of the path; *cst* is the cost of the path. The two types of message are of same structure but differ in the rule the messages are propagated.

The first phase of the algorithm is the propagation of the type-0 messages. The destination node *t* initializing the algorithm by sending *msg{0,t,∅,0,(t)}* to each of its neighbors. Each intermediate node *x*, upon receiving a type-0 message *msg{0,nid,pid,cst,path},* learns about a new path *q=(x)+path*. If the message is from its parent, *q* is the shortest path *SPT(x,t)* and is added to path set *Px*. Meanwhile, node *x* further propagates the type-0 message by sending *msg{0,x,(pid==∅)?x:pid,cst+c(x,parent(x)),q}* messages to all its uptree neighbors and horizontal neighbors. If the message is from horizontal neighbor, *q* is an alternative path to destination *t* and thus included into candidate path set *Cx*. The type-0 messages received from the horizontal neighbors are not propagated further. The propagation of type-0 messages terminates at the leaf nodes of *SPT* after the leaf nodes have sent and received type-0 message on their horizontal neighbors.

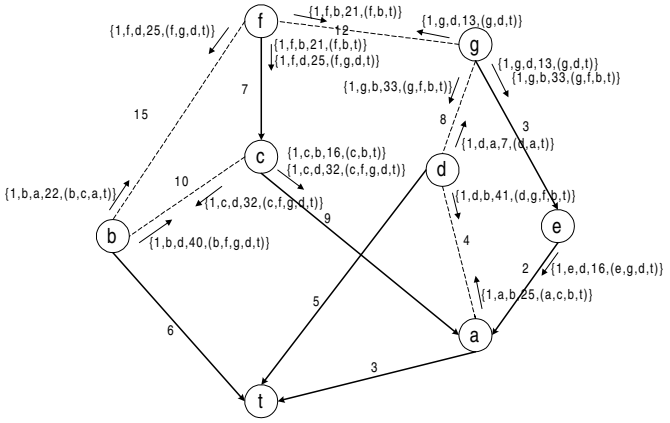
Once having received the type-0 messages from its parent and all its horizontal neighbor(s), node *x* checks its candidate path set *Cx* for disjoint paths. For each path *p* in *Cx*, node *x* includes *p* into *Px* if *p* is disjoint from any other path in *Px*. When *p* is included into *Px*, it is removed from *Cx*.

The second phase of the algorithm is to find more disjoint paths by exchange type-1 messages. The propagation of type-1 messages is initiated independently by each node at



At the end of phase 1, Px at each node:
Pa={a,3,(a,t)},{d,9,(a,d,t)}
Pb={b,6,(b,t)},{a,22,(b,c,a,t)}
Pc={a,12,(c,a,t)},{b,16,(c,b,t)}
Pd={d,5,(d,t)},{a,7,(d,a,t)}
Pe={a,5,(e,a,t)}
Pf={a,19,(f,c,a,t)},{b,21,(f,b,t)}
Pg={a,8,(g,e,a,t)},{d,13,(g,d,t)}

(a) type-0 message exchange and paths found in phase 1



At the end of phase 2, Px at each node:
Pa={a,3,(a,t)},{d,9,(a,d,t)},*{b,25,(a,c,b,t)}*
Pb={b,6,(b,t)},{a,22,(b,c,a,t)},*{d,40,(b,f,g,d,t)}*
Pc={a,12,(c,a,t)},{b,16,(c,b,t)},*{d,32,(c,f,g,d,t)}*
Pd={d,5,(d,t)},{a,7,(d,a,t)},*{b,41,(d,g,f,b,t)}*
Pe={a,5,(e,a,t)},*{d,16,(e,g,d,t)}*
Pf={a,19,(f,c,a,t)},{b,21,(f,b,t)},*{d,25,(f,g,d,t)}*
Pg={a,8,(g,e,a,t)},{d,13,(g,d,t)},*{b,33,(g,f,b,t)}*

(b) type-1 message exchange and paths found in phase 2

Figure 2 Illustration of the Algorithm

which alternative disjoint path(s) is found at the end of phase 1 (*SPT(x,t)* is not considered as an alternative path). For each alternative path *p*, node *x* form a type-1 message *msg{1,x,p.pid,p.cst,p)* and send it to all its type-1 eligible neighbors. A node *v* is defined as a type-1 eligible neighbor of *x* for path *p* if it satisfies the following conditions: (1) $v \in nbr(x)$; (2) $v \notin p$; (3) $v \notin upnbr(x)$.

