

# Evaluasi Sistem Pendeteksi Intrusi Berbasis Anomali dengan *N-gram* dan *Incremental Learning*

I Made Agus Adi Wirawan, Royyana Muslim Ijtihadie, dan Baskoro Adi Pratomo  
Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember (ITS)

Jl. Arief Rahman Hakim, Surabaya 60111 Indonesia  
*e-mail*: roy@if.its.ac.id, baskoro@if.its.ac.id

**Abstrak**—Keberadaan teknologi informasi yang terus berkembang dengan pesat menjadikan kebutuhan akan penggunaannya semakin hari semakin meningkat. Transaksi data melalui internet telah menjadi kebutuhan wajib hampir dari semua perangkat lunak yang ada saat ini. Perangkat lunak seperti *media social*, *colud server*, *online game*, aplikasi layanan pemerintah, aplikasi pengontrol suatu tempat secara *remote*, dsb. Tentu dengan berbagai macam penggunaan internet tersebut dibutuhkan metode untuk mengamankan jaringannya. Sistem pendeteksi intrusi atau yang pada umumnya disebut IDS (*Intrusion Detection System*) merupakan solusi untuk mengamankan suatu jaringan. Sistem ini nantinya bertugas untuk menentukan apakah suatu paket merupakan bentuk serangan atau paket biasa sesuai dengan kondisi tertentu. Saat ini telah banyak dikembangkan aplikasi IDS (*Intrusion Detection System*), namun sebagian besar yang dikembangkan berbasis *signature* atau menggunakan *rule*, dan sebagian kecil menggunakan anomali. Anomali adalah suatu metode untuk mencari penyimpangan dalam sebuah data. Pada aplikasi ini konsep IDS yang diterapkan adalah IDS berbasis anomali dimana analisis datanya pada informasi paket data yang dikirimkan. Pada tugas akhir ini menggunakan dua metode, yaitu metode *n-gram* yang digunakan untuk menghitung distribusi *byte* karakter pada paket data sedangkan metode *mahalanobis distance* digunakan untuk menghitung jarak antara paket data normal dan paket data yang berupa intrusi. Metode *mahalanobis distance* dapat membedakan paket data yang normal dan paket data yang berupa intrusi dengan menghitung rata-rata dan standar deviasi dari paket data.

**Kata Kunci**—*Incremental Learning*, *Mahalanobis Distance*, *N-gram*

## I. PENDAHULUAN

SEMAKIN pesatnya perkembangan teknologi informasi memudahkan orang-orang untuk saling tukar menukar data baik melalui internet maupun intranet. Tentunya dengan mudahnya berbagi data itulah sangat memungkinkan terjadinya serangan terhadap data tersebut terutama melalui jaringan komputer. Sistem pendeteksi intrusi atau yang pada umumnya disebut IDS (*Intrusion Detection System*) [1] merupakan senjata utama untuk mengamankan suatu jaringan dimana sistem ini nantinya bertugas untuk mengidentifikasi dan mencatat apakah suatu paket data tersebut merupakan bentuk serangan atau paket data biasa.

Saat ini telah banyak dikembangkan aplikasi IDS (*Intrusion Detection System*), namun sebagian besar yang dikembangkan berbasis *signature* atau menggunakan *rule*, dan sebagian kecil menggunakan *anomaly*. *Anomaly* pada dasarnya adalah

mencari data yang menyimpang dari sekumpulan data normal. IDS yang berbasis pada *anomaly* [2] bersifat lebih fleksibel, karena dapat mengenali pola serangan baru tanpa harus memperbaharui basis data pola serangan. IDS yang berbasis pada anomali memiliki sebuah kecerdasan buatan yang mampu mendeteksi dan mengenali sebuah serangan. IDS yang berbasis anomali menggabungkan metode analisis dan statistik untuk mengenali penyimpangan tersebut. Kelemahan dari metode ini adalah kemungkinan salah identifikasi pada data yang diolah.

Untuk membedakan paket data normal dan paket data yang berupa intrusi diperlukan sebuah sistem deteksi intrusi dimana nantinya sistem deteksi intrusi tersebut menggunakan gabungan metode analisis dan statistik yang berfungsi mengenali perbedaan paket data normal maupun paket data berupa intrusi. Selain itu, sistem deteksi intrusi yang dapat mempelajari paket data normal yang baru sebagai data *training*.

