Deteksi dan Pencegahan Serangan *Wormhole* pada Protokol *Routing* AOMDV Menggunakan Gabungan Metode Delphi dan RTT-TC pada Jaringan MANET

(*Detection and Prevention of Wormhole Attacks on the AOMDV Routing Protocol Using Delphi and RTT-TC Method on MANET*)

Nini Kurnia Safitri, Andy Hidayat Jatmika, Moh. Ali Albar
Program Studi Teknik Informatika, Fakultas Teknik, Universitas Mataram
Jl. Majapahit 62, Mataram, Lombok NTB, INDONESIA
Email: ninikurniasafitri@gmail.com, [andy**,** mohalialbar]@unram.ac.id

Penulis korespondensi

Abstract - Mobile Ad Hoc Network (MANET) is a network that doesn't have a fixed infrastructure where there is a set of nodes in it. Which the MANET was to be vulnerable to attacks that can interfere with the process of communication and data transmission on the network. One type of attack that can attack the MANET is a wormhole attack. The wormhole is an attack on a network where there are two wormhole nodes that are interconnected using a wormhole link to attack network traffic or discard data packets it receives. To prevent wormhole attacks, the method that can be used is the Delay Per Hop Indicator (Delphi) method and the Round Trip Time and Topological Comparison (RTT-TC) method. The Delphi and RTT-TC methods have their advantages and disadvantages. In this study, it was proposed to combine the Delphi and RTT-TC methods to cover the shortcomings of each method. Based on the results, it can be concluded that the merging of the Delphi and RTT-TC methods can detect and prevent wormhole attacks and improve poor network quality caused by wormhole attacks.

***Key words*: MANET, *Wormhole attack*, Delphi, RTT-TC.**

I. PENDAHULUAN

Mobile Ad Hoc Network (MANET) adalah jaringan yang terdiri dari beberapa node independen yang mana jaringan ini dapat dibentuk kapan saja dan dimana saja [1]. *Node-node* pada jaringan MANET melakukan proses komunikasi dan transmisi data dengan adanya suatu protokol *routing*. Ada 3 jenis tipe protokol *routing* yakni protokol *routing* reaktif, proaktif, dan *hybrid.* Protokol *routing* reaktif hanya akan melakukan pencarian rute apabila akan dilakukan proses pengiriman data ke suatu tujuan dengan cara mem-*broadcast* sebuah pesan. Pada penelitian ini, protokol *routing* yang akan diteliti adalah protokol *routing* reaktif, yakni *Ad Hoc On Demand Multipath Distance Vector* (AOMDV). Protokol *routing* AOMDV memiliki dua fitur, yakni *route discovery* dan *route maintenance.* Proses *route discovery* merupakan proses pencarian rute dari *node* sumber ke *node* tujuan dengan cara menyebarkan paket *route request* (RREQ) ke tetangganya. Sedangkan *route maintenance* merupakan proses pemeliharaan rute, dimana suatu *node* kirim paket *route error* (RERR) ke *node* sumber kalau terjadi kerusakan rute.

Wormhole merupakan suatu serangan pada jaringan dimana terdapat dua atau lebih *wormhole node* yang saling terhubung dan bekerja sama untuk menyerang lalu lintas jaringan ataupun membuang paket data yang diterimanya. Setiap *wormhole node* akan saling terhubung menggunakan *wormhole link* [2]. Proses *route discovery* pada AOMDV cukup rentan terhadap serangan *wormhole,* khususnya pada proses *broadcast* paket RREQ. Ketika proses *broadcast* paket RREQ berlangsung, *wormhole node* yang mengetahui adanya paket RREQ akan merespon paket RREQ dan mengklaim bahwa *wormhole node* tersebut punya rute paling pendek dan terbaru menuju *node* tujuan. Sehingga, rute yang terpilih untuk mengirim paket data adalah rute yang memiliki *wormhole node.* Hal ini akan membuat lalu lintas jaringan menjadi terganggu dan *wormhole node* dapat membuang paket data yang diterimanya.

Wormhole link memiliki ukuran yang sangat panjang, namun *wormhole node* dapat mengklaim bahwa dengan melewati *wormhole link* tersebut maka rute yang ditempuh untuk sampai ke tujuan akan lebih cepat. Sehingga untuk mendeteksi serangan *wormhole,* dibutuhkan metode yang dapat mendeteksi adanya *wormhole link* tersebut. Metode yang dapat digunakan adalah metode *Delay Per Hop Indicator* (Delphi) dan metode *Round Trip Time and Topological Comparison* (RTT-TC). Metode Delphi mendeteksi serangan *wormhole* dengan menghitung *delay* di setiap *hop* dari *node* sumber ke *node* tujuan. Ketika jaringan dalam kondisi normal, *delay* antar *hop* sama di sepanjang rute. Tetapi, pada saat terkena serangan *wormhole*, maka nilai *delay* akan menjadi tinggi karena adanya *wormhole link* yang menghubungkan *wormhole node* [3]. Sedangkan pada metode RTT-TC, serangan

wormhole dideteksi dengan menghitung nilai *round trip time* dan *topological comparison*. *Topological comparison* berfungsi untuk mendeteksi apakah terdapat *wormhole link* disuatu rute atau tidak pada suatu jaringan [4].

Metode Delphi dan RTT-TC memiliki kelebihan dan kekurangannya masing-masing. Kelebihan metode Delphi adalah dapat mendeteksi *wormhole* dengan menghitung *delay per hop* setiap *node,* namun tidak dapat menandai lokasi *wormhole* [5]. Sedangkan, kelebihan metode RTT-TC adalah dapat menandai lokasi *wormhole* dan menyimpan *wormhole* tersebut pada *detected list* (DET *list*) *node* sumber, namun penggunaan RTT dalam mendeteksi *wormhole* kurang *reliable* [4], sehingga pada penelitian ini diusulkan penggabungan metode Delphi dan RTT-TC untuk meningkatkan kinerja metode dalam mendeteksi dan mencegah serangan *wormhole*.

