ANALISIS KINERJA PROTOKOL AODV (AD HOC ON DEMAND DISTANCE VECTOR) TERHADAP SERANGAN *SYBIL* PADA JARINGAN MANET (MOBILE AD HOC NETWORK)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Ajisuryo Tricahyo Purnomo
NIM: 145150200111129

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2021

PENGESAHAN

ANALISIS KINERJA PROTOKOL AODV (AD HOC ON DEMAND DISTANCE VECTOR) TERHADAP SERANGAN *SYBIL* PADA JARINGAN MANET (MOBILE AD HOC NETWORK)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Ajisuryo Tricahyo Purnomo
NIM: 145150200111129

Skripsi ini telah diuji dan dinyatakan lulus pada

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Ir. Primantara Hari Trisnawan, M.Sc.	Reza Andria Siregar, S.T., M.Kom.
NIP. 19680912 199403 1 002	NIP. 19790621 200604 1 003

Mengetahui
Ketua Jurusan Teknik Informatika

Achmad Basuki, S.T., M.MG., Ph.D.
NIP. 19741118 200312 1 002

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar referensi.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 26 juli 2021

Ajisuryo Tricahyo Purnomo

NIM: 145150200111129

PRAKATA

Puji syukur bagi Allah SWT yang telah memberikan rahmat serta karunia-Nya kepada penulis, sehingga penulis dapat menyelesaikan skripsi yang berjudul "ANALISIS KINERJA PROTOKOL AODV (AD HOC ON DEMAND DISTANCE VECTOR) TERHADAP SERANGAN SYBIL PADA JARINGAN MANET (MOBILE AD HOC NETWORK)" sebagai syarat untuk menyelesaikan Program Sarjana (S1) pada Program Studi Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.

Penulis menyadari bahwa Skripsi ini tidak luput dari bantuan berbagai pihak yang telah mendukung dan membimbing penulis. Oleh karena itu, dengan penuh rasa bersyukur, penulis ingin mengucapkan rasa terima kasih kepada :

1. Kedua orang tua, Ir. Isbenu Purwoko dan Mujiningtyas yang senantiasa hadir untuk memberi dukungan moral maupun materil.
2. Bapak Ir. Primantara Hari Trisnawan, M.Sc. dan Bapak Reza Andria Siregar, S.T., M.Kom. selaku dosen pembimbing yang senantiasa sabar dalam menasehati, membimbing dan meluangkan waktunya untuk penulis dalam penyusunan skripsi ini.
3. Bapak Achmad Basuki, S.T., M.MG., Ph.D. selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
4. Bapak Adhitya Bhawiyuga, S.Kom., M.Sc. selaku Ketua Program Studi Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
5. Kedua kakak penulis, Aji Pratomo Suryo Prayugo dan Indah Ratna Dwijatidiri Andari yang tak henti-hentinya mendoakan serta menyemangati penulis.
6. Teman-teman satu angkatan Program Studi Teknik Informatika 2014 yang selalu memberikan informasi, semangat dan doa. Terutama kepada saudara angki, yogi, agri, diki, retno dan kemas.
7. Seluruh civitas akademika Fakultas Ilmu Komputer Universitas Brawijaya yang telah banyak memberi bantuan selama proses penyelesaian skripsi ini.
8. Semua pihak yang telah menolong dalam penyelesaian skripsi dan tidak dapat penulis sebutkan.

Penulis mengharapkan segala bentuk saran maupun kritik yang membangun pada skripsi ini. Semoga skripsi ini dapat bermanfaat bagi para pembaca dan semua pihak.

Malang, 26 juli 2021

Ajisuryo Tricahyo Purnomo
Ajisuryo6@gmail.com

ABSTRAK

MANET (*Mobile Ad hoc Network*) merupakan salah satu tipe jaringan yang tidak mempunyai infrastruktur tetap. Semua aktivitas pada jaringan seperti penentuan topologi dan pengiriman paket dilakukan oleh node itu sendiri. Dibandingkan dengan sekuritas jaringan kabel, MANET memiliki banyak permasalahan karena ketiadaan *authority* terpusat, topologi jaringan yang dinamis, *Bandwith* rendah, batasan baterai dan *memory* dari perangkat *mobile*. Pada MANET semua *node-node* bebas untuk bergabung dan keluar dari jaringan, ini disebut sebagai *open boundary networks.* MANET tidak memiliki garis pertahanan yang jelas, keamanan adalah prioritas utama dalam berkomunikasi di lingkungan apa pun. Berlandaskan latar belakang diatas, maka penulis membuat penelitian yang berjudul Analisis Kinerja Protokol AODV (Ad hoc On demand Distance Vector) Terhadap Serangan Sybil pada Jaringan MANET (Mobile Ad hoc Network). Hasil yang diperoleh membuktikan bahwa serangan Sybil memengaruhi performa protokol AODV. Pengujian dalam penelitian ini diterapkan pada *Network Simulator 2* dan melalui 6 metode evaluasi yaitu *packet delivery ratio, end to end delay, normalized routing protokol, throughput* dan tabel *routing.* Hasil menunjukkan bahwa nilai *packet delivery ratio* terendah terjadi pada skenario 100 *node* dengan 10 penyerang 71,87%. End to end delay paling tinggi terjadi pada skenario 100 *node* dengan 10 penyerang 399,42 ms. *Normalized routing load* tertinggi terjadi pada skenario 100 *node* dengan 10 penyerang 49,4. *Throughput* terendah terjadi pada skenario 100 *node* dengan 10 penyerang 61,45 kbps. Tabel *routing* terbanyak terjadi pada skenario 100 *node* dengan 10 penyerang sejumlah 53.

Kata kunci: MANET, AODV, *sybil*, *packet delivery ratio, end to end delay, normalized routing load, throughput*

ABSTRACT

MANET (Mobile Ad hoc Network) is a type of network that does not have a fixed infrastructure. In MANET all nodes are free to join and leave the network, these are called open boundary networks. MANET has no clear line of defense, security is the top priority in communicating in any environment. Based on the above background, the author made a research entitled AODV (Ad hoc On demand Distance Vector) Protocol Performance Analysis Against Sybil Attacks on the MANET Network (Mobile Ad hoc Network). The results obtained prove that the Sybil attack affects the performance of the AODV protocol. Tests in this study are applied to Network Simulator 2 and through 6 evaluation methods, namely packet delivery ratio, end to end delay, normalized routing protocol, throughput and routing table. The results show that the lowest packet delivery ratio value occurs in the 100 node scenario with 10 attackers worth 71.87%. The highest end to end delay occurs in the 100 node scenario with 10 attackers worth 399.42 ms. The highest normalized routing load occurs in the 100 node scenario with 10 attackers worth 49.4. The lowest throughput occurs in the scenario of 100 nodes with 10 attackers worth 61.45 kbps. Most routing tables occur in the 100 node scenario with 10 attackers totaling 53.

Keywords: MANET, AODV, *sybil*, *packet delivery ratio, end to end delay, normalized routing load, throughput*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE PROGRAM

BAB 1 PENDAHULUAN

1.1 Latar Belakang

MANET (*Mobile Ad hoc Network*) merupakan salah satu tipe jaringan yang tidak mempunyai infrastruktur tetap. Setiap *node-node* dilengkapi dengan *transmitter* dan *receiver* nirkabel. *Node-node* pada manet dapat bergerak secara bebas dan dapat mengatur diri mereka sendiri dengan cara acak. Protokol *routing* merupakan bagian penting pada jaringan *Ad hoc* karena topologi yang terus-menerus berubah yang disebabkan oleh pergerakan *node-node*. Semua aktivitas pada jaringan seperti discovering topologies dan pengiriman paket dilakukan oleh node itu sendiri. *Node* berkomunikasi melalui nirkabel, dan harus menghadapi kendala dalam komunikasi radio, seperti *noise* dan interferensi. Setiap *node* pada jaringan MANET berfungsi sebagai *host* dan juga sebagai *router*. Kontrol pada jaringan didistribusikan ke semua *node-node* pada jaringan. (Kumar T. L., 2016)

Dibandingkan dengan sekuritas jaringan kabel, MANET memiliki banyak permasalahan karena ketiadaan *authority* terpusat, topologi jaringan yang dinamis, *Bandwith* rendah, batasan baterai dan *memory* dari perangkat *mobile*. (Khirasaraya, 2013)

Pada MANET semua *node-node* bebas untuk bergabung dan keluar dari jaringan, ini disebut sebagai *open boundary networks.* Semua *node-node* perantara sumber dengan tujuan mengambil peran dalam *routing*, juga disebut sebagai komunikasi *hop-by-hop*. Karena media komunikasinya nirkabel, setiap *node* menerima paket dalam jangkauan nirkabel, apakah merupakan paket destinasi ataupun bukan. Karena sebab itu, setiap *node* dapat dengan mudah mendapatkan akses ke paket *node-node* yang lain atau menginjeksi paket *fault* dalam jaringan. Oleh sebab itu, mengamankan MANET dari perilaku dan *node-node* berbahaya menjadi salah satu tantangan terpenting pada MANET. (Dorri, Kamel, & Kheyrkhah, 2015)

MANET tidak memiliki garis pertahanan yang jelas, keamanan adalah prioritas utama dalam berkomunikasi di lingkungan apa pun. Node penyerang dapat menyerang dari kedua sisi jaringan. Dengan adanya node penyerang, perhatian utama dari jaringan tersebut adalah untuk menentukan solusi keamanan yang efektif, yang melindungi MANET dari serangan perutean yang berbeda. Semua node dalam jaringan meneruskan data ke tujuan; karena itu, MANET lebih rentan terhadap serangan oleh node jahat, seperti *blackhole*, *wormhole*, *Denial of Service*(DoS), *eavesdropping*, *spoofing*, dll. (Thapara & Sharmab, 2019)

Terdapat 2 jenis serangan, yakni serangan aktif dan serangan pasif. Pada serangan aktif, penyerang mencoba *bypass* atau menembus kedalam sistem. Serangan pasif memonitor trafik yang tidak terenkripsi dan mencari *password* yang tertulis dan juga informasi penting yang dapat digunakan untuk serangan-serangan lainnya. (saha, et al., 2012)

Pada MANET *node* tidak memiliki informasi fisik atas *node* lainnya satu-satunya cara agar dapat mengetahui adanya keberadaan *node* lain adalah dengan pesan *request* dan *response*. Sistem seperti ini harus dapat memastikan bahwa sebuah identitas merujuk sebuah entitas tertentu, jika tidak maka sebuah entitas dapat menggunakan beberapa identitas sehingga dapat menyebabkan kekacauan pada MANET. Serangan *Sybil* merupakan serangan dari perangkat penyerang yang mengadopsi beberapa identitas secara ilegal dan identitas tambahannya disebut sebagai *Sybil node* (Sinha, 2013).

Penelitian ini akan melihat bagaimana identitas lain dari *Sybil* digunakan untuk menyerang ketersediaan jalur komunikasi pada jaringan serta membandingkan kinerja protokol AODV sebelum dan setelah diserang dengan serangan *Sybil*.

1.2 Identifikasi Masalah

Berdasarkan dari latar belakang, maka dapat disusun identifikasi masalah sebagai berikut:

1. *Node* penyerang dapat membuat beberapa identitas palsu yang digunakan untuk mengirimkan paket *request* ke *node* target.
2. Ketiadaan otoritas terpusat mengakibatkan setiap node bebas menginjeksi paket *fault* dalam jaringan.

1.3 Rumusan Masalah

Berdasarkan latar belakang dari masalah yang telah dijelaskan, maka dapat disusun rumusan masalah berikut:

1. Bagaimana kinerja protokol AODV sebelum dan setelah diserang menggunakan serangan Sybil?
2. Bagaimana cara serangan *Sybil* dapat menginjeksi paket *fault* terhadap protokol *routing* AODV menggunakan perangkat lunak *Network Simulator-2?*

1.4 Tujuan

Berdasarkan latar belakang dari masalah yang telah dijelaskan, maka dapat disusun tujuan berikut:

1. Menerapkan aturan protokol AODV pada jaringan MANET.
2. Menganalisis kinerja protokol AODV.
3. Menganalisis kinerja sistem setelah diserang.
4. Mengetahui dampak serangan *Sybil* pada jaringan.

1.5 Manfaat

Manfaat dari penelitian ini adalah mendapatkan hasil dari analisis kinerja protokol *routing* AODV setelah diserang menggunakan *Sybil*.

1.6 Batasan Masalah

1. Penelitian menggunakan protokol *routing* AODV.
2. Simulasi dilakukan menggunakan *Network Simulator-2*.
3. Serangan yang dilakukan berupa serangan aktif yaitu *Sybil*.
4. Penilaian QoS(Quality of Service) dinilai dengan *packet delivery ratio*, end-to end delay, normalized routing load, throughput dan tabel *routing*.
5. *Node* penyerang memiliki 5 identitas tambahan . 4 identitas digunakan sebagai identitas palsu dan 1 identitas digunakan sebbagai identitas target.

1.7 Sistematika Pembahasan

Sistematika pembahasan dari penyusunan penelitian yang direncanakan adalah sebagai berikut :

BAB 1 PENDAHULUAN

Pada bab Pendahuluan berisikan latar belakang pembuatan tugas akhir, identifikasi, batasan masalah, rumusan masalah, tujuan dan manfaat serta sistematika penulisan dari tugas akhir yang berjudul "Analisis Kinerja Protokol AODV (Ad hoc On Demand Distance Vector) Terhadap Serangan *Sybil* Pada Jaringan MANET (Mobile Ad hoc Network)".

BAB 2 LANDASAN KEPUSTAKAAN

Bab landasan kepustakaan menguraikan tentang teori – teori yang dipakai dalam penelitian, temuan dan bahan penelitian yang diperoleh dari beberapa referensi yang menunjang penelitian dalam penulisan tugas akhir dengan judul "Analisis Kinerja Protokol AODV (Ad hoc On Demand Distance Vector) Terhadap Serangan *Sybil* Pada Jaringan MANET (Mobile Ad hoc Network)".

BAB 3 METODOLOGI

Bab ini membahas metodologi yang digunakan pada penelitian agar penelitian dapat berjalan secara terstuktur. Di sini berisi pemaparan langkah kerja yang terdiri atas studi literatur, analisis kebutuhan, perancangan sistem, metode evaluasi, teknik pengambilan kesimpulan dan saran, dan analisis.

BAB 4 IMPLEMENTASI

Bab ini berisi tentang implementasi dari perancangan serangan *Sybil* yang telah dipaparkan pada sub bab perancangan.

BAB 5 PENGUJIAN DAN PEMBAHASAN

Bab ini berisi tentang pengujian beserta pembahasan dari simulasi serangan *Sybil* pada protokol AODV.

BAB 6 PENUTUP

Bab ini membahas metodologi yang digunakan pada penelitian agar penelitian dapat berjalan secara terstuktur. Di sini berisi pemaparan langkah kerja yang terdiri atas studi literatur, analisis kebutuhan, perancangan sistem, pengujian, teknik pengambilan kesimpulan dan saran, dan analisis.