Metode *n-gram* dapat digunakan untuk membuat model paket data yang sederhana dan cepat untuk dihitung khususnya menghitung distribusi karakter pada suatu paket data. *N-gram* merupakan metode yang paling efisien dan efektif dalam membuat model dari suatu paket data.

Metode *mahalanobis distance* berguna untuk membedakan paket-paket data berdasarkan anomali yang terjadi. Untuk dapat mempelajari paket data normal yang baru menggunakan metode *incremental learning*, dimana metode ini nantinya memperbaharui rata-rata dan standar deviasi dari model paket data yang ada pada data *training*.

Artikel ini menggunakan artikel [3] sebagai acuan mengimplementasikan metode *mahalanobis distance* untuk mengklasifikasikan paket data normal dan paket data intrusi, dan metode *n-gram* untuk memodelkan paket data, serta *incremental learning* untuk memperbaharui model data *training*. Dan juga membahas evaluasi dari penambahan proses *incremental learning* pada pendeteksian intrusi berbasis *anomaly* dengan *n-gram*. Apakah penambahan proses *incremental learning* dapat meningkatkan akurasi pendeteksian intrusi atau tidak.

## II. DESAIN DAN PERANCANGAN

### A. *Anomaly Based Intrusion Detection System*

*Anomaly* pada dasarnya adalah mencari sebuah data yang menyimpang dari sekumpulan data normal. IDS yang berbasis *anomaly* menggabungkan metode analisis dan statistik untuk mengenali penyimpangan tersebut [2]. IDS yang berbasis pada *anomaly* bersifat lebih fleksibel, karena dapat mengenali pola

serangan baru tanpa harus memperbaharui basis data pola serangan. IDS yang berbasis pada *anomaly* memiliki sebuah kecerdasan buatan yang mampu mendeteksi dan mengenali sebuah serangan.

Kelemahan dari metode anomali ini adalah kemungkinan terjadinya salah identifikasi pada data yang diolah, juga ada kemungkinan terjadi kesalahan pada data normal yang menyebabkan aplikasi tidak dapat mengenali serangan.

**B. N-gram Payload Model**

Pada dasarnya, model *N-gram* [4] adalah model probabilistik yang awalnya dirancang oleh ahli matematika dari Rusia pada awal abad ke-20 dan kemudian dikembangkan untuk memprediksi *item* berikutnya dalam urutan *item*. Item bisa berupa huruf / karakter, kata, atau yang lain sesuai dengan aplikasi. Salah satunya, model *n-gram* yang berbasis kata digunakan untuk memprediksi kata berikutnya dalam urutan kata tertentu. Dalam arti bahwa sebuah *n-gram* hanyalah sebuah wadah kumpulan kata dengan masing-masing memiliki panjang *n* kata. Sebagai contoh, sebuah *n-gram* ukuran 1 disebut sebagai *unigram*; ukuran 2 sebagai *bigram*; ukuran 3 sebagai *trigram*, dan seterusnya.

Pada pembangkitan karakter, *N-gram* terdiri dari *substring* sepanjang *n* karakter dari sejumlah *string* dalam definisi lain *n-gram* adalah potongan sejumlah *n* karakter dari sebuah *string*. Metode *n-gram* ini digunakan untuk mengambil potongan-potongan karakter huruf sejumlah *n* dari sebuah kata secara kontinuitas dibaca dari teks sumber sehingga akhir dari dokumen. Sebagai contoh: kata "TEXT" dapat diuraikan ke dalam beberapa *n-gram* berikut:

- uni-gram* : T, E, X, T
  - bi-gram* : TE, EX, XT
  - tri-gram* : TEX, EXT
  - quad-gram* : TEXT, EXT\_
- dan seterusnya.

Sedangkan pada pembangkit kata, metode *n-gram* ini digunakan untuk mengambil potongan kata sejumlah *n* dari sebuah rangkaian kata (kalimat, paragraf, bacaan) yang secara kontinuitas dibaca dari teks sumber hingga akhir dari dokumen. Sebagai contoh : kalimat "saya dapat melihat cahaya itu." Dapat diuraikan ke dalam beberapa *n-gram* berikut :

- uni-gram* : saya, dapat , melihat, cahaya, itu
  - bi-gram* : saya dapat, dapat melihat, melihat cahaya, cahaya itu
  - tri-gram* : saya dapat melihat, dapat melihat cahaya, melihat cahaya itu\_
- dan seterusnya.