Pada penelitian ini akan dilakukan pendeteksian dan pencegahan serangan *wormhole* pada protokol *routing* AOMDV menggunakan metode Delphi, RTT-TC, serta gabungan metode Delphi dan RTT-TC.

II. Penelitian Terkait

Pada penelitian yang berjudul *Detection and Prevention of Wormhole Attack on AOMDV Routing Protocol using Hop-count and Communication range in WSN,* dilakukan pengujian untuk mengetahui kinerja metode *Hop-count* dalam mendeteksi serangan *wormhole* pada protokol *routing* AOMDV. Pengujian dilakukan dengan menggunakan jumlah *node* yang beragam, yakni 30, 50, dan 70 *node* dengan jumlah *wormhole node* yang digunakan adalah 2 *node*. Untuk mengetahui perbandingan kinerja metode *Hop-count* pada jaringan, maka digunakan parameter uji *packet delivery ratio, throughput,* dan jumlah *packet drops*. Hasil pengukuran kinerja metode *Hop-count* menggunakan parameter uji *packet delivery ratio* berubah-ubah namun cenderung meningkat seiring dengan peningkatan jumlah *node,* yakni 98,68%, 92,86%, dan 99,58%. Kemudian nilai *throughput* semakin meningkat dengan bertambahnya jumlah *node,* yakni 20,48 Kbps, 30,72 Kbps, dan 38,912 Kbps. Jumlah paket yang di-*drop* juga berubah-ubah seiring dengan peningkatan jumlah *node*. Pada saat jumlah *node* 30, jumlah paket *drop* adalah 2 paket, pada saat jumlah *node* 50, jumlah paket *drop* adalah 19 paket, sedangkan pada saat jumlah *node* 70, jumlah paket *drop* adalah 2 paket. Berdasarkan hasil pengujian tersebut, dapat disimpulkan bahwa metode *Hop-count* dapat mendeteksi serangan *wormhole* dengan baik [6].

Pada penelitian [7] dilakukan pencegahan serangan *wormhole* pada protokol *routing* AODV menggunakan metode Delphi. Dari hasil simulasi, dapat diketahui bahwa pada saat pencegahan serangan *wormhole* nilai *throughput* adalah 33% dan nilai *delay* 37%. Sedangkan jumlah *packet loss ratio* adalah 31%, hal ini disebabkan karena adanya paket *drop*. Dari hasil simulai dapat disimpulkan bahwa metode Delphi dapat mendeteksi dan mencegah serangan *wormhole*. Metode ini dapat membantu mengurangi *packet loss* dan meningkatkan nilai *throughput* [7].

Pada penelitian [4] dilakukan pengujian terhadap protokol AODV dengan parameter uji *detection rate* dan *accuracy of alarm*. Dari hasil simulasi dapat diketahui bahwa, tingkat *detection rate* mencapai 98% pada saat penggunaan metode RTT dan mencapai 100% pada saat penambahan metode *topological comparison*. Sedangkan, tingkat *accuracy of alarm* mencapai 88% pada saat penggunaan metode RTT dan meningkat menjadi 95% dengan penambahan metode *topological comparison*. Sehingga, dapat disimpulkan bahwa penggunaan metode RTT-TC untuk mendeteksi dan mencegah serangan *wormhole* sangat baik, hal ini dapat diketahui berdasarkan parameter uji yang digunakan yakni *detection rate* dan *accuracy of alarm* memberikan nilai yang cukup tinggi [4].

Pada penelitian [8] dilakukan evaluasi terhadap performansi protokol *routing* AODV dan DSR pada saat terdapat serangan *wormhole* dan pada saat tidak ada serangan *wormhole*. Dari hasil simulasi protokol *routing* AODV dan DSR pada saat tidak ada *wormhole,* dapat diketahui bahwa nilai PDR AODV dan DSR sama-sama meningkat seiring dengan peningkatan jumlah *node,* nilai PDR AODV meningkat dari 0,50% hingga 0,96% dan nilai PDR DSR meningkat dari 0,52% hingga 1%. Nilai *end-to-end-delay* AODV dan DSR sama-sama menurun seiring dengan peningkatan jumlah *node,* nilai *average end-to-end delay* AODV menurun dari 200 sec hingga 10 sec dan nilai *average end-to-end delay* DSR menurun dari 1600 sec hingga 10 sec. Sedangkan, AODV dan DSR memiliki nilai *throughput* yang sama dan sama-sama meningkat, yakni dari 8 kbps hingga 120 kbps. Kemudian, dari hasil simulasi protokol *routing* AODV dan DSR pada saat terdapat serangan *wormhole,* dapat diketahui bahwa nilai PDR AODV dan DSR menurun seiring dengan peningkatan jumlah *node,* nilai PDR AODV menurun dari 1% hingga 0,1% dan nilai PDR DSR menurun dari 0,96% hingga 0,21%. Nilai *end-to-end-delay* AODV meningkat dari 0 sec hingga 9 sec dan nilai *average end-to-end delay* DSR meningkat dari 0 sec hingga 30 sec. Sedangkan, nilai *throughput* AODV menurun dari 0,4 kbps hingga 0 kbps dan nilai *throughput* DSR menurun dari 0,6 kbps hingga 0 kbps. Dari hasil simulasi, dapat disimpulkan bahwa performansi protokol *routing* AODV menunjukkan hasil lebih baik daripada DSR [8].