Upon receiving a type-1 message *msg{1,nid,pid,cst,path},* node *x* learns about a new path *q=(x)+path*. Due to the distribution of the computation, node *x* may receive type-1 messages before its phase 1's path finding procedure finishes. If this is the case, node *x* will buffer the received type-1 messages until the phase 1 processing finishes. Node *x* includes *q* in *Px* if *q* is disjoint from any other path in *Px* of lower cost. If *x* includes *q* in *Px*, it excludes from *Px* every path that intersects with *q.* The path(s) removed from *Px,* and *q*, when not included in *Px*, are considered to be included in *Cx*. Node *x* includes path *q=(x,v,…,y,t)* in *Cx* if *q* is the cheapest path through *v* or it is the cheapest path with *pid=y* in *Cx*. Node *x* checks *Cx* for possible disjoint path(s) whenever there is a path removed from *Px*.

Whenever a new path *p* is added to *Px*, node *x* forms a type-1 message *msg{1,x,p.pid,p.cst,p.path}* and send it to all its type-1 eligible neighbors.

The propagation of type-1 messages terminates when no disjoint path is added to any path set *Px* or no node can send a type-1 message to any of its neighbors. At this time, each node has found a set of disjoint paths to the destination node *t*.

A.3 Example

Figure 2 shows an example to illustrate how the algorithm works. Graph *G* represents an arbitrary network of 8 nodes and 12 links. The shortest path tree *SPT,* rooted at an arbitrary destination node *t*, superimposes on *G*. Directed solid lines mark the tree links with arrows pointing from uptree to downdree. The nontree links are marked by dotted lines.

Figure 2(a) illustrates the exchange of type-0 messages. It is observed that by the end of phase 1, each node finds its shortest path to the destination node, which consists of only tree links. Each node also finds alternative disjoint path(s), which contains just one nontree link. Figure 2(b) illustrates the exchange of the type-1 messages. By exchanging type-1 messages, each node expands its

disjoint path set by adding paths consisting of more than one nontree links. The disjoint paths found for each node at the end of phase 1 and phase 2 are listed respectively.

IV. SIMULATION RESULTS AND DISCUSSION

We conduct a simulation study based on our algorithm and compare it with the algorithm proposed in [12]. The simulation is performed on 8 arbitrarily chosen networks. The first network is the one shown as the example in this paper. The rest 7 are all 20-node networks but with different parameters, as shown in the table. The notation *(M,D,d)* represents a network of *M* nodes with diameter *D* and average node degree *d*. All the links in the networks are assigned arbitrary link cost. The algorithm in [12] is referred to as A while ours as B. Notation U refers to the situation where our algorithm is performed on the same network but all the links are considered as of unit cost (i.e., we use the hop count as the cost metric).

It is observed from the simulation results that our algorithm is able to find more disjoint paths compared with algorithm A, which is most desirable in our approach. However, our algorithm reaches this by exchanging more messages, as we expect. Column 5 shows the average number of messages required per path. Although our algorithm needs slightly more messages per path, the value in such range of 2,3 or 4 messages per path is still quite acceptable.

Network (M,D,d)		Total # Path	Total # Msg	Msg / Path	Path/ Node Pair
(8,3,3)	*A*	138	180	1.30	2.46
	B	151	259	1.72	2.70
	U	149	251	1.68	2.66
(20,7,2)	*A*	422	448	1.06	1.11
	B	422	448	1.06	1.11
	U	422	448	1.06	1.11
(20,7,3)	*A*	775	1184	1.53	2.04
	B	788	1327	1.68	2.07
	U	845	1398	1.65	2.22
(20,6,3)	*A*	730	1117	1.53	1.92
	B	738	1301	1.76	1.94
	U	794	1401	1.76	2.09
(20,5.3)	*A*	714	1115	1.56	1.88
	B	747	1316	1.76	1.97
	U	783	1370	1.75	2.06
(20,5,4)	*A*	926	1788	1.93	2.44
	B	1101	2774	2.52	2.90
	U	1124	2850	2.54	2.96
(20,4,4)	*A*	1003	1799	1.79	2.64
	B	1130	3063	2.71	2.97
	U	1173	3041	2.59	3.09
(20,4,5)	*A*	1091	2390	2.19	2.87
	B	1288	4661	3.62	3.39
	U	1416	4922	3.48	3.73