Salah satu keunggulan menggunakan *n-gram* dan bukan suatu kata utuh secara keseluruhan adalah bahwa *n-gram* tidak terlalu sensitif terhadap kesalahan penulisan yang terdapat pada suatu dokumen.

**C. Simplified Mahalanobis Distance**

*Mahalanobis distance* [5] adalah sebuah metode statistika untuk menghitung jarak antara titik P dan distribusi D. Prinsip *Mahalanobis Distance* adalah menghitung jarak di ruang multidimensional antara sebuah pengamatan dengan pusat dari semua pengamatan. Pada artikel ini *Mahalanobis Distance* digunakan untuk menghitung jarak antara distribusi *byte* karakter dari *payload* baru terhadap model yang ada pada data

*training*. Semakin jauh jaraknya, semakin besar kemungkinan *payload* ini tidak normal.

*Mahalanobis distance* dari sebuah *payload* baru dapat dihitung jika sistem sudah mempunyai data *training*. Selanjutnya menghitung rata-rata dan standar deviasi dari model yang ada pada data *training*. Untuk menghitung rata-rata dari model yang ada pada data *training* dapat dilihat pada persamaan 1. Sedangkan untuk menghitung standar deviasi dari model yang ada pada data *training* dapat dilihat pada persamaan 2. Setelah selesai menghitung rata-rata dan standar deviasi dari model yang ada pada data *training* baru dapat menghitung jarak *mahalanobis* dari *payload* baru dengan menggunakan persamaan 1. Format data kasar yang ada pada *Mahalanobis distance* dapat dilihat pada *Tabel 1*.

$$d(x, \bar{y}) = \sum_{i=0}^{n-1} (|x_i - \bar{y}_i| / (\bar{\sigma}_i + \alpha)) \tag{1}$$

dimana,

*d* = jarak *mahalanobis*

*x<sub>i</sub>* = variabel ke-i dari *payload* baru

*y<sub>i</sub>* = rata-rata variabel ke-i dari model data *training*

*σ<sub>i</sub>* = standar deviasi variabel ke-i dari model data *training*

*α* = *smoothing factor*

Tabel 1  
Format data kasar didalam Mahalanobis Distance

| Object            | Variabel (karakteristik) |                 |     |                 |     |                    |                  |
|-------------------|--------------------------|-----------------|-----|-----------------|-----|--------------------|------------------|
|                   | X <sub>1</sub>           | X <sub>2</sub>  | ... | X <sub>i</sub>  | ... | X <sub>p-1</sub>   | X <sub>p</sub>   |
| 1                 | .                        | .               | ... | .               | ... | .                  | .                |
| 2                 | .                        | .               | ... | .               | ... | .                  | .                |
| 3                 | .                        | .               | ... | .               | ... | .                  | .                |
| .                 | .                        | .               | ... | .               | ... | .                  | .                |
| .                 | .                        | .               | ... | .               | ... | .                  | .                |
| K                 | X <sub>k1</sub>          | X <sub>k2</sub> | ... | X <sub>ki</sub> | ... | X <sub>k,p-1</sub> | X <sub>k,p</sub> |
| .                 | .                        | .               | ... | .               | ... | .                  | .                |
| .                 | .                        | .               | ... | .               | ... | .                  | .                |
| N                 | X <sub>N1</sub>          | X <sub>N2</sub> | ... | X <sub>Ni</sub> | ... | X <sub>N,p-1</sub> | X <sub>N,p</sub> |
| Average           | $\bar{X}_1$              | $\bar{X}_2$     | ... | $\bar{X}_i$     | ... | $\bar{X}_{p-1}$    | $\bar{X}_p$      |
| Standar deviation | S <sub>1</sub>           | S <sub>1</sub>  | ... | S <sub>1</sub>  | ... | S <sub>1</sub>     | S <sub>1</sub>   |

Persamaan untuk mencari rata-rata, yaitu:

$$\bar{X}_i = \frac{1}{N} \sum_{k=1}^N X_{ki} \tag{2}$$

dimana,

$\bar{X}_i$  = rata-rata variabel ke-i

*N* = jumlah *object* model

*X<sub>ki</sub>* = nilai variabel ke-i

Persamaan untuk mencari nilai standar deviasi, yaitu:

$$S_i = \sqrt{\frac{\sum_{k=1}^N (X_{ki} - \bar{X}_i)^2}{N-1}} \tag{3}$$

dimana,

*S<sub>i</sub>* = standar deviasi variabel ke-i

*X<sub>ki</sub>* = nilai dari variabel ke-i

$\bar{X}_i$  = rata-rata variabel ke-i

*N* = jumlah *object* model

**D. Incremental Learning**

*Incremental Learning* merupakan proses untuk memperbaharui nilai rata-rata dan standar deviasi dari model yang ada pada data *training* ketika menambahkan *payload* baru. Proses ini diperlukan untuk meningkatkan akurasi dari setiap model ketika ditambah data sampel baru.