III. Metode Penelitian

Tahap pertama yang harus dilakukan pada penelitian ini yaitu melakukan studi literatur dengan mengumpulkan referensi-referensi yang diperlukan, seperti buku, jurnal ataupun *website* yang berkaitan dengan topik pada penelitian yang diusulkan. Setelah itu dilakukan tahap kedua, yakni mempersiapkan *hardware* dan *software* yang digunakan untuk melakukan penelitian. Tahap selanjutnya adalah membuat *script* serangan *wormhole* untuk mengetahui efek serangan *wormhole* terhadap kinerja protokol *routing* AOMDV. *Script* simulasi dibuat untuk mengatur jumlah *node,* jumlah *wormhole node,* luas area simulasi, dan waktu simulasi yang digunakan. Jika pembuatan *script* sudah selesai, maka simulasi akan

dijalankan dan diperoleh *output* berupa *file trace*. Proses simulasi dilakukan selama lima kali untuk mengetahui pengaruh serangan *wormhole* terhadap perubahan kinerja protokol *routing* AOMDV. Dari simulasi yang telah dilakukan, maka akan diperoleh *output* yang akan di-*filter* untuk mengetahui kinerja protokol *routing* AOMDV standar maupun yang sudah dimodifikasi berdasarkan parameter uji *average end-to-end delay, packet delivery ratio* (PDR), dan *throughput* menggunakan *tool* AWK *script*. Kemudian dibuat grafik dari hasil *filtering file trace* dan akan dianalisis masing-masing grafiknya dan diperoleh suatu kesimpulan. Setelah itu dilakukan tahap akhir penelitian yakni pembuatan laporan penelitian.

A. *Mekanisme Protokol Routing AOMDV*

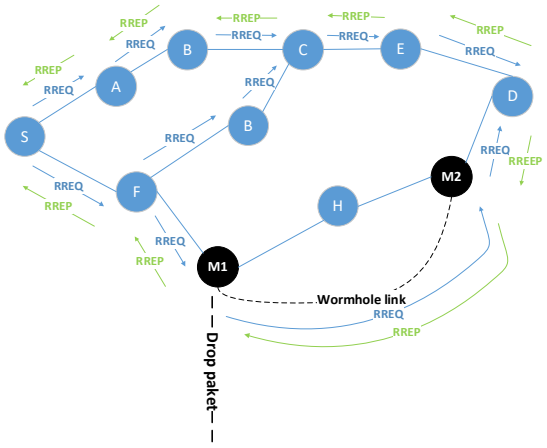
AOMDV merupakan salah satu protokol *routing* yang menerapkan konsep *multiple path* yang mendukung *multipath routing* dan merupakan bentuk pengembangan dari AODV, dimana rute yang diperoleh pada saat *route discovery* yakni lebih dari satu rute dan dapat digunakan sebagai rute cadangan. Protokol *routing* AOMDV memiliki beberapa karakteristik yang sama seperti protokol *routing* AODV. Protokol *routing* AOMDV berbasis vektor dan menggunakan pendekatan *hop by hop* seperti protokol *routing* AODV. Seperti AODV, protokol *routing* AOMDV juga memiliki dua fitur, yakni *route discovery* dan *route maintenance*. Sementara itu, perbedaan utama antara protokol *routing* AODV dan AOMDV adalah jumlah rute yang ditemukan di setiap pencarian rutenya. Pada saat rute sudah ditemukan, AODV hanya memperoleh satu rute, sedangkan AOMDV dapat memperoleh lebih dari satu rute.

Route maintenance merupakan proses pemeliharaan rute. Suatu *node* akan mengirimkan paket RERR ke *node* sumber secara *unicast* jika *link* yang menghubungkannya dengan *node* tetangganya terputus. Kemudian tabel *routing node* sumber akan di periksa, apakah terdapat rute cadangan ke *node* tujuan yang sama. Jika ada, maka rute dengan jumlah *hop* terkecil akan dipilih dan paket data dapat langsung dikirimkan. Namun, jika tidak terdapat rute cadangan, maka proses *route discovery* akan diulangi [9].

B. *Mekanisme Serangan Wormhole*

Serangan *wormhole* merupakan salah satu jenis serangan pada jaringan MANET, dimana terdapat dua *wormhole node* atau lebih yang terhubung menggunakan sebuah *link* yang disebut *wormhole link*. *Wormhole node* bekerja sama menciptakan ilusi bahwa *wormhole node* tersebut nampak menjadi *node* normal, padahal *node* tersebut merupakan *node* penyerang. *Wormhole* membuat dua atau lebih *wormhole node* untuk mengendalikan lalu lintas proses *routing*. *Wormhole link* yang digunakan oleh *wormhole node* bisa menjadi mekanisme komunikasi yang digunakan oleh jaringan.

Untuk meluncurkan serangan *wormhole,* sebuah *wormhole* menghubungkan dua atau lebih *wormhole node* secara berpasang-pasangan di jaringan menggunakan *wormhole link*. Dalam proses pencarian rute, *wormhole node* menerima paket di satu lokasi dan mengirimkannya ke *wormhole node* lainnya melalui *wormhole link* dan akan diteruskan ke *node* normal yang menjadi tetangganya hingga sampai ke *node* tujuan. Jika rute sudah ditemukan, maka paket data akan dikirimkan melalui rute tersebut dan *wormhole node* pada rute tersebut akan menjatuhkan paket data yang diterimanya, sehingga menyebabkan terjadinya gangguan jaringan [10].