BAB 2 LANDASAN KEPUSTAKAAN

Penelitian ini dikaji dengan penelitian-penelitian yang telah dilakukan sebelumnya yang akan dijadikan pedoman dalam penelitian ini. Berikut adalah kajian pustaka yang dijabarkan pada tabel 2.1

2.1 Kajian Pustaka

Tabel 2.1 Kajian Pustaka

No.	Judul dan Tahun	Penulis	Hasil Penelitian	Penelitian Penulis
1	A survey: Ad-hoc on Demand Distance Vector(AODV) Protocol[2017]	Meeta Singh, PhD, Sudeep Kumar	Menyediakan informasi dasar dari protokol AODV.	Mengimplementasikan protokol AODV pada NS-2
2	Security Attacks In MANET – A comprehensive Study[2020]	Manmohan Sharma, Mamoon Rashid	Menjelaskan secara ringkas serangan-serangan yang ada pada MANET	Mengimplentasikan serangan *Sybil*.
3	Securing TORA against *Sybil* attack in MANETs[2015]	Suraj Thawani, Hardik Upadhyay	Mengamankan protokol TORA dari serangan *Sybil*	Analisis serangan *Sybil* pada protokol AODV

Penelitian yang yang dilakukan Meeta Singh dan Sudeep Kumar yang berjudul *A survey: Ad-hoc on Demand Distance Vector*(AODV) *Protocol* (2017) menjelaskan bahwa protokol *routing* pada MANET berfungsi untuk menemukan dan menetapkan rute tiap-tiap *node* agar bisa mengirimkan paket data dari *node* sumber sampai ke *node* tujuan, klasifikasi protokol *routing* pada MANET yang dibedakan menjadi 3 yakni proaktif, reaktif dan hibrida. serta mekanisme pencarian rute dan perawatan rute pada AODV. Relasi pada penelitian ini adalah penggunaan protokol AODV yang akan diterapkan pada NS-2.

Penelitian yang dilakukan Manmohan Sharma dan Mamoon Rashid yang berjudul Security Attacks In MANET – A comprehensive Study(2020) memaparkan secara ringkas jenis-jenis serangan yang terdapat pada MANET. Jenis serangan umumnya dibedakan menjadi 2 yakni aktif dan pasif. Serangan aktif mencoba untuk merusak sistem keamanan yang ada di dalam jaringan untuk mendapatkan akses dari data atau *node*. Sedangkan serangan pasif akan mengamati dan menganalisa ketika terjadinya trafik. Tetapi nantinya, pengamatan ini dapat

digunakan untuk mengimplementasikan segala jenis serangan pada jaringan. relasi pada penelitian ini adalah sebagai identifikasi jenis-jenis serangan yang terdapat pada MANET.

Penelitian yang dilakukan Suraj Thawani dan Hardik Upadhyay yang berjudul Securing TORA against *Sybil* attack in MANETs (2015) menerapkan pendeteksian *Sybil* dengan menambahkan nilai RSS pada setiap node. Nilai ini akan digunakan sebagai ambang batas untuk membedakan *node* penyerang dengan *node* normal. kesimpulan dari penelitian tersebut menyatakan bahwa serangan *Sybil* adalah serangan yang melemahkan sebuah jaringan, penyerang membuat beberapa identitas-identitas palsu dalam jaringan dengan alamat yang berbeda-beda. Hasil yang diperoleh dari penelitian tersebut mengatakan bahwa serangan *Sybil* mengakibatkan kinerja seluruh jaringan menurun. Relasi pada penelitian ini adalah kesamaan jenis serangan yang diterapkan.

Berdasarkan penelitian-penelitian yang telah dilakukan sebelumnya, saya akan membuat penelitian yang berjudul "ANALISIS KINERJA PROTOKOL AODV (AD HOC ON DEMAND DISTANCE VECTOR) TERHADAP SERANGAN *SYBIL* PADA JARINGAN MANET (MOBILE AD HOC NETWORK)". Perbedaan penelitian yang akan dilakukan dengan penelitian sebelumnya terletak pada jenis serangan.

2.2 Dasar Teori

2.2.1 Mobile Ad hoc Network (MANET)

Dibandingkan jaringan nirkabel infrastruktur, dimana setiap pengguna berkomunikasi melalui *access point* ataupun *base station*, *mobile ad hoc network,* atau MANET, tidak memerlukan infrastruktur untuk proses operasinya. MANET terdiri dari kumpulan transitori otonom *node-node* yang saling berkomunikasi secara nirkabel. *Node-node* yang termasuk dalam jangkauan pengiriman satu sama lain dapat langsung berkomunikasi dan bertanggung jawab menemukan satu sama lain secara dinamis. Untuk mengaktifkan komunikasi antar *node-node* yang tidak dalam jangkauan, *node* perantara berperan sebagai router yang meneruskan paket yang dibuat *node* agar sampai ke tujuan. Perangkat-perangkat ini bebas untuk bergabung atau keluar dari jaringan, mereka juga dapat bergerak secara acak sehingga dapat menghasilkan perubahan topologi yang tidak terprediksi dan cepat. Dalam kondisi ini *node-node* perlu mengorganisir diri mereka sendiri secara dinamis untuk dapat menyediakan fungsionalitas jaringan yang diperlukan dengan ketidak hadirannya infrastruktur tetap atau administrasi pusat. (Hoebeke, Moerman, Dhoedt, & Demeester, 2004)

2.2.2 Protokol *Routing*

MANET adalah jaringan tanpa infrastruktur, terekelola sendiri dan *multi-hop* dengan topologi yang berubah-ubah karena pergerakan *node-node*nya sehingga koneksi nirkabel terputus dan terhubung kembali dengan cepat. (Anuj K. Gupta, 2011). Oleh karena itu, protokol *routing* diperlukan untuk menemukan dan

membuat jalur komunikasi antar *node-node* agar bisa saling berkomunikasi. Protokol pada manet dibagi menjadi 2 kategori, yakni reaktif dan proaktif

2.2.2.1 *Protokol Routing reaktif*

Dalam protokol *routing* reaktif, jalur dibuat hanya ketika dibutuhkan. Contoh dari protokol ini ialah *Dynamic Source Routing Protocol (DSR)* dan *Ad hoc On-demand Distance Vector Routing Protocol (AODV).* (Anuj K. Gupta, 2011)

2.2.2.2 *Protokol Routing proaktif*

Dalam protokol *routing* aktif atau *Table-driven*, *node-node* terus memperbarui tabel *routing* melalui pesan periode. Contoh dari protokol ini ialah *Optimized Link State Routing Protocol (OLSR)* dan *Destination Sequenced Distance Vector Protocol (DSDV).* (Anuj K. Gupta, 2011)

2.2.3 Ad hoc On-demand Distance Vector Routing(AODV)

Pada AODV untuk menemukan jalur sampai ke tujuan, *node* sumber akan mengirimkan pesan *broadcast route request packet* (RREQ). Pesan ini akan disebarkan pada jaringan hingga mencapai *node* perantara yang memiliki informasi akan *node* tujuan atau telah sampai ke *node* tujuan. Ketika *node* perantara mem-*forward* paket RREQ *node* ini menyimpan dalam tabelnya sendiri tempat pesan itu berasal. Informasi ini digunakan untuk membentuk jalur *reply* untuk rute *reply* karena AODV hanya menggunakan *symetric links.* Saat perjalanan balik paket *reply*, *node-node* yang dilewati akan menyimpan informasi *routing* dalam tabel mereka. Ketika ada kesalahan pada sambungan, *node* sumber diberi notifikasi dan penemuan rute (RREQ) dapat dikirim lagi jika diperlukan. (Anuj K. Gupta, 2011)

2.2.4 *Sybil* Attack

Serangan *Sybil* pertama kali dikenalkan oleh J.C. Douccur. Menurut Douccur, serangan *Sybil* adalah serangan dimana sebuah entitas dapat memiliki beberapa identitas sehingga dapat menimbulkan kekacauan dari sistem.

Adapun taksonomi dari serangan *Sybil* seperti berikut:

Dimensi 1 : *Direct* vs *Indirect*. Pada serangan *Direct, node* penyerang langsung berkomunikasi di dalam jaringan. Sedangkan pada serangan *Indirect* node penyerang akan berkomunikasi dengan jaringan melalui *node* perantara.

Dimensi 2 : *Fabricated* vs *Stolen*. pada serangan *Fabricated, node* penyerang membuat sendiri identitas yang akan digunakan. Sedangkan pada serangan *Stolen, node* penyerang akan menggunakan identitas yang sudah ada di dalam jaringan.

Dimensi 3 : *Simultaneous* vs *non-Simultaneous.* Pada serangan *Simultaneous*, *node* penyerang akan menggunakan identitas-identitasnya secara bersamaan.

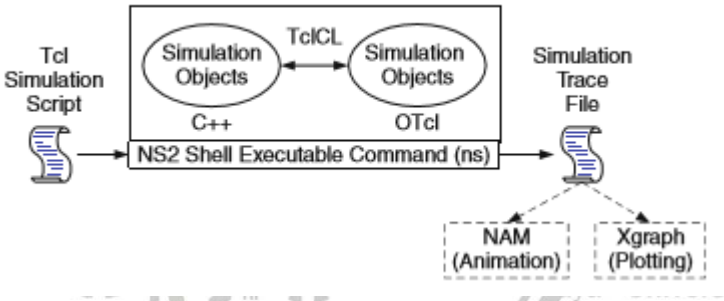
Sedangkan pada serangan *non-Simultaneous*, *node* penyerang akan merubah-ubah identitasnya secara berkala. (Sinha, 2013)

Pada MANET, satu-satunya cara agar sebuah entitas mendeteksi kehadiran entitas lain adalah dengan mengirim dan menerima pesan *broadcast*. Dengan memanfaatkan fitur ini, sebuah *node* penyerang dapat mengirim pesan dengan banyak identitas palsu. *Node* yang me-*spoofing* identitas *node* lain disebut sebagai *node* penyerang/penyerang *Sybil*, dan *node-node* yang identitas nya telah ter-*spoofed* disebut dengan *node Sybil*. (Kaur, 2014)

2.2.5 Network Simulator 2

Network Simulator 2 (NS2) adalah sebuah simulator *event* yang ditargetkan untuk penelitian jaringan. NS2 menyediakan dukungan untuk melakukan simulasi protokol TCP, pencarian rute, dan *multicast* melalui jaringan kabel dan nirkabel (lokal dan satelit). (Issariyakul, 2012).

NS2 terdiri dari 2 bahasa utama, yaitu C++ dan *Object-oriented Tool Command Language* (Otcl). Bahasa C++ digunakan untuk mendefinisikan mekanisme *internal* dari simulasi (*Backend*), sementara Octl berfungsi untuk mengatur proses simulasi dengan cara membangun dan mengkonfigurasi setiap objek pada simulasi dengan baik (*Frontend*).



Gambar 2.1 Arsitektur NS2

Sumber: Introduction to Network Simulator 2 (NS2)

Gambar 2.11 menjelaskan mengenai arsitektur dasar dari NS2, mulai dari *script TCL,* kemudian diproses dalam simulasi C++ dan Otcl yang akan menghasilkan *tracefile* berupa animasi serta rekaman pengiriman paket pada jaringan. Tabel 2.2 adalah *format* dalam setiap baris yang ada pada *tracefile.*

Tabel 2.2 Format Pada File Trace

peristiwa	Waktu	*Node*	*layer*	Unique ID	Tipe paket	Ukuran paket	Informasi Physical	informasi IP *trace*	Informasi AODV *trace*

2.2.6 *Packet Delivery Ratio*

Packet delivery ratio (PDR) merupakan jumlah semua paket data yang diterima dibagi dengan jumlah semua paket data yang dikirim (Harahap, 2014). Semakin tinggi nilai PDR maka hasil dari pengujiannya baik. Rumus untuk mencari *packet delivery ratio* adalah:

$$\textit{Packet Delivery Ratio} = \frac{Jumlah\ seluruh\ paket\ data\ yang\ diterima}{jumlah\ seluruh\ paket\ yang\ dikirim}\ X\ 100\% \quad (2.1)$$

2.2.7 End-to-End Delay

End-to-end delay merupakan jumlah total waktu pengiriman paket yang berhasil dibagi dengan jumlah seluruh paket yang terkirim (Harahap, 2014). Semakin tinggi nilai *delay* maka hasil dari pengujiannya buruk. Rumus untuk mencari *end-to-end delay* adalah:

$$\textit{End to End Delay} = \frac{Jumlah\ total\ waktu\ pengiriman\ paket\ yang\ berhasil\ terkirim}{jumlah\ seluruh\ paket\ yang\ terkirim} \quad (2.2)$$

2.2.8 *Normalized Routing Load*

Normalized routing load (NRL) merupakan jumlah total paket *routing* (RREQ, RREP, dan RRER) dibagi dengan seluruh paket data yang diterima. Semakin tinggi nilai NRL maka kurang protokol *routing* tersebut kurang efisien (Harahap, 2014). Rumus untuk mencari *normalized routing load* adalah:

$$\textit{Normalized Routing Load} = \frac{Jumlah\ total\ paket\ routing\ (RREQ,RREP,RRER)}{jumlah\ seluruh\ paket\ data\ yang\ diterima} \quad (2.3)$$

2.2.9 Throughput

Throughput merupakan total paket yang terkirim dibagi dengan total waktu pengiriman. (Goswami, Joardar, Das, & Kar, 2017). Rumus untuk mencari *throughput* adalah:

$$\textit{Throughput} = \frac{Jumlah\ total\ paket\ terkirim}{total\ waktu\ pengiriman} \quad (2.4)$$

BAB 3 METODOLOGI PENELITIAN

3.1 Kerangka Penelitian

Pada bagian ini menjelaskan metode yang digunakan dalam pengembangan penelitian ini. Gambar 3.1 merupakan tahapan-tahapan metodologi penelitian untuk penelitian ini.

Gambar 3.1 Diagram Alir Metode Penelitian

3.2 Studi Literatur

Studi literatur digunakan sebagai dasar dan landasan terhadap seluruh perancangan dan implementasi dair penelitian ini. Literatur yang digunakan adalah:

- *Mobile Ad-Hoc Network* (MANET)
- AODV
- *Sybil attack*

- *Network Simulator*-2

3.3 Analisis Kebutuhan

Analisis kebutuhan diperlukan untuk mengetahui apa saja yang diperlukan dalam penelitian ini. Berikut ini merupakan kebutuhkan diperlukan.

3.3.1 Kebutuhan Fungsional

Kebutuhan fungsional dibutuhkan untuk mengetahui apa saja yang akan dilakukan oleh sistem yang akan dibuat dan berisi informasi mengenai apa yang ada dan dapat dihasilkan sistem. Kebutuhan fungsional pada pengujian ini adalah:

1. Sistem dapat mengimplementasikan protokol routing AODV.
2. Sistem dapat mengatur pergerakan dan letak *node* selama simulasi.
3. Sistem dapat menerapkan mekanisme serangan berupa *Sybil*
4. Sistem dapat meletakkan *node* agen sebagai penyerang.
5. Sistem dapat memberikan rincian hasil simulasi berupa data selama proses dilakukan.

3.3.2 Kebutuhan Non-Fungsional

Kebutuhan Non-fungsional adalah kebutuhan perangkat keras dan perangkat lunak yang digunakan dalam penelitian ini. Berikut kebutuhan non-fungsional sistem ini, yaitu:

1. Perangkat keras:
   a. 1 buah komputer/laptop
2. Perangkat lunak:
   a. Ubuntu linux 20.10
   b. Network Simulator-2 (Software Simulasi)

3.4 Perancangan Sistem

Pada bab ini diterangkan tahapan apa saja yang diperlukan untuk mengimplementasikan pengujian skenario serangan *Sybil* pada protokol AODV di dalam MANET. Implementasi ini akan dilakukan menggunakan *Network Simulator 2*.