Untuk menghitung *Mahalanobis distance* versi *Incremental Learning* diperlukan rata-rata dan standar deviasi dari masing-masing karakter ASCII untuk setiap sampel baru yang dihitung. Untuk menghitung rata-rata dari sebuah karakter dapat dilihat pada persamaan 2. Selanjutnya agar dapat memperbaharui nilai rata-rata dari model yang ada pada data *training*, diperlukan jumlah sampel yang telah dihitung sebelumnya [6]. Untuk menghitung nilai rata-rata yang baru dapat dilihat pada persamaan 4.

Sedangkan untuk menghitung standar deviasi yang baru diperlukan rata-rata dari  $x_i^2$  pada model sebelumnya. Untuk menghitung standar deviasi yang baru dapat dilihat pada persamaan 5.

Persamaan untuk menghitung rata-rata baru dari model yang diamati, yaitu:

$$\bar{x} = \frac{\bar{x} \times N + x_{N+1}}{N+1} = \bar{x} + \frac{x_{N+1} - \bar{x}}{N+1} \tag{4}$$

dimana,

- $\bar{x}$  = rata-rata baru
- $x_{N+1}$  = nilai dari variabel yang baru
- $N$  = jumlah sampel sebelumnya

Persamaan untuk menghitung standar deviasi baru dari model yang diamati, yaitu:

$$S_i = \sqrt{\frac{(n+1) \times (\sum_{i=1}^n x_i^2 + x_{n+1}^2) - (\sum_{i=1}^n x_i + x_{n+1})^2}{(n+1)n}} \tag{5}$$

dimana,

- $S_i$  = standar deviasi variabel ke-i
- $x_i$  = nilai dari variabel ke-i
- $x_{n+1}$  = nilai dari variabel yang baru
- $n$  = jumlah *object* model

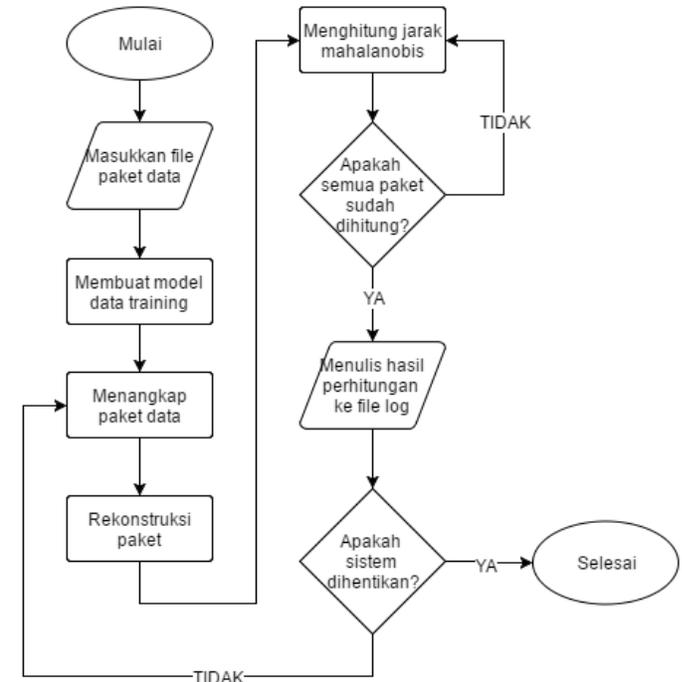
**E. Alur Kerja Sistem Secara Umum**

Secara umum sistem yang dibangun terdiri dari tiga proses utama, yaitu proses *training* data set, proses *sniffing* dan proses identifikasi serangan. Alur kerja sistem secara umum dapat dilihat pada *Gambar 1*.