Gambar. 1. Ilustrasi serangan *wormhole*

Gambar 1 menunjukkan proses pencarian rute dari *node* S menuju *node* D. *Node* S akan melakukan *broadcast* paket RREQ ke *node-node* tetangganya untuk menemukan rute menuju *node* D. Pada saat proses *broadcast, node* M1 menerima paket RREQ. *Node* M1 segera mengirimkan paket RREQ menuju *node* M2 melalui *wormhole link* dengan kecepatan tinggi, sehingga paket RREQ dapat lebih dulu sampai di *node* D. *Node* M2 akan meneruskan RREQ ke *node* tetangganya, yakni *node* D yang merupakan *node* tujuan. Setelah itu, *node* D akan mengirimkan paket RREP ke *node* S melalui *reverse path* yang melewati *node* M2 dan M1. Dengan demikian, rute terpilih yang akan digunakan adalah S-F-M1-M2-D. Jika rute sudah ditemukan, maka akan dilakukan pengiriman paket data. *Node* M1 yang menerima paket data akan membuang paket data tersebut. Hal ini menyebabkan paket data tidak sampai ke *node* tujuan.

C. *Metode Delay Per Hop Indicator (Delphi)*

Dalam pendeteksian serangan *wormhole,* metode *Delay Per Hop Indicator* (Delphi) mengumpulkan jumlah *hop* dan informasi *delay* dari *disjoint path* dan menghitung nilai *delay per hop* sebagai indikator untuk mendeteksi serangan *wormhole*. Dalam situasi normal, *delay* dalam suatu *hop* sama dengan *delay* di setiap *hop* di sepanjang rute. Namun jika terdapat serangan *wormhole*, *delay* antara *wormhole node* yang satu dengan yang lainnya sangat tinggi karena sebenarnya ada banyak *hop* di antara *wormhole node* tersebut. Jika dibandingkan *delay per hop* dari rute saat situasi jaringan normal dengan *delay per hop* pada saat terdapat serangan *wormhole*, *delay per hop* dari rute saat situasi jaringan normal lebih kecil. Oleh karena itu, jika sebuah rute memiliki nilai *delay per hop* yang tinggi, maka rute tersebut kemungkinan mengalami serangan *wormhole*

[11]. Ketika suatu *node* menemukan *wormhole node, node* tersebut akan berhenti meneruskan pesan ke *wormhole node* tersebut dan akan mengirim *blocking message* ke *node* sumber. *Node* sumber akan mem-*broadcast blocking message* ke semua *node* di jaringan agar semua *node* pada jaringan menghapus ID *wormhole node* dari tabel *routing* mereka [12].

Kelebihan metode Delphi adalah dapat mendeteksi serangan *wormhole* dengan menghitung *delay per hop* setiap *node,* dapat mendeteksi dua jenis serangan *wormhole,* yakni *hidden attack* dan *expose attack,* tidak membutuhkan sinkronisasi waktu dan posisi setiap *node,* dan tidak membutuhkan tambahan *hardware* untuk mendeteksi serangan *wormhole.* Sedangkan, kekurangan metode Delphi adalah tidak dapat menandai lokasi *wormhole* karena metode ini hanya menghitung *delay* setiap *node* pada suatu rute dan menyimpulkan bahwa nilai DPH pada rute normal lebih kecil daripada nilai DPH rute yang memiliki *tunnel link* [13] [14].

D. Metode Round Trip Time and Topological Comparison (RTT-TC)

Round Trip Time (RTT) didefinisikan sebagai interval waktu antara kapan paket dikirim dan kapan diterima. Dua *wormhole node* dengan *wormhole tunnel* di antaranya biasanya memiliki RTT lebih lama dibandingkan dengan RTT antara dua *node* normal. Metode RTT-TC berdasarkan pada *topological comparison* dan pengukuran *round trip time*. Metode RTT-TC mendeteksi serangan *wormhole* ketika *node* sumber menemukan bagian SUS yang tidak kosong dalam *neighbor list*-nya. Untuk melakukan *topological comparison*, digunakan *enquiry request* (ENQ) dan *enquiry reply* (ENQ_{rep}).

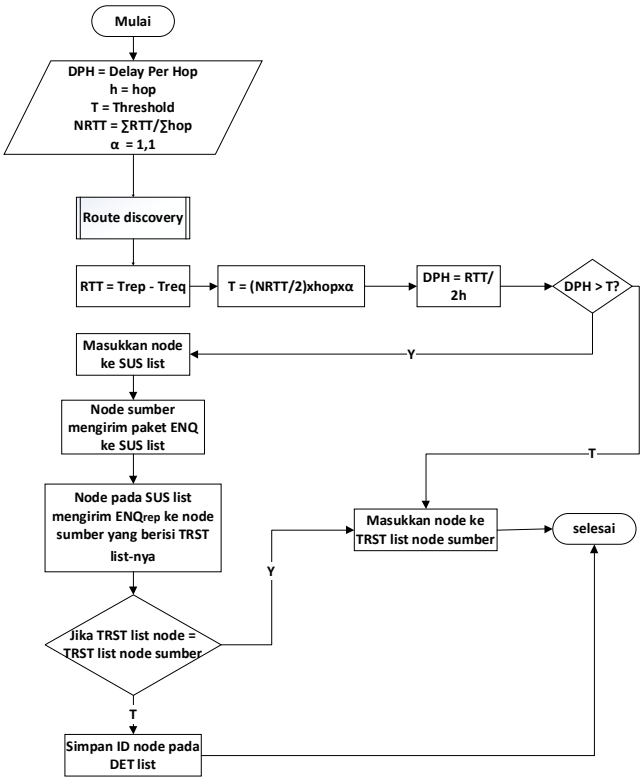
Langkah-langkah *topological comparison* adalah sebagai berikut:

- Setelah dilakukan *route discovery, node* sumber akan melakukan pengirimkan paket ENQ menuju *suspected node* yang terdapat pada SUS *list node* tersebut.
- *Suspected node* akan membalas paket ENQ dengan mengirim ENQ_{rep} yang berisi TRST *list* dari *suspected node*.
- *Node* sumber membandingkan TRST *list* yang diterima dengan TRST *list*-nya sendiri. Jika terdapat *node* yang terkena serangan *wormhole*, maka ID *node* tersebut akan disimpan dalam *detected list* (DET *list*). DET *list* digunakan untuk menyimpan *node-node* yang terkena serangan *wormhole*. Jika tidak terkena serangan *wormhole*, maka *suspected node* dihapus dari SUS *list node* sumber dan dimasukkan dalam TRST *list node* sumber [4].