3.4.1 Perancangan Parameter Pengujian

Perancangan parameter pengujian dapat dilihat pada Tabel 3.1

Tabel 3.1 parameter pengujian

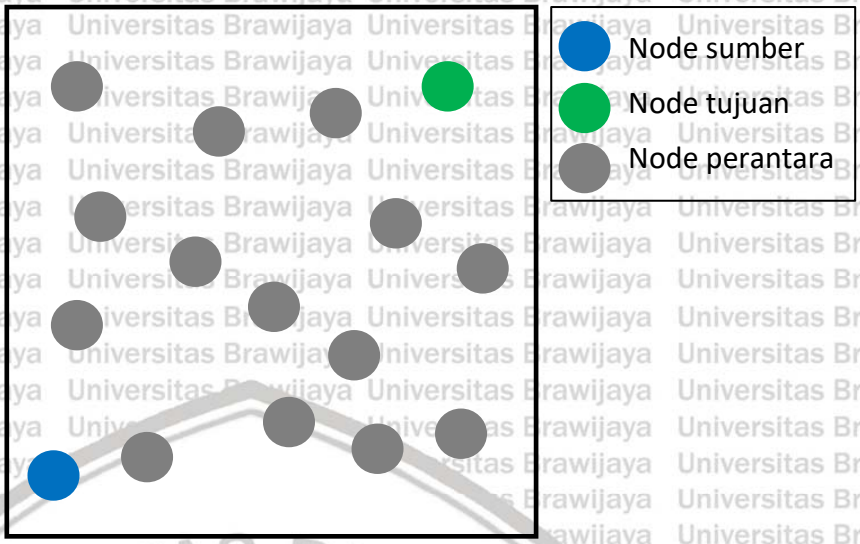
No.	Parameter	Spesifikasi
1.	Network Simulator	Network Simulator 2
2.	Routing Protocol	AODV
3.	Waktu simulasi	1000 detik
4.	Luas Area	1000 x 1000(meter)
5.	Jumlah *Nodes*	20,50,100
6.	Jumlah node penyerang	2,4,6,8,10
7.	Model Pergerakan *Node*	Random Way Point
8.	Pause Time	1 detik
9.	Tipe Koneksi	UDP(CBR)
10.	Data rate	10 mbps
11.	Besar paket data	1024 bytes
12.	Kecepatan node	Random, maksimal 5 m/s

Perangkat lunak yang digunakan ialah *Network Simulator* 2.Protokol yang digunakan yaitu AODV. Pengujian terhadap protokol AODV diterapkan pada beberapa skenario, yaitu skenario 20, 50 dan 100 node. Adapun serangan yang digunakan yaitu serangan *Sybil*. Jumlah node penyerang juga bervariasi antara 0, 2, 4, 8 dan 10. Pengiriman dimulai pada detik 50 dan berakhir pada detik 1000. Besar ukuran paket yang dikirim yakni 1024 Bytes. Memiliki *Transmission Data Rate* sebesar 0,1 mbps. Node penyerang aktif pada detik 50 pada skenario yang menerapkan serangan. Identitas palsu yang digunakan dipilih secara acak diantara 0 sampai dengan 1000 dan identitas target dipilih secara acak dengan kondisi identitas bukan merupakan bagian dari topologi.

3.4.2 Perancangan Topologi Jaringan

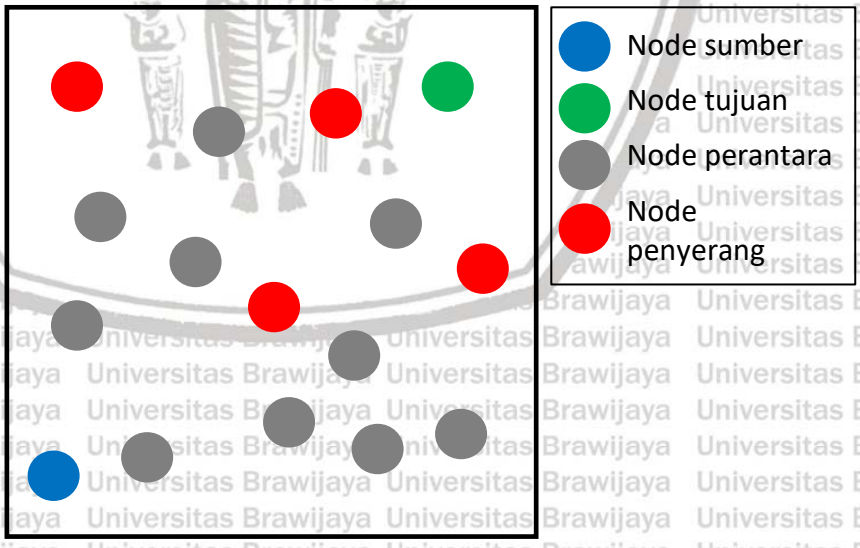
Perancangan topologi jaringan yang akan dibuat dalam pengujian ini adalah perancangan topologi yang terdiri atas 20, 50, dan 100 *nodes* yang diatur secara

acak sesuai skenario yang telah dibuat. Berikut adalah gambaran topologi jaringan 20, 50, dan 100 *nodes*.



Gambar 3.2 rancangan topologi jaringan tanpa serangan

Gambar 3.2 merupakan rancangan topologi jaringan tanpa adanya serangan pada sistem yang memiliki luas area 1000m x 1000m. dengan notasi *node* berwarna biru sebagai node sumber yang mengirimkan paket data pada saat simulasi berjalan, node berwarna hijau sebagai node tujuan yang menerima paket data, node abu-abu sebagai node perantara yang bertugas meneruskan paket dari *node* sumber ke *node* tujuan.



Gambar 3.3 rancangan topologi jaringan dengan serangan

Gambar 3.3 merupakan rancangan topologi jaringan dengan adanya serangan pada sistem yang memiliki luas area 1000m x 1000m. dengan notasi *node* berwarna biru sebagai node sumber yang mengirimkan paket data pada saat simulasi berjalan, node berwarna hijau sebagai node tujuan yang menerima paket

data, node abu-abu sebagai node perantara yang bertugas meneruskan paket dari *node* sumber ke *node* tujuan. *Node* berwarna merah sebagai *node* penyerang yang bertugas mengganggu kinerja jaringan.

3.4.3 Perancangan Pergerakan *Node*

Random Way Point adalah sebuah model pergerakan acak untuk setiap *node* yang berada pada jaringan dan bagaimana lokasi, kecepatan *node* dapat berubah sewaktu-waktu dalam area simulasi. Setiap pergerakan *node* tidak terikat satu sama lain/bebas. Proses pergerakan dari mode *Random Way Point* ini adalah *node* akan diberikan tujuan, kecepatan, dan arah secara acak. Setelah *node* tersebut mencapai tujuan pertama, maka *node* tersebut akan berhenti sebentar sebelum akhirnya bergerak kembali menuju ke lokasi tujuan selanjutnya. Pada NS2 tersedia fungsi *setdest* untuk mengimplementasikan pergerakan *node*.

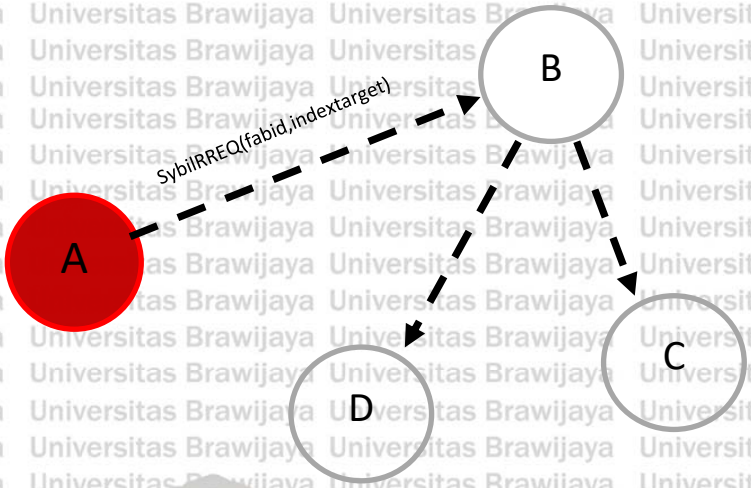
3.4.4 Perancangan Serangan

3.4.4.1 Perancangan skenario tanpa serangan

Perancangan skenario dilakukan dengan mengimplementasikan parameter pengujian pada *script* tcl. Menentukan tipe protokol *routing* yakni AODV, membuat koneksi UDP dari node sumber ke node tujuan dengan model *traffic constant bit-rate*(CBR). Skenario pengujian dibagi menjadi 3 bagian yaitu simulasi dengan jumlah *node* 20, 50 dan 100 dengan durasi selama 1000 detik. Paket akan dikirimkan pada detik ke 0 sampai detik ke 1000. Skenario akan menggunakan pergerakan *node* yang telah dibuat secara acak dengan fungsi *setdest.*

3.4.4.2 Perancangan skenario dengan serangan *Sybil*

Pada simulasi serangan *Sybil* ini dilakukan dengan menambahkan tingkah laku *node Sybil* pada file *.tcl, aodv.h dan aodv.cc. Serangan *Sybil* yang diimplementasikan merupakan serangan dari dalam sehingga penyerang sudah berada dalam topologi jaringan. *Node* penyerang akan memiliki 5 identitas tambahan yakni index0, index1, index2, index3 dan indextarget. *Node* penyerang akan mengirim pesan RREQ yang telah dimanipulasi dengan memasukkan identitas tambahan dari node penyerang secara berkala setiap 5 detik. Pesan palsu ini akan dikirimkan dengan indextarget sebagai tujuan.



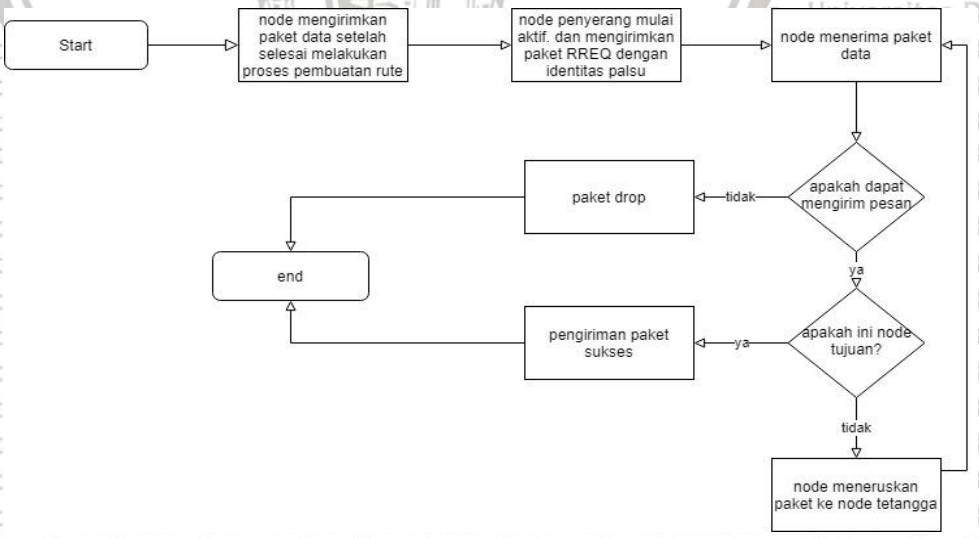
Gambar 3.4 Rancangan serangan *Sybil*

Gambar 3.4 merupakan rancangan serangan *Sybil* pada sebuah topologi jaringan. *node* A merupakan *node* penyerang dan akan mengirimkan pesan dengan identitas palsu beserta identitas tujuan pesan tersebut. *Node* B, C dan D merupakan *node* normal, jika bukan merupakan tujuan dari pesan maka paket dari penyerang akan diteruskan ke-*node* yang lain.

Skenario serangan *Sybil* yang akan digunakan dalam pengujian adalah sebagai berikut :

1. Skenario 20 node dengan *node* penyerang sebanyak 0, 2, 4, 6, 8, 10 *node*
2. Skenario 50 node dengan *node* penyerang sebanyak 0, 2, 4, 6, 8, 10 *node*
3. Skenario 100 node dengan *node* penyerang sebanyak 0, 2, 4, 6, 8, 10 *node*

Gambar 3.5 merupakan *flowchart* serangan *Sybil*:



Gambar 3.5 flowchart serangan *Sybil*

Gambar 3.5 merupakan *flowchart* serangan *Sybil* pada simulasi

3.5 Metode Evaluasi

Pada tahap ini penulis memaparkan tentang parameter dan skenario uji yang akan diterapkan saat penelitian.

3.5.1 Skenario Pengujian

Pada penelitian ini pengujian dilakukan dengan menggunakan 3 skenario 20, 50 dan 100 *node*. Adapun serangan yang digunakan yaitu serangan *Sybil*. Jumlah *node* penyerang bervariasi antara 0, 2, 4, 8 dan 10. *Node* penyerang aktif pada detik 50 pada skenario yang menerapkan serangan. Identitas palsu yang digunakan dipilih secara acak diantara 0 sampai dengan 1000 dan identitas target dipilih secara acak dengan kondisi identitas bukan merupakan bagian dari topologi.

3.5.2 Pengujian Packet Delivery Ratio

Proses pengujian *Packet Delivery Ratio* dilakukan melalui pengolahan data *tracefile*. Data pada *tracefile* dipilah untuk menentukan total paket yang terkirim dan paket yang diterima, kemudian membagi paket yang diterima dengan paket yang terkirim. Perhitungan ini diterapkan pada program AWK.

3.5.3 Pengujian End-to-End Delay

Proses pengujian *end to end delay* dilakukan melalui pengolahan data *tracefile*. Data pada *tracefile* dipilah untuk menentukan jumlah total waktu pengiriman paket dan jumlah total pengiriman paket, kemudian membagi total waktu dengan total pengiriman paket. Perhitungan ini diterapkan pada program AWK.

3.5.4 Pengujian Normalized Routing Load

Proses pengujian *Normalized Routing Load* dilakukan melalui pengolahan data *tracefile*. Data pada *tracefile* dipilah untuk menentukan jumlah total paket diterima dan jumlah total *routing* paket, kemudian membagi total *routing* paket dengan total paket yang diterima. Perhitungan ini diterapkan pada program AWK.

3.5.5 Pengujian throughput

Proses pengujian *throughput* dilakukan melalui pengolahan data *tracefile*. Data pada *tracefile* dipilah untuk menentukan jumlah total paket diterima dan lama durasi pengiriman, kemudian membagi total paket dengan lama durasi pengiriman. Perhitungan ini diterapkan pada program AWK.

3.5.6 tabel Routing

proses pengujian tabel *routing* dilakukan dengan membandingkan tabel *routing node* sumber sebelum dan setelah terjadi serangan.

3.6 Implementasi

Implementasi merupakan realisasi dari perancangan sistem dan metode evaluasi.

3.7 Pengujian dan Pembahasan

Hasil dan analisis digunakan untuk menarik kesimpulan dari pengujian yang telah berlangsung. Disini juga berisi tentang pembahasan dari hasil penelitian.

3.8 Kesimpulan dan Saran

Pengambilan kesimpulan dilakukan berdasarkan pengujian dan analisis terhadap kinerja dari implementasi protokol *routing* AODV yang telah dilakukan. Berdasarkan hasil pengujian, dapat ditentukan perbandingan kedua protokol manakah yang memiliki QoS paling optimal.

BAB 4 IMPLEMENTASI

4.1 Implementasi

Pada bab ini akan dijelaskan langkah-langkah bagaimana cara menerapkan rancangan yang telah dibuat sebelumnya pada *network simulator-2*.

4.1.1 Konfigurasi Skenario Pada Network Simulator-2

Dalam penelitian ini implementasi akan dilakukan menggunakan perangkat lunak *Network Simulator-2*(NS2). Untuk dapat menerapkan skenario yang telah dibuat, akan ada tambahan konfigurasi pada NS2. Untuk konfigurasi serangan *Sybil* perubahan akan dilakukan pada *file* aodv.h dan aodv.cc yang berada dalam direktori home/ns-allinone/ns-2.35/aodv. Kemudian konfigurasi parameter skenario akan dilaksanakan di dalam *file script* *.tcl. Pada penelitian ini implementasi dilakukan pada protokol *routing* AODV dalam lingkungan yang berbeda berdasarkan jumlah node dan juga jumlah node penyerang. Sehingga akan menghasilkan 6 *script* aodv20.tcl, aodv50.tcl, aodv100.tcl, aodv20*Sybil*.tcl, aodv50*Sybil*.tcl, aodv100*Sybil*.tcl.