Table 1 Simulation Results

In general, the number of disjoint paths exist in the network is solely dependent on the network topology, while the number of disjoint paths found by the algorithm also depends on the link cost assignment. Unreasonable link cost assignment may decrease the number of paths found. It is observed from the simulation results that the algorithm tends to find more disjoint paths when we treat all the links as of unit cost, corresponding to the cost structure independent of traffic. This is a useful approach when we need as many as possible disjoint paths while the cost of the link is not a major concern. For example, if we need to deliver a message to a destination with high security, we may tolerate certain delay, therefore, we would use as many disjoint paths as possible to decrease the probability of interception and recovery of the original message to be secured.

Although the *N* completely disjoint path solution is most desirable in this approach, due to the restriction of the network topology, it may not be available at all circumstances. It would be wise to tune the value of parameter *N* based on the number of disjoint paths found. In addition, *N* different paths with partial node/link sharing may be used so long as there exists no node/link which is shared by *T* or more than *T* paths. This also guarantees that no intruder can recover the message by comprising any one of the intermediate nodes. The detailed algorithm to find paths with dependence degree d, d=1,2,…, is still under research now, where the dependence degree of path in a path set is defined as the maximum number of paths in the set sharing a single node.

As a final remark about our scheme, we notice that *T* can be tuned according to the security levels in combination of value *N*. Messages with lower security requirements can use smaller thresholds (*T,N*), while messages with higher security demand may choose use larger values (*T,N*). If a message with high security request does not find the required number of paths, it either delays the data delivery or uses better encryption algorithms. More relevant issues will be discussed elsewhere.

V. CONCLUSIONS

Data confidentiality service is an important issue in network security. Data encryption/decryption is the common approach used in practice. In this paper, we propose a novel approach to enhance data confidentiality

service in the network. The idea is to integrate the secret sharing scheme and multipath routing. We first describe the secret sharing scheme and how it is applied to the secure message to be transmitted, then we develop a distributed multipath routing algorithm to find the desired multiple paths. The algorithm takes path independence, path quantity, as well as path cost into consideration. The simulation shows that with comparably low complexity, the proposed algorithm is able to find, for each source-destination pair in the network, a set of disjoint paths. The proposed algorithm is compared with another disjoint path finding algorithm and the result shows that our algorithm has better performance in terms of number of paths found.

REFERENCE

[1] G. J. Simmons, "An introduction to shared secret and/or shared control schemes and the application", in *Contemporary Cryptology: The Science of Information Integrity*, IEEE Press, 1992, pp.441-497

[2] J. Chen, "New approaches to routing for large-scale data networks", Ph.D dissertation, Rice university, 1999

[3] N. Taft-Plotkin, B. Bellur, and R. Ogier, "Quality-of-service routing using maximally disjoint paths", *1999 Seventh International Workshop on Quality of Service,* 1999

[4] D. Sidhu, S. Abdallah, R. Nair, "Congestion control in high speed networks via alternative path routing", *Journal of High Speed Networks*, 2(1992),129-144

[5] A. Banerjea, "On the use of dispersity routing for fault tolerant realtime channels", *European Transactions on Telecommunications*, vol.8, No. 4, pp.393-407, July/August 1997

[6] B. Bollobas, *Graph Theory*, Springer-Verlag, 1979

[7] A. Shamir, "How to share a secret", *Communications of the ACM*, 22(11):612-613, Nov 1979

[8] B. Schneier, *Applied Cryptography*, 2nd edition, chapter 23, John Wiley & Sons, 1996

[9] J. Y. Yen, "Finding the k shortest loopless paths in a network", *Management Science*, 17(1971)712-716

[10] D. Eppstein, "Finding the k shortest paths", *SIAM J. Computing* 28(2):652-673, 1999

[11] R. Ogier, V. Rutenburg, N. Shacham, "Distributed algorithms for computing shortest pairs of disjoint paths", *IEEE trans. Information Theory*, vol.39, No.2, Mar 1993

[12] D. Sidhu, R. Nair, S. Abdallah, "Finding disjoint paths in networks", *Proc. of ACM SIGCOMM*, 1991