E. Gabungan Metode Delphi dan RTT-TC

Metode Delphi dan RTT-TC memiliki kelebihan dan kekurangannya masing-masing. Kelebihan metode Delphi adalah dapat mendeteksi *wormhole* dengan menghitung *delay per hop* setiap *node,* namun tidak dapat menandai lokasi *wormhole* [14]. Sedangkan, kelebihan metode RTT-TC adalah dapat menandai lokasi *wormhole* dan menyimpan *wormhole* tersebut pada *detected list* (DET *list*) *node* sumber, namun penggunaan RTT dalam mendeteksi *wormhole* kurang *reliable* [4]. Sehingga pada penelitian ini diusulkan penggabungan metode Delphi dan RTT-TC untuk menutupi kekurangan masing-masing metode.

Untuk pendeteksian serangan *wormhole* digunakan perhitungan metode Delphi dan untuk mencegahannya digunakan metode *topological comparison.* Berikut merupakan diagram alir penggabungan metode Delphi dan metode RTT-TC.



Gambar. 2. Diagram alir gabungan metode Delphi dan RTT-TC

F. Lingkungan Simulasi

Untuk menunjang jalannya simulasi, dibutuhkan beberapa parameter. Parameter-parameter yang dibutuhkan untuk menunjang jalannya simulasi tersaji pada Tabel I.

- *Average End-to-End Delay*

  Average end-to-end delay menghitung rata-rata waktu yang diperlukan suatu paket dari saat paket dikirim sampai diterima oleh *node* tujuan [15]. Persamaan (1) merupakan rumus yang digunakan untuk menghitung *average end-to-end delay*:

$$Average\ end-to-end\ delay = \frac{Total\ delay}{Total\ paket\ yang\ diterima} \quad (1)$$

TABEL I. PARAMETER SIMULASI

Parameter	Nilai
Simulator	NS-2.35
Protokol *routing*	AOMDV
Jenis serangan	*Wormhole*
Jumlah *normal node*	50
Jumlah *wormhole node*	2
Luas area simulasi	800 x 800 m^2 dan 1000 x 1000 m^2
Model mobilitas	*Random Waypoint*
Tipe *traffic*	CBR
Waktu simulasi	150 detik
MAC *Layer*	802.11
Channel	*Wireless*
Parameter uji	*Average end-to-end delay, Packet Delivery Ratio* (PDR), dan *Throughput*

- *Packet delivery ratio* (PDR)
  PDR adalah perbandingan jumlah paket data yang berhasil diterima oleh *node* tujuan dibagi dengan jumlah paket data yang dikirimkan oleh *node* sumber [15]. Untuk menghitung PDR digunakan rumus pada Persamaan (2) berikut:

$$PDR = \frac{Jumlah\ data\ yang\ diterima}{Jumlah\ data\ yang\ dikirim} \times 100 \qquad (2)$$

- *Throughput*
  Throughput yaitu jumlah rata-rata data (bit) yang dikirimkan ke node tujuan per satuan waktu [9]. Untuk menghitung *throughput* digunakan rumus pada Persamaan (3) berikut:

$$Throughput = \frac{Waktu\ ukuran\ data\ yang\ diterima}{Waktu\ pengiriman\ data} \qquad (3)$$

IV. HASIL DAN PEMBAHASAN

Pengujian telah dilakukan pada lima skenario simulasi, yakni simulasi protokol *routing* AOMDV, simulasi protokol *routing* AOMDV dengan serangan *wormhole,* simulasi protokol *routing* AOMDV dengan serangan *wormhole* dan penerapan metode Delphi, simulasi protokol *routing* AOMDV dengan serangan *wormhole* dan penerapan metode RTT-TC, serta simulasi protokol *routing* AOMDV dengan serangan *wormhole* dan penerapan gabungan metode Delphi dan RTT-TC. Setiap skenario simulasi dibagi menjadi dua kondisi, yakni dengan luas area 800 x 800 m^2 dan 1000 x 1000 m^2, dimana disetiap area simulasi kondisi *traffic rate* akan berubah-ubah mulai dari 0,1 MB/s hingga 1,0 MB/s. Kemudian setiap hasil simulasi diuji menggunakan parameter uji *average end-to-end delay,* PDR, dan *throughput.*