Konfigurasi serangan *Sybil* diperlukan agar penyerang memilki perilaku dan dapat menjalankan tugasnya saat simulasi sedang berjalan.

4.1.2 Konfigurasi Serangan *Sybil*

Konfigurasi dilakukan dengan cara menambahkan *script* pada *file* aodv.h dan aodv.cc. Kode Program 4.1 merupakan implementasi Serangan *Sybil* pada aodv.h

Kode Program 4.1 Konfigurasi Serangan *Sybil* pada aodv.h

Kode Program 1 : Konfigurasi Serangan *Sybil* pada aodv.h

```
1   #define SYBIL_INTERVAL  5
…   …
2   int           attacker;
3   nsaddr_t      index0;
4   nsaddr_t      index1;
5   nsaddr_t      index2;
6   nsaddr_t      index3;
7   nsaddr_t      indextarget;
…   …
8   class SybilTimer : public Handler
9           {
10        public:
11          SybilTimer(AODV* a):      agent(a){}
12          void      handle(Event*);
13        private:
```

```
14          AODV        *agent;
15          Event       intr;
16      };
17
18  SybilTimer stimer;
19  friend class SybilTimer;
20  void  SybilRREQ(nsaddr_t src, nsaddr_t dst);
```

Penjelasan:

1. Nomor 1 : mendefinisikan interval pemanggilan fungsi *Sybil*RREQ tiap 50 detik.
2. Nomor 2 : penambahan parameter attacker dengan tipe integer pada *constructor* AODV untuk membedakan node attacker dengan node lainnya.
3. Nomor 3-7 : penambahan parameter index0,index1,index2,index3 dan indextarget dengan type nsaddr_t pada *constructor* AODV sebagai tempat menyimpan identitas lainnya, sedangkan indextarget digunakan sebagai alamat tujuan pengiriman fungsi *SybilRREQ*.
4. Nomor 8-16 : penambahan *class timer* pada aodv.h dengan nama *SybilTimer*. Kemudian mendefinisikan bahwa class ini akan menggunakan protokol AODV pada setiap *agent*-nya.
5. Nomor 18 : mendefinisikan objek stimer sebagai bagian dari *class SybilTimer.*
6. Nomor 19 : mendefinisikan *class SybilTimer* sebagai *friend* dari class aodv.h
7. Nomor 20 : mendefinisikan fungsi *Sybil*RREQ dengan parameter src dan dst yang memiliki type nsaddr_t

Kode Program 4.2 merupakan implementasi Serangan *Sybil* pada aodv.cc

Kode Program 4.2 Konfigurasi Serangan *Sybil* pada aodv.cc

Kode Program 2 : Konfigurasi Serangan *Sybil* pada aodv.cc

```
1.  if(strcmp(argv[1], "attacker") == 0)
2.  {
3.          attacker = 1;
4.          return TCL_OK;
5.  }
6.  if(strcmp(argv[1], "indextarget") == 0)
7.  {
8.          indextarget = atoi(argv[2]);
9.          return TCL_OK;
10. }
11. else if(argc == 6){
12. if(strcmp(argv[1], "setOtherId") == 0)
13. {
```

```
14.          index0 = atoi(argv[2]);
15.          index1 = atoi(argv[3]);
16.          index2 = atoi(argv[4]);
17.          index3 = atoi(argv[5]);
18.          return TCL_OK;
19.    }}
…    …
20.  stimer.handle((Event*) 0);
21.  stimer(this)
22.  void SybilTimer::handle(Event*)
23.  {
24.              if (agent->attacker==1)
25.               {
26.              agent->SybilRREQ(agent->index0, agent->indextarget);
27.               agent->SybilRREQ(agent->index1, agent->indextarget);
28.              agent->SybilRREQ(agent->index2, agent->indextarget);
29.               agent->SybilRREQ(agent->index3, agent->indextarget);
30.               }
31.                Scheduler::instance().schedule(this,
32.  &intr, SYBIL_INTERVAL);
33.   }
34.  void AODV::SybilRREQ(nsaddr_t fabid, nsaddr_t dst)
35.  {
36.    Packet *p = Packet::alloc();
37.    struct hdr_cmn *ch = HDR_CMN(p);
38.    struct hdr_ip *ih = HDR_IP(p);
39.    struct hdr_aodv_request *rq = HDR_AODV_REQUEST(p);
40.    aodv_rt_entry *rt = rtable.rt_lookup(dst);
41.    ch->prev_hop_ = fabid;
42.    ih->saddr() = fabid;
43.    rq->rq_src = fabid;
```

Penjelasan:

1. Nomor 1-5 : perintah untuk merubah nilai *attacker* menjadi 1 jika mendapat *input* "attacker" pada *script* tcl.
2. Nomor 6-10 : perintah untuk merubah nilai *indextarget* yang dimiliki oleh *node* dengan nilai *input* yang berasal dari *script* tcl.
3. Nomor 11-19 : perintah untuk merubah nilai index0, index1, index2 dan index3 dengan nilai *input* yang berasal dari *script* tcl.
4. Nomor 20 : menambahkan stimer.handle ((Event*) 0)

5. Nomor 21 : menambahkan stimer(this)
6. Nomor 22-33 : merupakan class timer yang akan dipanggil jika *node* memiliki nilai *attacker* = 1, dan akan memanggil fungsi *Sybil*RREQ dengan *input* index0,index1,index2,index3 sebagai node sumber dan indextarget sebagai node tujuan.
7. Nomor 34-43 : *Sybil*RREQ merupakan fungsi sendRequest yang telah dimodifikasi untuk dapat mengirim *request* dengan identitas palsu. Dengan merubah nilai ch->prev_hop_, ih->saddr() dan rq->rq_src dengan parameter fabid.

4.1.3 Konfigurasi Sistem

Kode Program 4.3 merupakan implementasi Parameter Pengujian pada *script* .tcl

Kode Program 4.3 Konfigurasi Parameter pengujian pada *script* .tcl

Kode Program 3 : Konfigurasi Parameter pengujian pada .tcl

```
1.  set val(chan)    Channel/WirelessChannel    ;
2.  set val(prop)    Propagation/TwoRayGround   ;
3.  set val(netif)   Phy/WirelessPhy            ;
4.  set val(mac)     Mac/802_11                 ;
5.  set val(ifq)     Queue/DropTail/PriQueue    ;
6.  set val(ll)      LL                         ;
7.  set val(ant)     Antenna/OmniAntenna        ;
8.  set val(ifqlen)  50                         ;
9.  set val(nn)      20                         ;
10. set val(rp)      AODV                       ;
11. set val(sc)   ~/ns-allinone-2.35/ns-2.35/indep-utils/cmu-scen-
12. gen/setdest/scen20.tcl
13. set val(cp)   ~/ns-allinone-2.35/ns-2.35/indep-utils/cmu-scen-
14. gen/20cbr.tcl
15. set val(x)       1000                       ;
16. set val(y)       1000                       ;
17. set val(stop)    1000.0                     ;
```

Penjelasan:

1. Nomor 1 : pemilihan tipe kanal yang digunakan yakni *WirelesChannel.*
2. Nomor 2 : pemilihan jenis *radio-propagation* yang digunakan yakni *TwoRayGround.*
3. Nomor 3 : pemilihan jenis *network interface* yang digunakan yakni WirelessPhy.
4. Nomor 4 : pemilihan jenis MAC yang digunakan yakni 802_11.
5. Nomor 5 : pemilihan jenis *interface queue* yang digunakan yakni PriQueue.
6. Nomor 6 : pemilihan jenis *link layer* yang digunakan yakni LL.

7. Nomor 7 : pemilihan jenis antena yang digunakan yakni *OmniAntenna*.
8. Nomor 8 : penentuan nilai *interface queue length* maksimum yakni 50.
9. Nomor 9 : penentuan jumlah *node* dalam simulasi yang memiliki variasi 20, 50 dan 100.
10. Nomor 10 : pemilihan protokol *routing* yang digunakan yakni AODV.
11. Nomor 11-14 : merupakan direktori *file* pergerakan *node* dan direktori *file* prosedur pengiriman paket.
12. Nomor 15-17 : penentuan luas area simulasi dengan nilai 1000m x 1000m dengan durasi simulasi 1000 detik.

4.1.4 Konfigurasi Node Penyerang

Pada penelitian ini konfigurasi *node* penyerang diterapkan pada setiap *script* Tcl. Setiap skenario memiliki jumlah variasi node 0, 2, 4, 6, 8 dan 10. Kode Program 4.4 merupakan implementasi Serangan *Sybil* pada *script* .Tcl

Kode Program 4.4 Konfigurasi Node *Sybil*

Kode Program 4 : Konfigurasi Node *Sybil*

```
1.  $ns_ at 50.0 "[$node_(6) set ragent_] attacker"
2.  $ns_ at 50.0 "[$node_(6) set ragent_] setOtherId 997 228 191 217"
3.  $ns_ at 50.0 "[$node_(1) set ragent_] attacker"
4.  $ns_ at 50.0 "[$node_(1) set ragent_] setOtherId 117 89 95 267"
5.  $ns_ at 50 "[$node_(0) set ragent_] attacker"
6.  $ns_ at 50.0 "[$node_(0) set ragent_] setOtherId 137 88 224 365"
7.  $ns_ at 50 "[$node_(12) set ragent_] attacker"
8.  $ns_ at 50.0 "[$node_(12) set ragent_] setOtherId 171 139 100 232"
9.  $ns_ at 50 "[$node_(9) set ragent_] attacker"
10. $ns_ at 50.0 "[$node_(9) set ragent_] setOtherId 222 78 456 201"
11. $ns_ at 50 "[$node_(11) set ragent_] attacker"
12. $ns_ at 50.0 "[$node_(11) set ragent_] setOtherId 77 68 123 200"
13. $ns_ at 50 "[$node_(4) set ragent_] attacker"
14. $ns_ at 50.0 "[$node_(4) set ragent_] setOtherId 123 18 93 200"
15. $ns_ at 50 "[$node_(5) set ragent_] attacker"
16. $ns_ at 50.0 "[$node_(5) set ragent_] setOtherId 237 348 929 212"
17. $ns_ at 50 "[$node_(18) set ragent_] attacker"
18. $ns_ at 50.0 "[$node_(18) set ragent_] setOtherId 111 333 119 170"
19. $ns_ at 50 "[$node_(19) set ragent_] attacker"
20. $ns_ at 50.0 "[$node_(19) set ragent_] setOtherId 233 128 99 200"
```

Konfigurasi dilaksanakan dengan menuliskan baris "`$ns_ at X "[$node_(Y) set ragent_] Z"` dengan nilai X untuk menentukkan waktu dan Y sebagai node dan Z sebagai argumen tambahan. Jika Z bernilai *attacker*, maka akan terjadi perubahan nilai *attacker* menjadi 1(satu) pada *node* dan jika bernilai *setOtherId* maka akan memasukkan 4 nilai berikutnya sebagai index0, index1, index2, index3 secara beruntun.

4.1.5 Konfigurasi Topologi dan Pergerakan Jaringan

Konfigurasi topologi dan pergerakan dalam jaringan dibuat menggunakan CMU *scenario generator* yang berada pada direktori home/ns-allinone/ns-2.35/indep-utils/cmu-scen-gen/setdest.

Konfigurasi topologi jaringan dibuat sesuai dengan rancangan parameter. Tiap skenario memiliki masing-masing 20, 50 dan 100 *node*. Berikut Kode Program 4.5 merupakan baris perintah pemanggilan fitur *setdest*:

Kode Program 4.5 Konfigurasi Pergerakan Node

Kode Program 5 : Konfigurasi Pergerakan *Node*

```
$ ./setdest -v 1 -n 50 -M 5 -P 1 -x 1000 -y 1000 > scen50.tcl
```

Baris diatas merupakan perintah untuk menjalankan *scenario generator*. *Setdest* akan membuat skenario sesuai dengan input pengguna. Penjelasan baris akan dijabakarkan sebagai berikut:

-v : versi *setdest*

-n : jumlah node

-M : nilai kecepatan maksimal *node*

-P : *pause time*

-x : panjang dimensi X

-y : panjang dimensi Y

File script disimpan dengan nama scen50.tcl

4.1.6 Konfigurasi Pengiriman Paket

Konfigurasi pengiriman paket dibuat sesuai dengan rancangan topologi dan parameter yang telah ditentukan pada bab sebelumnya. *Node* sumber dan *node* tujuan dipilih sesuai dengan hasil konfigurasi topologi yang saling berjauhan agar paket terkirim melalui beberapa *intermediate node*. Konfigurasi pengiriman paket dibuat dengan menggunakan CMU *scenario generator* yang berada pada direktori home/ns-allinone/ns-2.35/indep-utils/cmu-scen-gen/.

Berikut Kode Program 4.6 merupakan baris perintah pemanggilan fitur cbrgen.tcl:

Kode Program 4.6 Konfigurasi Pengiriman Paket

Kode Program 6 : Konfigurasi Pengiriman Paket

```
$ ns cbrgen.tcl -type cbr -nn 50 -seed 1 -mc 5 -rate 10.0 > 50cbr.tcl
```

Penjelasan baris akan dijabarkan sebagai berikut:

-type : penentuan protokol *transport*
-nn : jumlah *node*
-seed : nilai *seed*
-mc : koneksi maksimum
-rate : data rate

File script disimpan dengan nama 50cbr.tcl

Berikut merupakan Kode Program 4.7 merupakan hasil eksekusi kode program 4.6

Kode Program 4.7 Konfigurasi Protokol Transport

Kode Program 7 : Konfigurasi Protokol Transport

```
1.  set udp_(0) [new Agent/UDP]
2.  $ns_ attach-agent $node_(7) $udp_(0)
3.  set null_(0) [new Agent/Null]
4.  $ns_ attach-agent $node_(2) $null_(0)
5.  set cbr_(0) [new Application/Traffic/CBR]
6.  $cbr_(0) set packetSize_ 1024
7.  $cbr_(0) set interval_ 0.1
8.  $cbr_(0) set random_ null
9.  $cbr_(0) attach-agent $udp_(0)
10. $ns_ connect $udp_(0) $null_(0)
11. $ns_ at 50 "$cbr_(0) start"
12. $ns_ at 1000 "$cbr_(0) stop"
```

Penjelasan :

1. Nomor 1-5 : inisiasi *Agent* baru pada node yang ditentukan. *Agent* UDP merupakan *node* sumber, sedangkan *Agent Null* merupakan *node* tujuan. CBR merupakan jenis *traffic* yang digunakan.
2. Nomor 6 : penentuan ukuran paket.
3. Nomor 7 : penentuan interval.
4. Nomor 8 : penentuan nilai *random.*
5. Nomor 9 : menghubungkan aplikasi *traffic* dengan *Agent* UDP.
6. Nomor 10 : Menghubungkan *node* sumber dangan *node* tujuan.
7. Nomor 11 : penentuan waktu aplikasi dimulai.
8. Nomor 12 : penentuan waktu aplikasi berhenti.

4.1.7 Konfigurasi Pemrosesan Data Output

Berikut Kode Program 4.8 merupakan implementasi pemrosesan Data Output yang terdapat pada *script* tcl.