A. Perubahan Average End-to-End delay Terhadap Traffic Rate dan Luas Area

TABEL II. KUALITAS AVERAGE END-TO-END DELAY TERHADAP TRAFFIC RATE DAN LUAS AREA

Average End-to-End Delay (second)						
Luas Area (m^2)	Traffic Rate	AOMDV	AOMDV + Wormhole (WAOMDV)	WAOMDV + Delphi	WAOMDV + RTT-TC	WAOMDV + Gabungan Metode
800 x 800 m^2	0.1	0.0376	0.2312	0.0449	0.0639	0.0432
	0.2	0.0111	0.2000	0.0178	0.0308	0.0166
	0.3	0.0093	0.1979	0.0123	0.0209	0.0126
	0.4	0.0086	0.1382	0.0119	0.0182	0.0123
	0.5	0.0082	0.1031	0.0109	0.0171	0.0112
	0.6	0.0075	0.1256	0.0102	0.0158	0.0100
	0.7	0.0080	0.0890	0.0098	0.0158	0.0094
	0.8	0.0072	0.0827	0.0092	0.0139	0.0094
	0.9	0.0074	0.0548	0.0088	0.0129	0.0095
	1.0	0.0072	0.0465	0.0086	0.0131	0.0092
Luas Area (m^2)	Traffic Rate	AOMDV	AOMDV + Wormhole (WAOMDV)	WAOMDV + Delphi	WAOMDV + RTT-TC	WAOMDV + Gabungan Metode
1000 x 1000 m^2	0.1	0.0679	0.3818	0.0859	0.1260	0.0887
	0.2	0.0159	0.3003	0.0182	0.0389	0.0199
	0.3	0.0127	0.3134	0.0141	0.0351	0.0166
	0.4	0.0102	0.2668	0.0127	0.0279	0.0133
	0.5	0.0098	0.2302	0.0118	0.0247	0.0124
	0.6	0.0096	0.1829	0.0110	0.0231	0.0121
	0.7	0.0092	0.1788	0.0108	0.0205	0.0111
	0.8	0.0089	0.1776	0.0101	0.0171	0.0111
	0.9	0.0082	0.1270	0.0097	0.0168	0.0112
	1.0	0.0084	0.1497	0.0096	0.0163	0.0101

Berdasarkan Tabel II, dapat diketahui bahwa nilai terbaik *average end-to-end delay* AOMDV pada saat kondisi normal adalah 0,0072 sec kemudian meningkat menjadi 0,0465 sec saat terdapat serangan *wormhole.* Nilai *average end-to-end delay* kembali menurun setelah diimplementasikan metode Delphi dan RTT-TC, dimana nilai terbaik *average end-to-end delay* adalah 0,0086 sec pada saat menggunakan metode Delphi dan 0,0131 sec pada saat menggunakan metode RTT-TC. Pada saat gabungan metode Delphi dan RTT-TC diimplementasikan, nilai *average end-to-end delay* AOMDV menjadi lebih baik dan hampir mendekati nilai *average end-to-end delay* AOMDV saat kondisi normal, yakni 0,0092 sec. Sedangkan, nilai *average end-to-end delay* AOMDV di luas area 1000 x 1000 m^2 mencapai nilai terbaik pada saat kondisi normal yakni 0,0084 sec kemudian meningkat menjadi 0,1497 sec saat terdapat serangan *wormhole.* Nilai *average end-to-end delay* kembali menurun setelah diimplementasikan metode Delphi dan RTT-TC dimana nilai terbaik *average end-to-end delay* adalah 0,0096 sec pada saat menggunakan metode Delphi dan 0,0163 sec pada saat menggunakan metode RTT-TC. Pada saat gabungan metode Delphi dan RTT-TC diimplementasikan, nilai *average end-to-end delay* AOMDV menjadi lebih optimal dan hampir mendekati nilai *average end-to-end delay* AOMDV saat kondisi normal, yakni 0,0101 sec.

Berdasarkan pengimplementasian ketiga metode tersebut, yakni metode Delphi, RTT-TC, dan gabungan metode Delphi dan RTT-TC, dapat diketahui bahwa gabungan metode Delphi dan RTT-TC dapat menurunkan *delay* pada protokol *routing* AOMDV yang mengalami kenaikan karena adanya serangan *wormhole*. Selain itu, nilai *average end-to-end delay* pada luas area 1000 x 1000 m^2 juga lebih tinggi dibandingkan dengan luas area 800 x

800 m^2, karena luas area yang semakin bertambah menyebabkan jarak antar *node* di suatu jaringan menjadi cukup jauh, sehingga menyebabkan semakin panjang rute yang akan dilewati untuk mencapai *node* tujuan.

B. Perubahan Packet Delivery Ratio (PDR) Terhadap Traffic Rate dan Luas Area

Berdasarkan Tabel III, pada luas area 800 x 800 m^2 nilai tertinggi PDR AOMDV pada saat kondisi normal adalah 71,836% kemudian menurun menjadi 30,928% saat terdapat serangan *wormhole*. Pada saat metode Delphi dan RTT-TC diimplementasikan, nilai PDR AOMDV meningkat dibandingkan pada saat terkena serangan *wormhole* dan mencapai nilai tertinggi yakni 48,800% pada saat menggunakan metode Delphi dan 61,975% pada saat menggunakan metode RTT-TC. Pada saat gabungan metode Delphi dan RTT-TC diimplementasikan, nilai PDR AOMDV menjadi lebih baik, yakni 67,701% dan hampir mendekati nilai PDR AOMDV pada saat kondisi normal. Sedangkan, nilai tertinggi PDR AOMDV di luas area 1000 x 1000 m^2 saat kondisi normal adalah 68,067% kemudian menurun menjadi 24,411% saat terdapat serangan *wormhole*. Pada saat metode Delphi dan RTT-TC diimplementasikan, nilai PDR AOMDV meningkat dibandingkan pada saat terkena serangan *wormhole* dan mencapai nilai tertinggi yakni 46,308% pada saat menggunakan metode Delphi dan 58,978% pada saat menggunakan metode RTT-TC. Pada saat gabungan metode Delphi dan RTT-TC diimplementasikan, nilai PDR AOMDV menjadi lebih baik dan hampir mendekati nilai PDR AOMDV pada saat kondisi normal yakni 60,848%.

TABEL III. KUALITAS PDR TERHADAP TRAFFIC RATE DAN LUAS AREA

Packet Delivery Ratio (%)						
Luas Area (m^2)	**Traffic Rate**	**AOMDV**	**AOMDV + Wormhole (WAOMDV)**	**WAOMDV + Delphi**	**WAOMDV + RTT-TC**	**WAOMDV + Gabungan Metode**
800 x 800 m^2	0.1	71.836	30.928	48.800	61.975	67.701
	0.2	47.199	19.965	29.884	42.879	43.561
	0.3	45.520	15.280	28.174	39.377	42.402
	0.4	45.021	13.533	26.544	40.149	41.903
	0.5	45.455	13.876	27.325	39.530	41.987
	0.6	44.801	14.271	26.895	39.390	41.085
	0.7	44.056	15.370	27.163	37.670	40.518
	0.8	45.119	14.768	27.726	37.862	40.300
	0.9	44.473	14.914	27.418	36.513	39.669
	1.0	43.569	14.580	27.207	37.406	39.264
Luas Area (m^2)	**Traffic Rate**	**AOMDV**	**AOMDV + Wormhole (WAOMDV)**	**WAOMDV + Delphi**	**WAOMDV + RTT-TC**	**WAOMDV + Gabungan Metode**
1000 x 1000 m^2	0.1	68.067	24.411	46.308	58.978	60.848
	0.2	43.871	12.938	23.221	33.685	34.398
	0.3	35.408	9.376	22.568	31.891	32.460
	0.4	36.357	8.161	22.536	31.354	33.500
	0.5	35.414	8.036	22.949	30.569	32.816
	0.6	34.850	8.156	22.755	30.781	33.029
	0.7	34.910	7.990	22.339	30.522	31.576
	0.8	35.042	8.000	22.127	30.788	31.544
	0.9	34.606	8.364	22.754	30.323	30.127
	1.0	34.468	8.786	22.580	30.482	31.105

C. Perubahan Throughput Terhadap Traffic Rate dan Luas Area

Berdasarkan Tabel IV, dapat diketahui bahwa nilai tertinggi *throughput* AOMDV pada saat kondisi normal adalah 3,01 Kbps, kemudian menurun menjadi 1,23 Kbps saat terdapat serangan *wormhole*. Nilai *throughput* kembali meningkat setelah diimplementasikan metode Delphi dan RTT-TC dan mencapai nilai tertinggi yakni 1,85 Kbps pada saat menggunakan metode Delphi dan 2,57 Kbps pada saat menggunakan metode RTT-TC. Pada saat gabungan metode Delphi dan RTT-TC diimplementasikan, nilai *throughput* AOMDV menjadi lebih baik, yakni 2,65 Kbps. Sedangkan, nilai tertinggi *throughput* AOMDV pada saat kondisi normal adalah 2,37 Kbps kemudian menurun menjadi 0,77 Kbps saat terdapat serangan *wormhole*. Nilai *throughput* kembali meningkat setelah diimplementasikan metode Delphi dan RTT-TC dan mencapai nilai tertinggi, yakni 1,63 Kbps pada saat menggunakan metode Delphi dan 2,06 Kbps pada saat menggunakan metode RTT-TC. Pada saat gabungan metode Delphi dan RTT-TC diimplementasikan, nilai *throughput* AOMDV menjadi lebih baik, yakni 2,09 Kbps.

TABEL IV. KUALITAS THROUGHPUT TERHADAP TRAFFIC RATE DAN LUAS AREA

Throughput (Kbps)						
Luas Area (m2)	**Traffic Rate**	**AOMDV**	**AOMDV + Wormhole (WAOMDV)**	**WAOMDV + Delphi**	**WAOMDV + RTT-TC**	**WAOMDV + Gabungan Metode**
800 x 800 m^2	0.1	0.55	0.26	0.37	0.45	0.48
	0.2	0.67	0.33	0.43	0.61	0.61
	0.3	0.99	0.39	0.59	0.84	0.92
	0.4	1.23	0.49	0.72	1.13	1.15
	0.5	1.58	0.62	0.95	1.39	1.46
	0.6	1.86	0.69	1.10	1.66	1.67
	0.7	2.15	0.84	1.28	1.84	1.92
	0.8	2.41	0.97	1.54	2.09	2.21
	0.9	2.73	1.11	1.69	2.29	2.44
	1.0	3.01	1.23	1.85	2.57	2.65
Luas Area	**Traffic Rate**	**AOMDV**	**AOMDV + Wormhole (WAOMDV)**	**WAOMDV + Delphi**	**WAOMDV + RTT-TC**	**WAOMDV + Gabungan Metode**
1000 x 1000 m^2	0.1	0.52	0.20	0.32	0.44	0.45
	0.2	0.54	0.23	0.35	0.49	0.50
	0.3	0.76	0.26	0.49	0.65	0.67
	0.4	1.01	0.31	0.67	0.87	0.93
	0.5	1.21	0.38	0.83	1.05	1.13
	0.6	1.44	0.47	0.98	1.26	1.36
	0.7	1.70	0.50	1.13	1.47	1.52
	0.8	1.91	0.59	1.28	1.72	1.73
	0.9	2.12	0.68	1.47	1.89	1.87
	1.0	2.37	0.77	1.63	2.06	2.09

Pengimplementasian gabungan metode Delphi dan RTT-TC cukup efektif untuk meningkatkan kualitas *throughput* pada protokol *routing* AOMDV yang mengalami penurunan karena adanya serangan *wormhole*. Selain itu, nilai *throughput* pada area 800 x 800 m^2 juga lebih tinggi dibandingkan dengan area 1000 x 1000 m^2, karena pada luas area yang semakin bertambah menyebabkan jarak antara *node* yang satu dengan yang lainnya di suatu jaringan menjadi cukup jauh, sehingga menyebabkan semakin panjang rute yang akan dilewati untuk mencapai *node* tujuan.