Kode Program 4.8 Konfigurasi Pemrosesam Data Output

Kode Program 8 : Konfigurasi Pemrosesan Data *Output*

```
1.  set tracefile [open AODV20Sybil.tr w]
2.  $ns_ trace-all $tracefile
3.  set namfile [open AODV20Sybil.nam w]
4.  $ns_ namtrace-all $namfile
```

Penjelasan:

1. Nomor 1 : perintah *set* variable *tracefile* untuk membuka *file* AODV20*Sybil*.tr dan memiliki wewenang untuk menulis data.
2. Nomor 2 : perintah variable ns_ untuk memanggil fungsi *trace-all* pada variable *tracefile.*
3. Nomor 3 : perintah *set* variable *namfile* untuk membuka *file* AODV20*Sybil*.nam dan memiliki wewenang untuk menulis data.
4. Nomor 4 : perintah variable ns_ untuk memanggil fungsi *namtrace-all* pada variable *namfile.*

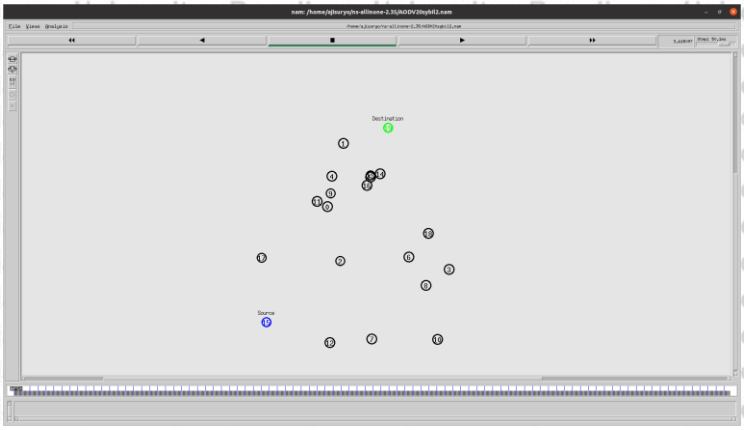
4.2 Simulasi Implementasi *Network Simulator*-2

Simulasi merupakan hasil dari implementasi rancangan skenario yang telah dibuat pada penelitian ini.Simulasi pada Network Simulator-2 dijalankan dengan menggunakan perintah "ns [*filname*].tcl".Gambar 4.1 merupakan gambar hasil eksekusi dari *file* dengan ekstensi Tcl.

```
ajisuryo@SKRIPSI:~/ns-allinone-2.35$ ns AODV50sybil2.tcl
num_nodes is set 50
INITIALIZE THE LIST xListHead
Loading connection pattern...
Loading scenario file...
Load complete...
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5,  distCST_ = 550.0
SORTING LISTS ...DONE!
```

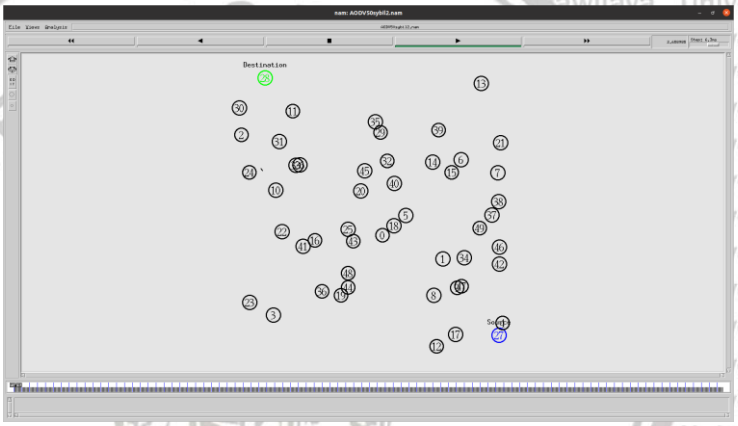
Gambar 4.1 Hasil eksekusi *file* AODV50*Sybil*2.tcl

Jika eksekusi perintah telah berhasil terlaksana dan tidak ada pesan "*Error*", maka akan menampilkan hasil berupa *network animation file* yang memiliki format .nam. Tampilan hasil akan ditunjukkan pada Gambar 4.2, Gambar 4.3 dan Gambar 4.4.



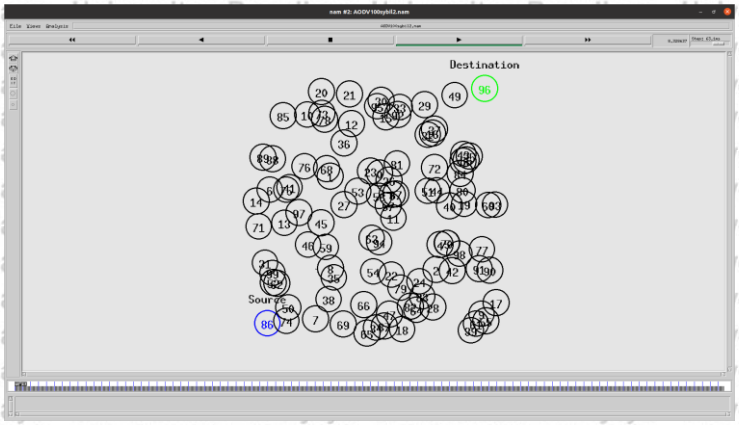
Gambar 4.2 Hasil Simulasi AODV20*Sybil*2.nam

Gambar 4.2 merupakan hasil simulasi dengan *file* AODV20*Sybil*2.nam. Penempatan *node* sumber dengan *node* tujuan yang sedemikian rupa sehingga pengiriman melalui beberapa *node* perantara.



Gambar 4.3 Hasil Simulasi AODV50*Sybil*2.nam

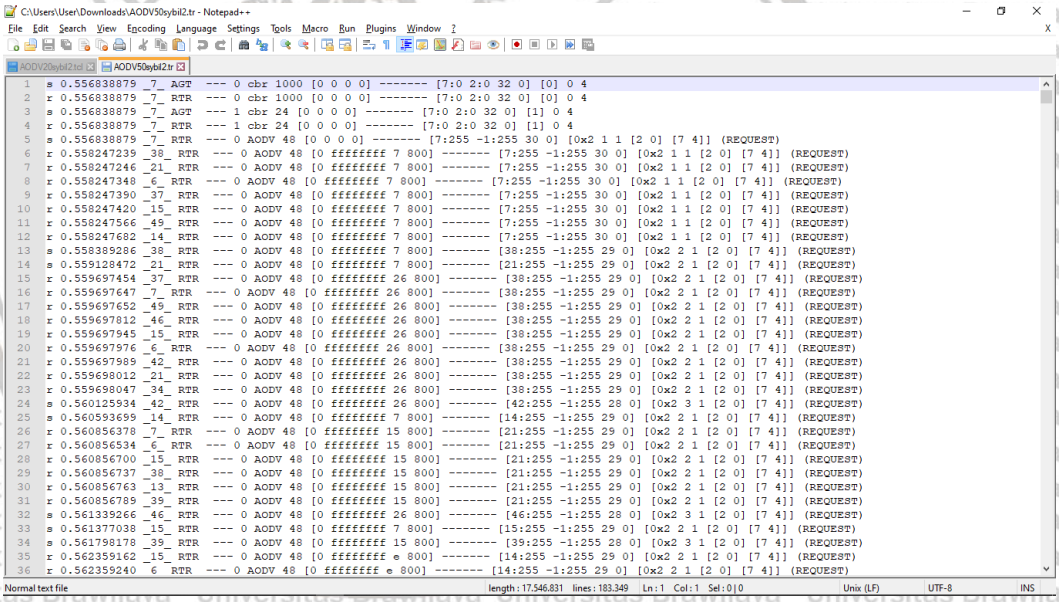
Gambar 4.3 merupakan hasil simulasi dengan *file* AODV50*Sybil*2.nam. Penempatan *node* sumber dengan *node* tujuan yang sedemikian rupa sehingga pengiriman melalui beberapa *node* perantara.



Gambar 4.4 Hasil Simulasi AODV100*Sybil*2.nam

Gambar 4.4 merupakan hasil simulasi dengan *file* AODV100*Sybil*2.nam. Penempatan *node* sumber dengan *node* tujuan yang sedemikian rupa sehingga pengiriman melalui beberapa *node* perantara.

Hasil lainnya berupa *tracefile* akan tersimpan dengan format (.tr). *File* ini menyimpan rekaman seluruh lalu lintas data yang terjadi selama simulasi berjalan. Tampilan hasil akan ditunjukkan pada Gambar 4.5



Gambar 4.5 hasil simulasi AODV50*Sybil*2.tr

Selain dari *network animation file* dan *tracefile*. Peneliti juga mencetak tabel *routing node* sumber. *File* ini disimpan dengan format .txt.

Node id	time	destination	nexthop	hop count	seq.no
NODE: 27	972.647283	33	0	255	5
NODE: 27	972.647283	38	0	255	7
NODE: 27	972.647283	44	0	255	11
NODE: 27	972.647283	2	0	255	7
NODE: 27	972.647283	19	0	255	7
NODE: 27	972.647283	18	0	255	5
NODE: 27	972.647283	26	0	255	9
NODE: 27	972.647283	11	0	255	5
NODE: 27	972.647283	10	0	255	7
NODE: 27	972.647283	28	7	4	106

Gambar 4.6 hasil tabel *routing* pada *node* sumber

Gambar 4.6 merupakan tampilan tabel *routing* yang telah disimpan. Kolom pertama berisi tentang identitas *node* 27, kolom kedua berisi tentang waktu pencetakkan tabel *routing*, kolom ketiga berisi tentang alamat tujuan, kolom keempat berisi tentang jalur *nexthop*, kolom kelima berisi tentang jumlah hop dan kolom keenam berisi tentang nomor *sequence*.

4.3 Pengumpulan dan Pengambilan Data

Pengumpulan dan pengambilan data dilakukan dengan mengolah data yang tersimpan pada *tracefile*. Data yang telah tersimpan pada *tracefile* akan diproses menggunakan pemrograman AWK. Pemrograman AWK dapat mengolah data berdasarkan kolom pada sebuah *file.*

peristiwa	Waktu	*Node*	*layer*	Unique ID	Tipe paket	Ukuran paket	Informasi Physical	informasi IP *trace*	Informasi AODV *trace*

Penjelasan :

1. peristiwa : memuat keterangan berupa *send*, *request*, *drop*, *collision* dan *forward.*
2. Waktu : memuat keterangan waktu saat kejadian berlangsung.
3. Node : memuat keterangan tempat terjadinya peristiwa.
4. *Layer* : memuat keterangan pada layer mana kejadian berlangsung. Contoh: RTR, IFQ, MAC dan AGT.
5. Unique ID : identitas paket.
6. Tipe paket : tipe paket yang dikirimkan. Dapat berupa protokol *transfer* (CBR/UDP) ataupun protokol *routing* (AODV).
7. Ukuran paket : ukuran paket dalam satuan *bytes.*
8. Informasi *Physical* :

- kolom pertama memuat waktu pengiriman.
- Kolom kedua memuat keterangan MAC *address* tujuan
- Kolom ketiga memuat keterangan MAC *address* sumber.
- “800” menandakan bahwa IP paket berjalan pada jaringan Ethernet.

9. Informasi IP *trace* :
   - kolom pertama memuat keterangan IP sumber beserta portnya.
   - kolom kedua memuat keterangan IP tujuan beserta portnya.
   - kolom ketiga memuat keterangan TTL.
   - kolom keempat memuat keterangan *nexthops.*
10. Informasi AODV *trace* :
   - kolom pertama memuat keterangan RREQ dengan ID "0x2"
   - kolom kedua memuat keterangan jumlah *hop*.
   - Kolom ketiga memuat keterangan *broadcast ID.*
   - Kolom keempat memuat keterangan IP tujuan dan *Sequence number*.
   - Kolom memuat keterangan IP sumber dan *Sequence number.*
   - Kolom memuat keterangan bahwa ini merupakan paket RREQ

4.3.1 Pengumpulan dan Pengambilan Data *Packet Delivery Ratio*

Proses pengumpulan dan pengambilan data Packet Delivery Ratio dilakukan melalui pengolahan data *tracefile*. Data pada *tracefile* dipilah untuk menentukan total paket yang terkirim dan paket yang diterima, kemudian membagi paket yang diterima dengan paket yang terkirim. Kode Program 4.9 merupakan implementasi *packet delivery ratio*:

Kode Program 4.9 Program AWK Packet Delivery Ratio

Kode Program 9 : Program AWK Packet Delivery Ratio

```
1.  BEGIN {
2.   sent=0;
3.   received=0;
4.   drop=0;
5.   }{
6.     if($1=="s" && $4=="AGT")
7.     { sent++;}
8.      else if($1=="r" && $4=="AGT")
9.       {received++;}
10.     else if($1=="D" && $4=="AGT")
11.      {drop++;}
12.  }END{
13.  print "Generated packet= ",sent;
14.  print "Packet Received= ",received;
15.  print "Packet Delivery Ratio=  " (received/sent) * 100 "%";
```

```
16.  print "Total Dropped Packets = " drop;
17.  }
```

Penjelasan :

1. Nomor 1-3 : inisiasi awal program serta membuat variabel *sent* dan *received* yang berfungsi untuk menyimpan jumlah paket terkirim dan diterima.
2. Nomor 5-6 : kondisi jika nilai kolom pertama bernilai "s" dan juga kolom keempat bernilai "AGT" maka nilai *sent* bertambah 1.
3. Nomor 7-8 : kondisi jika nilai kolom pertama bernilai "r" dan juga kolom keempat bernilai "AGT" maka nilai *received* bertambah 1.
4. Nomor 10-11 : kondisi jika nilai kolom pertama bernilai "D" dan juga kolom keempat bernilai "AGT" maka nilai *drop* bertambah 1.
5. Nomor 12 : akhir dari program
6. Nomor 13 : mencetak "*Generated packet* =" beserta nilai *sent*
7. Nomor 14 : mencetack "*Received Packet* =" beserta nilai *received*
8. Nomor 15 : mencetak "*Packet Delivery Ratio* =" beserta nilai *received* dibagi dengan sent x 100%
9. Nomor 16 : mencetak "*Total Dropped Packets* =" beserta nilai *drop*.

Berikut merupakan Gambar 4.7 setelah program AWK berhasil dijalankan.

```
ajisuryo@SKRIPSI:~/ns-allinone-2.35$ gawk -f pdr.awk AODV50sybil2.tr


GeneratedPackets = 19980
ReceivedPackets = 19833
Packet Delivery Ratio = 99.2643%
Total Dropped Packets = 71
```

Gambar 4.7 Hasil eksekusi program pdr.awk

4.3.2 Pengumpulan dan Pengambilan Data *End to End Delay*

Proses pengumpulan dan pengambilan data *end to end delay* dilakukan melalui pengolahan data *tracefile*. Data pada *tracefile* dipilah untuk menentukan jumlah total waktu pengiriman paket dan jumlah total pengiriman paket, kemudian membagi total waktu dengan total pengiriman paket. Kode Program 4.10 merupakan program *end to end delay*:

Kode Program 4.10 Program AWK End-to-End Delay

Kode Program 10 : Program AWK End-to-End Delay

```
1.  BEGIN {
2.          seqno= -1;
3.          count = 0;
4.  }{
5.      if ($4 == "AGT" && $1 == "s" && seqno < $6) {
```

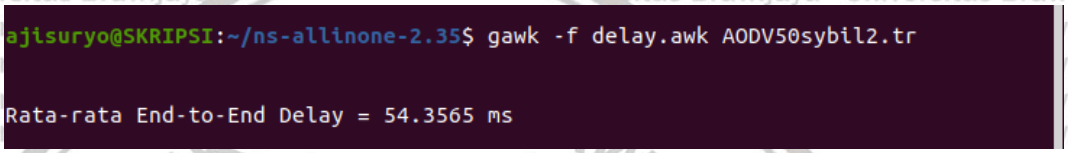
```
6.      seqno = $6; }
7.      if($4 == "AGT" && $1 == "s") {
8.      start_time[$6] = $2; }
9.      else if(($7 == "cbr") && ($1 == "r")) {
10.     end_time[$6] = $2; }
11.     else if($1 == "D" && $7 == "cbr") {
12.     end_time[$6] = -1; }
13.  }END {
14.        for(i=0; i<=seqno; i++) {
15.           if(end_time[i] >=0) {
16.           delay[i] = end_time[i] - start_time[i];
17.           count++;
18.           }
19.           else
20.           {
21.           delay[i] = -1; } }
22.           for(i=0; i<count; i++) {
23.              if(delay[i] >= 0) {
24.              n_to_n_delay = n_to_n_delay + delay[i]; } }
25.              n_to_n_delay = n_to_n_delay/count;
26.  print "Rata-rata End-to-End Delay = " n_to_n_delay * 1000 " ms";
```

Penjelasan :

1. Nomor 1-3 : inisiasi program serta membuat variabel seqno dan count untuk menyimpan *sequence number*, juga *count* untuk menghitung jumlah pengiriman paket.
2. Nomor 5-6 : kondisi jika kolom keempat bernilai “AGT”, kolom pertama bernilai “s” dan nilai seqno lebih kecil dari kolom keenam. Maka nilai seqno diperbaharui dengan nilai yang ada pada kolom keenam.
3. Nomor 7-8 : kondisi jika kolom keempat bernilai “AGT” dan kolom pertama bernilai “s”. maka nilai array *start_time* akan diperbaharui dengan nilai pada kolom kedua sesuai urutan kolom keenam.
4. Nomor 9-10 : kondisi jika kolom ketujuh bernilai “cbr” dan kolom pertama bernilai “r”. maka nilai array *end_time* akan diperbaharui dengan nilai pada kolom kedua sesuai urutan kolom keenam.
5. Nomor 9-10 : kondisi jika kolom ketujuh bernilai “cbr” dan kolom pertama bernilai “D”. maka nilai array *end_time* akan diperbaharui dengan nilai -1. ini sebagai kondisi jika paket gagal sampai ke tujuan.
6. Nomor 13 : akhir dari program
7. Nomor 14 : pengulangan selama nilai i lebih kecil dari nilai seqno, maka perhitungan akan terus berjalan.