Peningkatan *traffic rate* menyebabkan semakin cepat lalu lintas paket data. Hal ini juga menyebabkan semakin banyak paket data yang berhasil diterima oleh *node* tujuan serta waktu pengiriman paket data yang semakin cepat.

Dengan adanya peningkatan *traffic rate,* kualitas *delay* semakin baik dan kualitas *throughput* semakin meningkat. Sedangkan, kualitas PDR semakin menurun seiring dengan peningkatan *traffic rate.* Hal ini disebabkan karena PDR tidak hanya dipengaruhi oleh jumlah paket data yang berhasil diterima saja, tetapi juga dipengaruhi oleh jumlah paket data yang dikirimkan. Semakin tinggi *traffic rate,* semakin banyak pula paket data yang dikirimkan dan berhasil diterima. Selisih antara paket yang dikirim dan diterima pun semakin meningkat. Kondisi inilah yang menyebabkan kualitas PDR semakin menurun seiring dengan peningkatan *traffic rate*.

V. KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan penelitian dan analisis yang telah dilakukan, maka diperoleh kesimpulan dan saran sebagai berikut.

1. Penerapan serangan *wormhole* menyebabkan peningkatan *delay* serta penurunan nilai PDR dan *throughput* pada protokol *routing* AOMDV.
2. Peningkatan *traffic rate* menyebabkan semakin cepat lalu lintas paket data. Hal ini menyebabkan semakin banyak paket data yang berhasil dikirimkan ke *node* tujuan serta waktu pengiriman paket data yang semakin cepat.
3. Perbedaan luas area dapat mempengaruhi perbedaan nilai *average end-to-end delay, packet delivery ratio,* dan *throughput.* Hal ini disebabkan karena semakin luas suatu area, maka pergerakan *node* semakin leluasa dan jarak antar *node* pada jaringan semakin jauh. Dari hasil simulasi yang diperoleh dapat diketahui bahwa, penambahan luas area menyebabkan *average end-to-end delay* semakin meningkat serta nilai PDR dan *throughput* semakin menurun seiring dengan adanya serangan *wormhole.*
4. Berdasarkan hasil pengujian, penerapan gabungan metode Delphi dan RTT-TC dapat mendeteksi dan mencegah serangan *wormhole*. Gabungan metode Delphi dan RTT-TC dapat menurunkan nilai *delay* serta meningkatkan nilai PDR dan *throughput* pada protokol *routing* AOMDV.

B. Saran

Pada penelitian yang akan datang diharapkan adanya simulasi dan pengujian pada berbagai macam protokol *routing* untuk mengetahui perbedaan kinerja di setiap protokol-protokol tersebut. Selain itu, diharapkan adanya variasi jumlah *node* penyerang serta pengujian kinerja menggunakan beberapa parameter uji lainnya untuk pengembangan penelitian yang lebih baik lagi.

DAFTAR PUSTAKA

[1] A. T. S. Putranto, Analisis Penggunaan Energy AODV dan DSDV pada Mobile Ad Hoc Network, Yogyakarta: Program Studi Teknik Informatika Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Sanata Dharma, 2016.

[2] F. A. Jenefer and D. Vydeki, "Performance Analysis of Mobile Ad Hoc Network in the Presence of Wormhole Attack," *International Journal of Advance Computer Engineering and Communication Technology,* vol. 2, no. 1, pp. 2278-5140, February 2013.

[3] H. S. Chiu and K.-S. Lui, "DelPHI: Wormhole Detection Mechanism for Ad Hoc Wireless Networks," *The 1st International Symposium on Wireless Pervasive Computing,* February 2006.

[4] M. R. Alam, Detecting Wormhole and Byzantine Attacks in Mobile ad hoc Networks, Curtin University of Technology : School of Electrical and Computer Engineering , 2011.

[5] S. Kaushal and R. Aggarwal, "Avoidance of Wormhole Attack by Using Delphi Method," *International Research Journal of Engineering and Technology,* vol. 02, no. 07, October 2015.

[6] R. Kaur and A. Bansal, "Detection and Prevention of Wormhole Attack on AOMDV Routing Protocol using Hop-count and Communication Range in WSN," *International Journal of Innovations & Advancement in Computer Science,* vol. 7, no. 4, April 2018.

[7] S. K. Arora and M. Ayushree, "Detection and Performance Analysis of Wormhole Attack in MANET Using DELPHI Technique," *International Journal of Security and Its Applications,* vol. 10, pp. 321-330, 2016.

[8] M. G. P. Sanaei, I. F. Isnin and M. Bakhtiari, "Performance Evaluation of Routing Protocol on AODV and DSR under Wormhole Attack," *International Journal of Computer Networks and Communication Security,* vol. 1, pp. 1-6, June 2013.

[9] B. B. Putra and R. Anggoro, "Studi Kinerja Multipath AODV dengan Menggunakan Network Simulator 2 (NS-2)," vol. 5, pp. A652-A656, 2016.

[10] F. A. Jenefer and D. Vydeki, "Performance Analysis of Mobile Ad Hoc Network in the Presence of Wormhole Attack," *International Journal of Advance Computer Engineering and Communication Technology,* vol. 2, no. 1, pp. 2278-5140, February 2013.

[11] H. S. Chiu and K.-S. Lui, "DelPHI: Wormhole Detection Mechanism for Ad Hoc Wireless Networks," February 2006.

[12] S. Shamaei and A. Movaghar, "A Two-Phase Wormhole Attack Detection Scheme in MANETs," *The ISC int'l of Information Security,* vol. 6, pp. 183-191, July 2014.