8. Nomor 15-17 : kondisi jika nilai *end_time* lebih dari 0 (paket telah diterima), maka *delay* akan bernilai dari hasil pengurangan *end_time* oleh *start_time*. Penambahan jumlah *count* + 1.
9. Nomor 19-21 : kondisi lainnya, maka nilai *delay* akan menjadi -1 (paket gagal sampai ke tujuan).
10. Nomor 22 : pengulangan selama nilai i lebih kecil dari nilai *count*, maka perhitungan akan terus berjalan.
11. Nomor 23-24 : kondisi jika nilai *delay* lebih dari 0, maka nilai total *end to end delay* akan dijumlahkan dengan nilai *delay*.
12. Nomor 25 : perhitungan rata-rata *end to end delay* dibagi dengan *count.*

Berikut merupakan Gambar 4.8 setelah program AWK berhasil dijalankan



Gambar 4.8 Hasil eksekusi program delay.awk

4.3.3 Pengumpulan dan Pengambilan Data *Normalized Routing Load*

Proses pengumpulan dan pengambilan data *Normalized Routing Load* dilakukan melalui pengolahan data *tracefile*. Data pada *tracefile* dipilah untuk menentukan jumlah total paket diterima dan jumlah total *routing* paket, kemudian membagi total *routing* paket dengan total paket yang diterima. Kode Program 4.11 merupakan implementasi program *Normalized Routing Load*.

Kode Program 4.11 Kode Program AWK Normalized Routing Load

Kode Program 11 : Kode Program AWK *Normalized Routing Load*

```
1.  BEGIN{
2.  recvd = 0;
3.  rt_pkts = 0;
4.  }
5.  {
6.  if (( $1 == "r") && ( $7 == "cbr") && ( $4=="AGT" )) recvd++;
7.  if (($1 == "s" || $1 == "f") && $4 == "RTR" && ($7 =="AODV")
8.  rt_pkts++;
9.  }
10. END{
11. printf("\n");
12. printf("data packets received = %.3f\n", recvd);
13. printf("routing packets received = %.3f\n", rt_pkts);
```

```
14.  printf("Normalized Routing Load = %.3f\n", rt_pkts/recvd);
15.  printf("\n");
16.  }
```

Penjelasan :

1. Nomor 1-3 : inisiasi program serta variabel recvd dan rt_pkts
2. Nomor 6 : kondisi jika kolom pertama bernilai “r” dan kolom ketujuh bernilai “cbr” dan kolom keempat bernilai “AGT”, maka nilai recvd bertambah +1.
3. Nomor 7 : kondisi jika kolom pertama bernilai “s” atau “f” dan kolom keempat bernilai “RTR” dan kolom ketujuh bernilai “AODV”, maka nilai rt_pkts bertambah +1.
4. Nomor 10 : akhir dari program.
5. Nomor 11-16 : mencetak paket yang diterima, routing paket yang diterima dan hasil *Normalized Routing Load.*

Berikut merupakan Gambar 4.9 setelah program AWK berhasil dijalankan.

```
ajisuryo@SKRIPSI:~/ns-allinone-2.35$ gawk -f normalizedroutingload.awk AODV50syb
il2.tr

data packets received = 19833.000
routing packets received = 3002.000
Normalized Routing Load = 0.151
```

Gambar 4.9 Hasil eksekusi program normalizedroutingload.awk

4.3.4 Pengumpulan dan Pengambilan Data *Throughput*

Proses pengumpulan dan pengambilan data *throughput* dilakukan melalui pengolahan data *tracefile*. Data pada *tracefile* dipilah untuk menentukan jumlah total paket diterima dan jumlah total *routing* paket, kemudian membagi total *routing* paket dengan total paket yang diterima. Kode Program 4.12 merupakan implementasi program *Throughput*.

Kode Program 4.12 Kode Program AWK Throughput

Kode Program 12 : Kode Program AWK Throughput

```
1.  BEGIN{
2.          recv_size=0
3.          sTime=1e6
4.          spTime=0
5.          NumOfRecd=0
6.  }
7.  {
8.  event=$1
9.  packet=$4
```

```
10.  time=$2
11.  pkt_id=$6
12.  packet_size=$8
13.  if(packet=="AGT" && sendTime[pkt_id] == 0 &&(event == "+" || event
14.  == "s")){
15.          if(time<sTime){
16.                  sTime=time
17.          }
18.          sendTime[pkt_id]=time}
19.  if (packet=="AGT" && event=="r"){
20.          if(time>spTime){
21.                  spTime=time
22.          }
23.          recvTime[pkt_id] = time
24.          recv_size=recv_size + packet_size
25.          NumOfRecd = NumOfRecd + 1}}
26.  END{
27.          printf("start time %d\n",sTime)
28.          printf("stop time %d\n",spTime)
29.          printf("received packet time %d\n",NumOfRecd)
30.          printf("average throughput in kbps
31.  %f\n",((recv_size/(spTime-sTime)*8)/1000))}
```

Penjelasan :

1. Nomor 1-7 : inisiasi program serta variabel recv_size, sTime, spTime dan NumOfRecd.
2. Nomor 8-12 : penentuan nilai variabel event, packet, time, pkt_id, packet_size dengan kolom pada *tracefile*.
3. Nomor 13-18 : kondisi untuk menentukan waktu awal pengiriman
4. Nomor 19-25 : kondisi untuk menentukan waktu akhir pengiriman serta menghitung total *payload* dan juga banyaknya pengiriman
5. Nomor 26 : akhir program
6. Nomor 27 : mencetak waktu awal pengiriman data
7. Nomor 28 : mencetak waktu akhir pengiriman data
8. Nomor 29 : mencetak banyaknya pengiriman data
9. Nomor 30-31 : hasil *throughput*

Berikut merupakan Gambar 4.10 setelah program AWK berhasil dijalankan.

```
ajisuryo@SKRIPSI:~/ns-allinone-2.35$ gawk -f a_throughput.awk AODV50sybil2.tr
start time 1
stop time 999
received packet time 19833
average throughput in kbps 84.548900
```

Gambar 4.10 Hasil eksekusi program a_throughput.awk

4.3.5 Pengumpulan dan Pengambilan Data Tabel *Routing*

Proses pencetakan table *routing* dilakukan dengan menambahkan baris kode pada aodv.h dan aodv.cc. Kode Program 4.13 merupakan implementasi rt_print pada aodv.h

Kode Program 4.13 Kode rt_print pada aodv.h

Kode Program 13 : Kode rt_print pada aodv.h

```
1.  Void rt_print (nsaddr_t nodeid);
```

Penjelasan : menambahkan fungsi rt_print dengan parameter nodeid bertipe nsaddr_t pada *file* aodv.h

Kode Program 4.14 merupakan implementasi rt_print pada aodv.c

Kode Program 4.14 Kode rt_print pada aodv.cc

Kode Program 14 : Kode rt_print pada aodv.cc

```
1.  void  AODV::rt_print(nsaddr_t nodeid) {
2.  FILE *fp;
3.  aodv_rt_entry *rt;
4.  fp = fopen("route_table.txt", "a");
5.  fprintf(fp,"Node id \t time \t destination \t nexthop \t hop count
6.  \t seq.no\n");
7.  for (rt=rtable.head();rt; rt = rt->rt_link.le_next) {
8.  fprintf(fp,"NODE: %i \t %f \t %i \t %i \t \t %i \t \t %i \n", nodeid,
9.  CURRENT_TIME, rt->rt_dst, rt->rt_nexthop, rt->rt_hops, rt-
10. >rt_seqno);
11. }
12. fclose(fp);
13. }
…   …
14. void
15. AODV::recvReply(Packet *p) {
…   …
16  if (ih->daddr() == index) {
17          rt_print(index);
```

Penjelasan :

1. Nomor 1 : mendefinisikan fungsi rt_print berparameter nodeid

2. Nomor 2 : inisiasi variable fp bertipe FILE
3. Nomor 3 : inisiasi variable rt bertipe aodv_rt_entry
4. Nomor 4 : membuka *file* "route_table.txt"
5. Nomor 5-6 : mencetak kolom *Node id, time, destination, nexthop, hop count* dan *sequence number.*
6. Nomor 7-8 : pengulangan untuk setiap tabel *routing* yang tersimpan pada *node* untuk mencetak data ke dalam *file* "route_table.txt"
7. Nomor 12 : menutup variable fp.
8. Nomor 14-17: pemanggilan fungsi rt_print ketika menerima paket RREP hanya ketika nilai IP *header destination* address sama dengan nilai index, kondisi ini menyatakan bahwa *node* ini merupakan *node* sumber.

Kemudian tabel *routing* akan tersimpan pada direktori Network Simulator 2 dengan nama "*routing_table.txt*". Gambar 4.11 merupakan tampilan dari *file routing_table.txt* :

Node id	time	destination	nexthop	hop count	seq.no
NODE: 27	972.647283	33	0	255	5
NODE: 27	972.647283	38	0	255	7
NODE: 27	972.647283	44	0	255	11
NODE: 27	972.647283	2	0	255	7
NODE: 27	972.647283	19	0	255	7
NODE: 27	972.647283	18	0	255	5
NODE: 27	972.647283	26	0	255	9
NODE: 27	972.647283	11	0	255	5
NODE: 27	972.647283	10	0	255	7
NODE: 27	972.647283	28	7	4	106

Gambar 4.11 hasil dari fungsi *rt_print*.

BAB 5 PENGUJIAN DAN PEMBAHASAN

Dalam bab ini akan dilakukan pembahasan mengenai hasil penelitian berikut analisis pada protokol AODV sesuai hasil yang didapatkan dari skenario pengujian. Analisis akan dilakukan terhadap hasil pengujian variasi jumlah *node* dan jumlah *node* penyerang. Dari hasil analisis akan diketahui bagaimana pengaruh serangan *Sybil* terhadap kinerja protokol AODV berdasarkan parameter *packet delivery ratio, end to end delay, normalized routing load* dan *routing table node* sumber.

5.1 Packet Delivery Ratio

5.1.1 Serangan *Sybil* pada Skenario 20 *node*

Dari hasil pengujian program *packet delivery ratio* yang telah dilakukan menggunakan program AWK pada *tracefile* skenario 20 *node* maka diperoleh hasil pada Tabel 5.1 :

Tabel 5.1 Tabel hasil pengujian *Packet Delivery Ratio* pada skenario 20 *node*

Protokol	Jumlah node	Jumlah Node Penyerang					
AODV	20	0	2	4	6	8	10
		76,07%	78,58%	75,03%	73,36%	73,44%	74,51%

Pada saat pengujian protokol tanpa *node* penyerang hasil yang didapat mencapai 76,07%, untuk 2 *node* penyerang mencapai 78,58%, untuk 4 *node* penyerang mencapai 75,03%, untuk 6 *node* penyerang mencapai 73,36%, untuk 8 *node* penyerang mencapai 73,44% dan untuk 10 *node* penyerang mencapai 74,51%.

5.1.2 Serangan *Sybil* pada Skenario 50 *node*

Dari hasil pengujian program *packet delivery ratio* yang telah dilakukan menggunakan program AWK pada *tracefile* skenario 50 *node* maka diperoleh hasil pada Tabel 5.2 :

Tabel 5.2 Tabel hasil pengujian *Packet Delivery Ratio* pada skenario 50 *node*

Protokol	Jumlah node	Jumlah Node Penyerang					
AODV	50	0	2	4	6	8	10
		99,26%	99,16%	97,98%	96,69%	92,66%	91,78%

Pada saat pengujian protokol tanpa *node* penyerang hasil yang didapat mencapai 99,26%, untuk 2 *node* penyerang mencapai 99,16%, untuk 4 *node*

penyerang mencapai 97,98%, untuk 6 *node* penyerang mencapai 96,69%, untuk 8 *node* penyerang mencapai 92,66% dan untuk 10 *node* penyerang mencapai 91,78%.

5.1.3 Serangan *Sybil* pada Skenario 100 *node*

Dari hasil pengujian program *packet delivery ratio* yang telah dilakukan menggunakan program AWK pada *tracefile* skenario 100 *node* maka diperoleh hasil pada Tabel 5.3 :

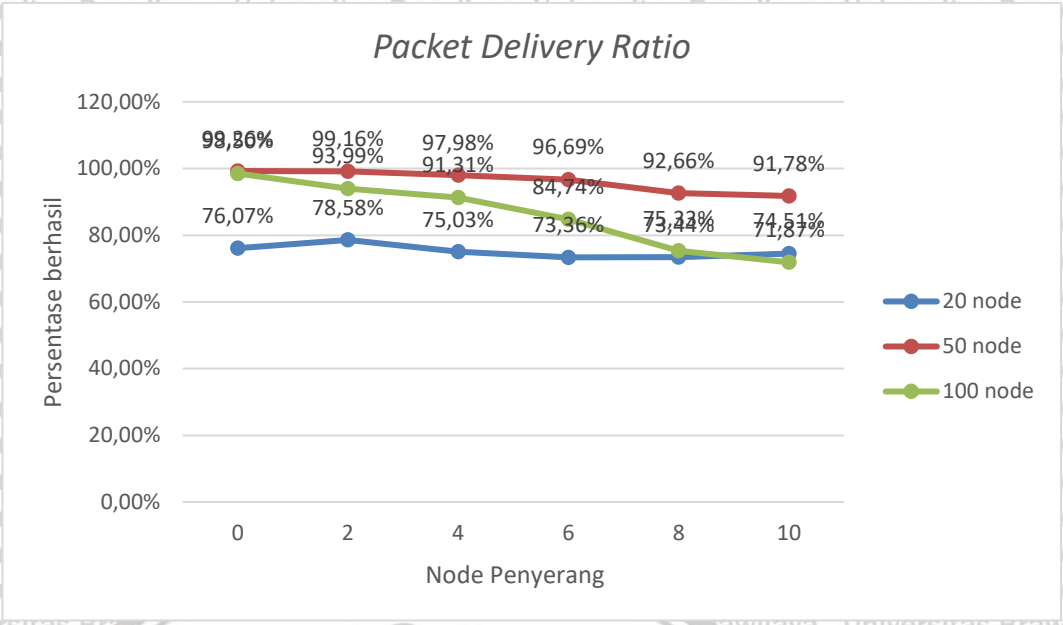
Tabel 5.3 Tabel hasil pengujian *Packet Delivery Ratio* pada skenario 100 *node*

Protokol	Jumlah node	Jumlah Node Penyerang					
AODV	100	0	2	4	6	8	10
		98,50%	93,99%	91,31%	84,74%	75,33%	71,87%

Pada saat pengujian protokol tanpa *node* penyerang hasil yang didapat mencapai 98,50%, untuk 2 *node* penyerang mencapai 93,99%, untuk 4 *node* penyerang mencapai 91,31%, untuk 6 *node* penyerang mencapai 84,74%, untuk 8 *node* penyerang mencapai 75,33% dan untuk 10 *node* penyerang mencapai 71,87%.

5.1.4 Pembahasan *Packet Delivery Ratio*

Dari hasil yang telah diperoleh maka dapat dibentuk grafik perbandingan *packet delivery ratio* antara skenario 20 *node,* 50 *node* dan 100 *node.* Gambar 5.1 merupakan grafik perbandingan *packet delivery ratio* antara skenario 20 *node*, 50 *node* dan 100 *node*.



Gambar 5.1 Grafik perbandingan *packet delivery ratio* antara skenario 20 *node*, 50 *node* dan 100 *node*.

Gambar 5.1 menunjukkan bahwa secara garis besar serangan *Sybil* menurunkan persentase keberhasilan pengiriman data. Pada skenario 20 *node* terjadi fluktuasi sementara, hal ini disebabkan oleh kerenggangan antar *node* yang mengakibatkan seringnya terjadi *packet drop*. Pada skenario 50 *node* dan 100 *node* persentase pengiriman turun secara konsisten dan dapat diambil kesimpulan bahwa semakin padat topologi dan jumlah *node* penyerang maka serangan *Sybil* akan semakin buruk dampaknya.

Pada skenario 20 meskipun jumlah *node* penyerang mencapai 50% dari jumlah total *node* nilai *packet delivery ratio* tidak berubah secara signifikan dari nilai ketika tanpa adanya *node* penyerang. Hal ini disebabkan karena beban komunikasi masih dapat ditanggung. Namun pada skenario 50 dan skenario 100 beban komunikasi bertambah karena terjadinya peningkatan jumlah paket yang terkirim.

5.2 End to End Delay

5.2.1 Serangan *Sybil* pada Skenario 20 *node*

Dari hasil pengujian program *end to end delay* yang telah dilakukan menggunakan program AWK pada *tracefile* skenario 20 *node* maka diperoleh hasil pada Tabel 5.4 :

Tabel 5.4 Tabel hasil pengujian *End to End Delay* pada skenario 20 *node*

Protokol	Jumlah node	Jumlah Node Penyerang					
AODV	20	0	2	4	6	8	10

		69,35 ms	83,92 ms	110,11 ms	118,09 ms	153,44 ms	94,51 ms

Pada saat pengujian protokol tanpa *node* penyerang *end to end delay* yang didapat mencapai 69,35 ms untuk 2 *node* penyerang mencapai 83,92 ms untuk 4 *node* penyerang mencapai 110,10 ms untuk 6 *node* penyerang mencapai 118,09 ms untuk 8 *node* penyerang mencapai 153,44 ms dan untuk 10 *node* penyerang mencapai 94,51 ms.

5.2.2 Serangan *Sybil* pada Skenario 50 *node*

Dari hasil pengujian program *end to end delay* yang telah dilakukan menggunakan program AWK pada *tracefile* skenario 50 *node* maka diperoleh hasil pada Tabel 5.5:

Tabel 5.5 Tabel hasil pengujian *End to End Delay* pada skenario 50 *node*

Protokol	Jumlah node	Jumlah Node Penyerang					
AODV	50	0	2	4	6	8	10
		54,35 ms	60,92 ms	90,11 ms	128,59 ms	171,38 ms	171,91 ms

Pada saat pengujian protokol tanpa *node* penyerang *end to end delay* yang didapat mencapai 54,35 ms untuk 2 *node* penyerang mencapai 60,92 ms untuk 4 *node* penyerang mencapai 90,11 ms untuk 6 *node* penyerang mencapai 128,59 ms untuk 8 *node* penyerang mencapai 171,38 ms dan untuk 10 *node* penyerang mencapai 171,91 ms.

5.2.3 Serangan *Sybil* pada Skenario 100 *node*

Dari hasil pengujian program *end to end delay* yang telah dilakukan menggunakan program AWK pada *tracefile* skenario 100 *node* maka diperoleh hasil pada Tabel 5.6:

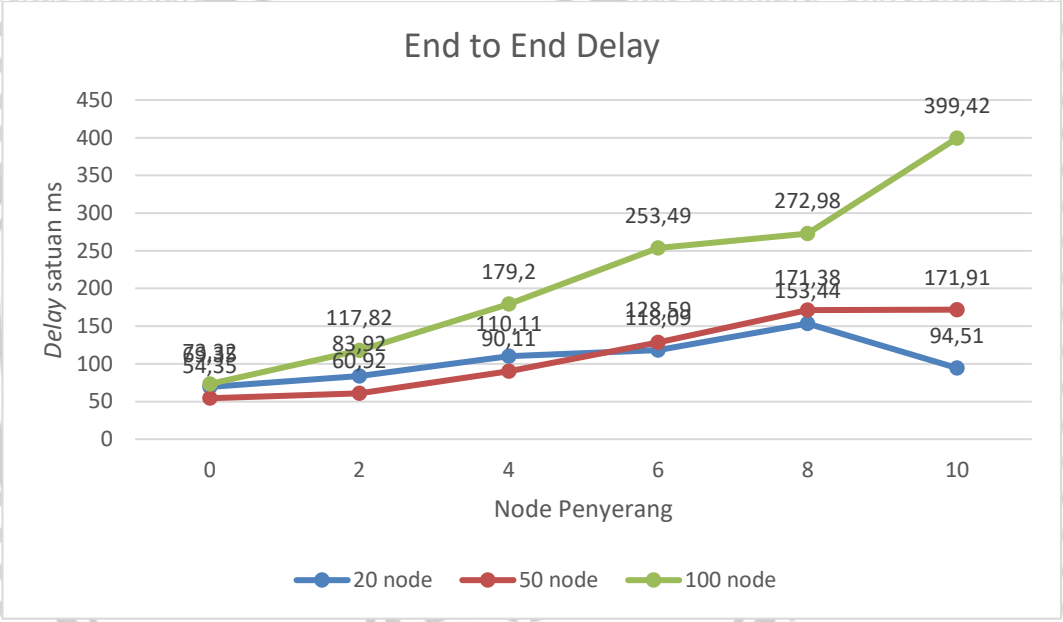
Tabel 5.6 Tabel hasil pengujian *End to End Delay* pada skenario 100 *node*

Protokol	Jumlah node	Jumlah Node Penyerang					
AODV	100	0	2	4	6	8	10
		73,32 ms	117,82 ms	179,20 ms	253,49 ms	272,98 ms	399,42 ms

Pada saat pengujian protokol tanpa *node* penyerang *end to end delay* yang didapat mencapai 73,32 ms untuk 2 *node* penyerang mencapai 117,82 ms untuk 4 *node* penyerang mencapai 179,20 ms untuk 6 *node* penyerang mencapai 253,49 ms untuk 8 *node* penyerang mencapai 272,98 ms dan untuk 10 *node* penyerang mencapai 399,42 ms.

5.2.4 Pembahasan *End to End Delay*

Dari hasil yang telah diperoleh maka dapat dibentuk grafik perbandingan *End to End Delay* antara skenario 20 *node,* 50 *node* dan 100 *node*. Gambar 5.2 merupakan grafik perbandingan *End to End Delay* antara skenario 20 *node*, 50 *node* dan 100 *node*.



Gambar 5.2 Grafik perbandingan *end to end delay* antara skenario 20 *node,* 50 *node* dan 100 *node*.

Gambar 5.2 menunjukkan bahwa secara garis besar serangan *Sybil* waktu *end to end delay* pengiriman data bertambah. Pada skenario 20 *node* terjadi fluktuasi sementara, hal ini disebabkan oleh kerenggangan antar *node* yang mengakibatkan seringnya terjadi *packet drop*. Pada skenario 50 *node* dan 100 *node* nilai delay bertambah secara konsisten. Perubahan nilai *delay* terjadi karena paket yang dikirim oleh *node* penyerang masuk kedalam jaringan dan mengakibatkan antrian pada paket normal. Dapat diambil kesimpulan bahwa semakin padat topologi dan jumlah *node* penyerang maka serangan *Sybil* akan semakin buruk dampaknya.

5.3 *Normalized Routing Load*

5.3.1 Serangan *Sybil* pada Skenario 20 *node*

Dari hasil pengujian program *normalized routing load* yang telah dilakukan menggunakan program AWK pada *tracefile* skenario 20 *node* maka diperoleh hasil pada Tabel 5.7:

Tabel 5.7 Tabel hasil pengujian *normalized routing load* pada skenario 20 *node*.

Protokol	Jumlah node	Jumlah Node Penyerang					
AODV	20	0	2	4	6	8	10
		0,14	2,03	3,70	6,99	8,11	8,38

Pada saat pengujian protokol tanpa *node* penyerang *normalized routing load* yang didapat mencapai 0,14 untuk 2 *node* penyerang mencapai 2,03 untuk 4 *node* penyerang mencapai 3,7 untuk 6 *node* penyerang mencapai 6,99 untuk 8 *node* penyerang mencapai 8,11 dan untuk 10 *node* penyerang mencapai 8,38.

5.3.2 Serangan *Sybil* pada Skenario 50 *node*

Dari hasil pengujian program *normalized routing load* yang telah dilakukan menggunakan program AWK pada *tracefile* skenario 50 *node* maka diperoleh hasil pada Tabel 5.8:

Tabel 5.8 Tabel hasil pengujian *normalized routing load* pada skenario 50 *node*.

Protokol	Jumlah node	Jumlah Node Penyerang					
AODV	50	0	2	4	6	8	10
		0,15	3,78	7,51	11,15	14,94	18,21

Pada saat pengujian protokol tanpa *node* penyerang *normalized routing load* yang didapat mencapai 0,15 untuk 2 *node* penyerang mencapai 3,73 untuk 4 *node* penyerang mencapai 7,51 untuk 6 *node* penyerang mencapai 11,15 untuk 8 *node* penyerang mencapai 14,94 dan untuk 10 *node* penyerang mencapai 18,21.

5.3.3 Serangan *Sybil* pada Skenario 100 *node*

Dari hasil pengujian program *normalized routing load* yang telah dilakukan menggunakan program AWK pada *tracefile* skenario 100 *node* maka diperoleh hasil pada Tabel 5.9:

Tabel 5.9 Tabel hasil pengujian *normalized routing load* pada skenario 100 *node*.

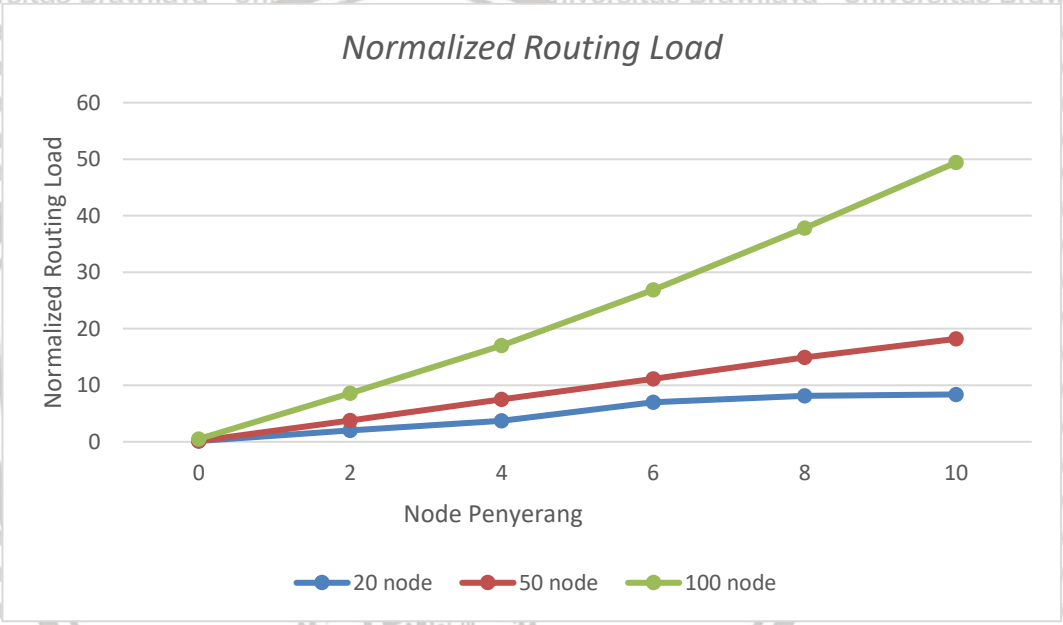
Protokol	Jumlah node	Jumlah Node Penyerang					
AODV	100	0	2	4	6	8	10

		0,5	8,6	17,02	26,86	37,8	49,4

Pada saat pengujian protokol tanpa *node* penyerang *normalized routing load* yang didapat mencapai 0,5 untuk 2 *node* penyerang mencapai 8,6 untuk 4 *node* penyerang mencapai 17,02 untuk 6 *node* penyerang mencapai 26,86 untuk 8 *node* penyerang mencapai 37,8 dan untuk 10 *node* penyerang mencapai 49,4.

5.3.4 Pembahasan *Normalized Routing Load*

Dari hasil yang telah diperoleh maka dapat dibentuk grafik perbandingan *Normalized Routing Load* antara skenario 20 *node,* 50 *node* dan 100 *node*. Gambar 5.3 merupakan grafik perbandingan *End to End Delay* antara skenario 20 *node*, 50 *node* dan 100 *node*.



Gambar 5.3 Grafik perbandingan *normalized routing load* antara skenario 20 *node,* 50 *node* dan 100 *node*.

Gambar 5.3 menunjukkan bahwa secara garis besar serangan *Sybil* membuat nilai *normalized routing load* terus bertambah. Pada skenario 20 *node* ,50 *node* dan 100 *node* nilai *normalized routing load* bertambah secara konsisten. Hal ini disebabkan oleh meningkatnya jumlah paket rute yang dikirim oleh serangan *Sybil.* Ini menandakan bahwa efektivitas proses perutean berkurang dan dapat diambil kesimpulan bahwa semakin padat topologi dan jumlah *node* penyerang maka serangan *Sybil* akan semakin buruk dampaknya.

5.4 Throughput

5.4.1 Serangan *Sybil* pada Skenario 20 *node*

Dari hasil pengujian program *throughput* yang telah dilakukan menggunakan program AWK pada *tracefile* skenario 20 *node* maka diperoleh hasil pada Tabel 5.10 :

Tabel 5.10 Tabel hasil pengujian *throughput* pada skenario 20 *node*

Protokol	Jumlah node	Jumlah Node Penyerang					
AODV	20	0	2	4	6	8	10
		64,89 kbps	67,07 kbps	64,09 kbps	62,47 kbps	62,77 kbps	63,70 kbps

Pada saat pengujian protokol tanpa *node* penyerang, hasil yang diperoleh mencapai 64,89 kbps, untuk 2 *node* penyerang mencapai 67,07 kbps, untuk 4 *node* penyerang mencapai 64,09 kbps, untuk 6 *node* penyerang mencapai 62,47 kbps, untuk 8 *node* penyerang mencapai 62,77 kbps dan untuk 10 *node* penyerang mencapai 63,70 kbps.

5.4.2 Serangan *Sybil* pada Skenario 50 *node*

Dari hasil pengujian program *throughput* yang telah dilakukan menggunakan program AWK pada *tracefile* skenario 50 *node* maka diperoleh hasil pada Tabel 5.11 :

Tabel 5.11 Tabel hasil pengujian *throughput* pada skenario 50 *node*

Protokol	Jumlah node	Jumlah Node Penyerang					
AODV	50	0	2	4	6	8	10
		84,54 kbps	84,45 kbps	83,45 kbps	82,36 kbps	79,01 kbps	78,25 kbps

Pada saat pengujian protokol tanpa *node* penyerang hasil yang diperoleh mencapai 84,54 kbps, untuk 2 *node* penyerang mencapai 84,45 kbps, untuk 4 *node* penyerang mencapai 83,45 kbps, untuk 6 *node* penyerang mencapai 82,36 kbps, untuk 8 *node* penyerang mencapai 79,01 kbps dan untuk 10 *node* penyerang mencapai 78,25 kbps.

5.4.3 Serangan *Sybil* pada Skenario 100 *node*

Dari hasil pengujian program *throughput* yang telah dilakukan menggunakan program AWK pada *tracefile* skenario 100 *node* maka diperoleh hasil pada Tabel 5.12 :

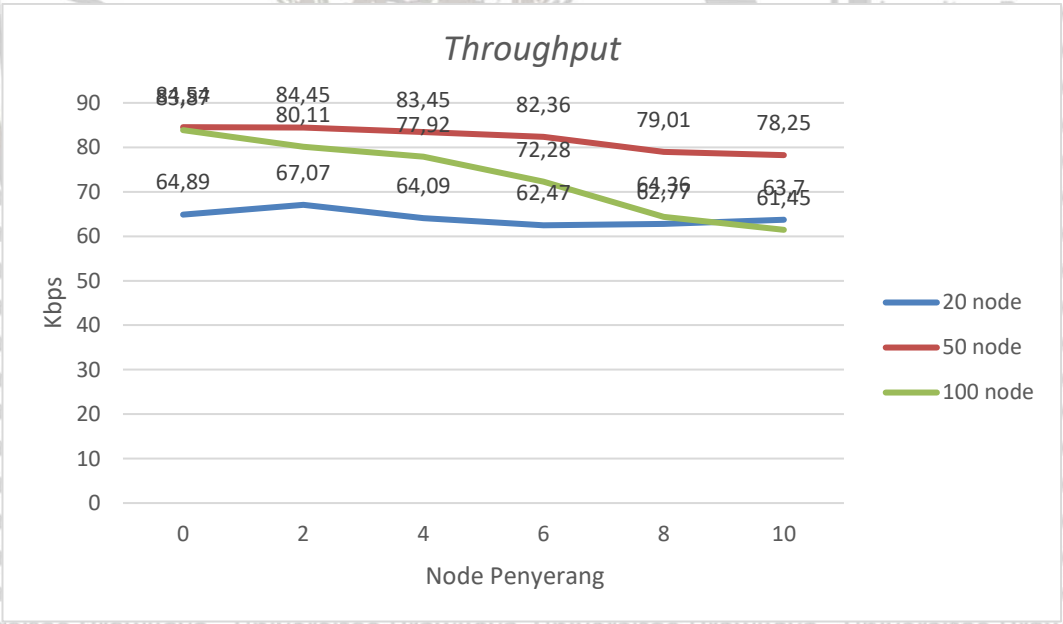
Tabel 5.12 Tabel hasil pengujian *throughput* pada skenario 100 *node*

Protokol	Jumlah node	Jumlah Node Penyerang					
AODV	100	0	2	4	6	8	10
		83,87 kbps	80,11 kbps	77,92 kbps	72,28 kbps	64,36 kbps	61,45 kbps

Pada saat pengujian protokol tanpa *node* penyerang hasil yang didapat mencapai 83,87 kbps, untuk 2 *node* penyerang mencapai 80,11 kbps, untuk 4 *node* penyerang mencapai 77,92 kbps, untuk 6 *node* penyerang mencapai 72,28 kbps, untuk 8 *node* penyerang mencapai 64,36 kbps dan untuk 10 *node* penyerang mencapai 61,45 kbps.

5.4.4 Pembahasan *Throughput*

Dari hasil yang telah diperoleh maka dapat dibentuk grafik perbandingan *throughput* antara skenario 20 *node,* 50 *node* dan 100 *node.* Gambar 5.1 merupakan grafik perbandingan *packet delivery ratio* antara skenario 20 *node*, 50 *node* dan 100 *node*.



Gambar 5.4 Grafik perbandingan *throughput* antara skenario 20 *node,* 50 *node* dan 100 *node*.

Gambar 5.4 menunjukkan bahwa secara garis besar serangan *Sybil* menurunkan nilai throughput. Pada skenario 20 *node* terjadi fluktuasi sementara , hal ini disebabkan oleh kerenggangan antar *node* yang mengakibatkan seringnya terjadi *packet drop*. Pada skenario 50 *node* dan 100 *node* throughput turun secara konsisten. Grafik menunjukkan bahwa *throughput* dan *packet delivery ratio*

memiliki pergerakan yang serupa. Pengiriman paket palsu dari *node* penyerang secara terus menerus akan membebani jaringan sehingga nilai *throughput* akan menurun. Dapat diambil kesimpulan bahwa semakin padat topologi dan jumlah *node* penyerang maka serangan *Sybil* akan semakin efektif.

5.5 Tabel *Routing*

Berdasarkan hasil eksekusi *Network Simulator-2* maka *routing table* dari *node* sumber akan tersimpan pada *file "routing_table.txt"*. data berikut merupakan *routing table* dari *node* sumber terbaru pada akhir simulasi.

5.5.1 Skenario 20 *node*

Dari hasil pencetakan tabel *routing node* sumber yang telah dilakukan menggunakan fungsi *rt_print* pada *file* aodv.cc. Maka diperoleh hasil pada Tabel 5.13 :

Tabel 5.13 jumlah tabel *routing* pada *node* sumber skenario 20 *node*

Protokol	Jumlah node	Jumlah Node Penyerang					
AODV	20	0	2	4	6	8	10
		5	13	19	28	38	43

Tabel 5.13 menunjukkan bahwa jumlah tabel *routing* yang tersimpan pada *node* sumber ketika terdapat 0, 2, 4, 6, 8 dan 10 secara berturut-turut yakni 5, 13, 19, 28, 38 dan 43. Penambahan jumlah rute ini diakibatkan oleh rute palsu yang dibuat *node* penyerang.

5.5.2 Skenario 50 *node*

Dari hasil pencetakan tabel *routing node* sumber yang telah dilakukan menggunakan fungsi *rt_print* pada *file* aodv.cc. Maka diperoleh hasil pada Tabel 5.14 :

Tabel 5.14 jumlah tabel *routing* pada *node* sumber skenario 50 *node*

Protokol	Jumlah node	Jumlah Node Penyerang					
AODV	50	0	2	4	6	8	10
		10	13	27	28	34	44

Tabel 5.14 menunjukkan bahwa jumlah tabel *routing* yang tersimpan pada *node* sumber ketika terdapat 0, 2, 4, 6, 8 dan 10 secara berturut-turut yakni 10, 13, 27, 28, 34 dan 44. Penambahan jumlah rute ini diakibatkan oleh rute palsu yang dibuat *node* penyerang.

5.5.3 Skenario 100 node

Dari hasil pencetakan tabel *routing node* sumber yang telah dilakukan menggunakan fungsi *rt_print* pada *file* aodv.cc. Maka diperoleh hasil pada Tabel 5.15 :

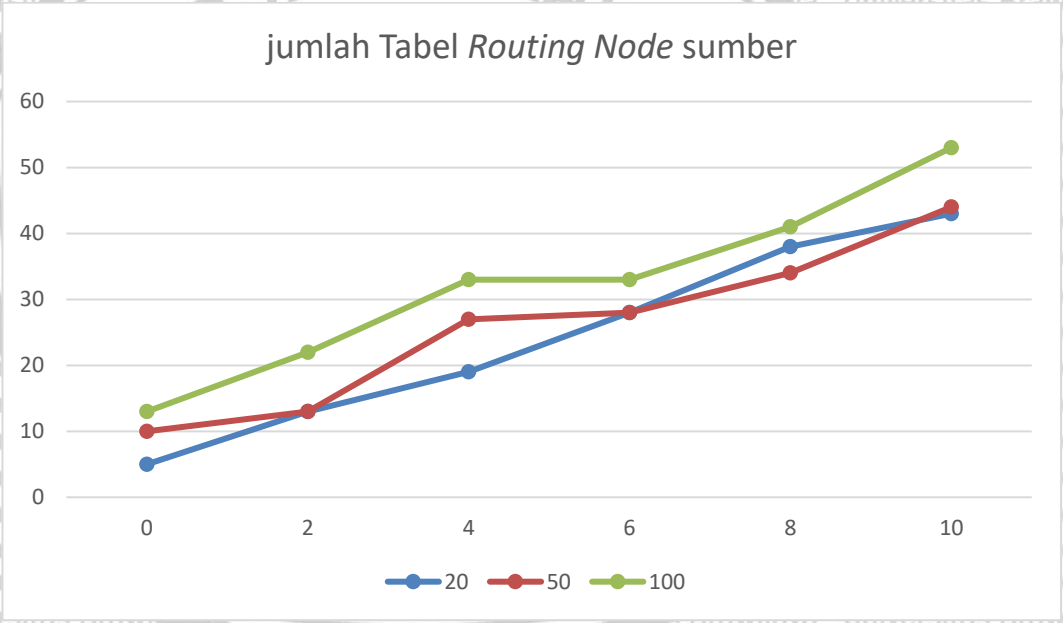
Tabel 5.15 jumlah tabel *routing* pada *node* sumber skenario 100 *node*

Protokol	Jumlah node	Jumlah Node Penyerang					
AODV	50	0	2	4	6	8	10
		13	22	33	33	41	53

Tabel 5.15 menunjukkan bahwa jumlah tabel *routing* yang tersimpan pada *node* sumber ketika terdapat 0, 2, 4, 6, 8 dan 10 secara berturut-turut yakni 13, 22, 33, 33, 41 dan 53. Penambahan jumlah rute ini diakibatkan oleh rute palsu yang dibuat *node* penyerang.

5.5.4 Pembahasan jumlah Tabel *Routing Node* Sumber

Dari hasil yang telah diperoleh maka dapat dibentuk grafik perbandingan jumlah tabel *routing node* sumber antara skenario 20 *node,* 50 *node* dan 100 *node.* Gambar 5.4 merupakan grafik perbandingan *End to End Delay* antara skenario 20 *node*, 50 *node* dan 100 *node*.



Gambar 5.5 Grafik perbandingan jumlah tabel *routing node* sumber antara skenario 20 *node,* 50 *node* dan 100 *node*.

Gambar 5.5 menunjukkan penambahan jumlah tabel *routing node* sumber. Grafik diatas menunjukkan bahwa serangan *Sybil* memperbanyak jumlah tabel *routing* pada *node* sumber. Pada ketiga skenario terjadi penambahan tabel *routing* yang berbanding lurus dengan jumlah *node* penyerang. Hal ini disebabkan oleh perilaku *node* penyerang yang mengirimkan paket rute palsu dengan identitas lain, kemudian rute palsu akan tersimpan pada tabel *routing node* sumber. Paket rute palsu tersebut dikirim dengan identitas target sebagai tujuan pengiriman.

BAB 6 PENUTUP

6.1 Kesimpulan

Berdasarkan hasil analisis kinerja protokol AODV terhadap serangan *Sybil* dapat disimpulkan bahwa:

1. Kinerja protokol AODV terhadap serangan *Sybil* secara garis besar mengalami penurunan pada parameter *packet delivery ratio* dan *delay*. Tetapi penurunan yang signifikan terjadi pada parameter *normalized routing load*. Hal ini disebabkan oleh *behaviour node* penyerang yang terus-menerus mengirimkan paket *request* dengan identitas palsu. Sehingga terjadi lonjakan pada pengiriman paket *routing* yang tak seharusnya terjadi pada protokol reaktif AODV.
2. Pengiriman identitas palsu dapat terjadi pada protokol AODV karena tidak ada sistem keamanan di dalam jaringan MANET (*Mobile Ad hoc Network*).
3. Pengiriman paket *request* dengan identitas target yang tidak berada di dalam jaringan akan membuat paket tersebut akan diteruskan dengan tiada henti sehingga dapat mengurangi ketersediaan sumber daya jaringan tersebut.

6.2 Saran

Saran yang dapat disampaikan oleh penulis untuk penelitian selanjutnya adalah:

1. Perlu dilakukannya penelitian lebih lanjut dengan protokol *routing* lainnya seperti protokol reaktif DSR maupun proaktif OLSR dan DSDV.
2. Perlu dilakukannya penelitian lebih lanjut dengan mengaplikasikan model energi pada *Network Simulator – 2*.
3. Perlu dilakukannya pengujian serangan *Sybil* dengan taksonomi lain seperti *whitewash*, *indirect attack* dan dapat berganti-ganti identitas secara lebih dinamis.

DAFTAR PUSTAKA

Anuj K. Gupta, H. S. (2011). Review of Various Routing Protocols for MANETs. *International Journal of Information and Electronics Engineering, 1*(3), 251-258.

Dorri, A., Kamel, S. R., & Kheyrkhah, E. (2015). Security Challenges in Mobile Ad Hoc Networks: A Survey. *International Journal of Computer Science & Engineering Survey (IJCSES), 6*(1), 15-29.

Goswami, S., Joardar, S., Das, C. B., & Kar, S. a. (2017). *erformance Analysis of Three Routing Protocols in MANET Using the NS-2 and ANOVA Test with Varying Speed of Nodes.* IntechOpen.

Harahap, E. H. (2014). Analisis Performansi Protokol AODV (Ad Hoc On Demand Distance Vector) Dan DSR (Dynamic Source Routing) Terhadap Active Attack Pada MANET (Mobile Ad Hoc Network) Ditinjau Dari QoS (Quality of Service) Jaringan. *e-Proceeding of Engineering, 1*, 118-125.

Hoebeke, J., Moerman, I., Dhoedt, B., & Demeester, P. (2004). An Overview of Mobile Ad Hoc Network : Applications and Challenges. *Journal- Communications Network, 3*(3), 60-66.

Issariyakul, T. (2012). *Introduction to Network Simulator 2 (NS 2).* Springer, Boston, MA.

Kaur, S. G. (2014). Detection and Optimisation Techniques against Sybil Attack on MANET. *International Journal of Advanced Research in Computer Science and Software Engineering, 4*(8), 369-375.

Khirasaraya, H. R. (2013). Simulation Study of Jellyfish Attack in MANET using AODV Routing Protocol. *Journal of Information, knowledge and research in computer engineering, 2*(2), 344-347.

Kumar, S., Goyal, M., Goyal, D., & Poonia, R. C. (2017). Routing Protocols and Security Issues in MANET. *2017 International Conference on Infocom Technologies and Unmanned Systems (ICTUS'2017)*, 818-824.

Kumar, T. L. (2016). MANET: Security and Challenges. *IJCSIT, 7*(0975-9646), 2381-2384.

saha, H. N., Bhattacharyya, D. D., Banerjee, D. P., Bhattacharyya, A., Banerjee, A., & Bose, D. (2012). Study of Different Attacks In MANET With its Detection & Mitigation Schemes. *International Journal of Advanced Engineering Technology, 3*(1), 383-388.

Sinha, S. A. (2013). THE SYBIL ATTACK IN MOBILE ADHOC NETWORK: ANALYSIS AND DETECTION. *IET Conference Publication, 2013*(10), 458-466.

Suraj, T. H. (2015). Securing TORA against Sybil Attack in MANETs. ahmedabad,india: ABLAZE.

Thapara, S., & Sharmab, S. K. (2019). Attacks and Security Issues of Mobile Ad Hoc Networks . *International Conference on Sustainable Computing in Science, Technology & Management (SUSCOM-2019)*, 1463-1470.