Cara kerja sistem yang lebih detail yaitu, membaca *file* data set, membuat model data *training*, menyimpan model data *training*, menangkap paket, memproses paket dan membandingkan jarak *mahalanobis* paket dengan model serta pengambilan keputusan terhadap hasil perbandingan. Membuat model data *training* adalah proses dimana membaca *file* paket data dan ditampung pada *array of object*. *File* yang dapat dibaca hanya *file* yang berekstensi *\*.cap*, *\*.pcap*, *\*.tcpdump* dengan memanfaatkan *library* *Jpcap* [7]. Proses selanjutnya adalah menangkap *packet* dari *network* interface dengan bantuan *library* *Jpcap* lalu menyimpannya pada *array of object*. Selanjutnya adalah proses membandingkan jarak *mahalanobis packet* dengan model data. *Packet* yang dihitung hanya *packet* yang memiliki port tujuan yang kurang dari 1024. Setelah terdapat *packet* yang memenuhi syarat tersebut akan dilakukan

proses perhitungan, mulai dari perhitungan rata-rata dan standar deviasi dari *packet*. Dan selanjutnya proses pemanggilan fungsi *Mahalanobis Distance* untuk menghitung jarak *mahalanobis packet* dengan model data *training*. Setelah mendapatkan jarak *mahalanobis*, lalu dibandingkan dengan nilai *threshold* yang sudah ditentukan sebelumnya. Nilai *threshold* setiap port memiliki besaran yang berbeda. Jika jarak *mahalanobis packet* melebihi nilai *threshold*, maka *packet* tersebut dapat dikategorikan paket yang tidak normal.

Proses diatas diulangi sampai aplikasi dihentikan oleh pengguna dan menulis hasil keputusan terhadap proses perbandingan ke sebuah *file log*.



Gambar 1 Diagram alir kerja sistem secara umum

**III. UJI COBA DAN EVALUASI**

**A. Skenario Pengujian**

Pengujian yang dilakukan dibagi kedalam dua bagian, yakni:

1. Pengujian akurasi tanpa menambahkan proses *incremental learning*; dan
2. Pengujian akurasi dengan menambahkan proses *incremental learning*.

Parameter pengujian yang digunakan adalah besarnya ukuran *window*. *Window size* yang dimaksud adalah jumlah paket data yang ditangkap, jika jumlah paket data yang ditangkap sudah memenuhi *window size* selanjutnya paket data tersebut diproses untuk mengetahui jarak *mahalanobis* setiap paket.

**B. Data Pengujian**

*Data uji* yang nantinya akan diproses dengan metode *two fold cross validation* [8]. Metode pengujian ini digunakan karena sistem yang dibuat memiliki sifat mirip dengan aplikasi-aplikasi yang berbasis *machine learning*. Tahap *learning* yang dimaksud pada sistem ini adalah tahap untuk penentuan *threshold* pendeteksian. Berangkat dari hal ini, maka metode *two fold cross validation* dipilih sehingga kebutuhan untuk mendapatkan *threshold* dan data uji terpenuhi. Dengan metode *two fold cross validation*, kumpulan data uji akan

dibagi 2 sama banyak. Kemudian salah satu kumpulan data akan dijadikan data training sekaligus menentukan threshold dari aplikasi pendeteksian. Setelah selesai, maka kegiatan diulang kembali dengan kumpulan data yang sebelumnya menjadi data uji akan dijadikan data training dan kumpulan data training dijadikan data uji. Berikut pada

Tabel 2 disajikan data uji dimana parameter pembeda antar data uji adalah besarnya ukuran *window*. *Window size* yang dimaksud adalah jumlah paket data yang ditangkap, jika jumlah paket data yang ditangkap sudah memenuhi *window size* baru paket data tersebut diproses untuk mengetahui jarak *mahalanobis* setiap paket. Data uji yang digunakan adalah file paket data hasil tangkapan paket data pada jaringan eksternal DARPA [9] pada minggu ke-4 yang berjumlah 5 file paket data.

Data yang akan dicatat nantinya adalah jarak *mahalanobis* dari paket data normal dan paket data yang berupa intrusi. Kedua variable tersebut nantinya akan diolah menjadi nilai *threshold* dengan cara menjumlahkan jarak *mahalanobis* terkecil dari paket data yang berupa intrusi dengan jarak *mahalanobis* terbesar dari paket data normal dan kemudian dibagi 2. Berikut pada persamaan 6 dijabarkan cara untuk mendapatkan nilai *threshold* pendeteksian.

$$Threshold = \frac{\min \text{ jarak serangan} + \max \text{ jarak normal}}{2} \quad (6)$$

Tabel 2  
Data uji

| No | Ukuran <i>window</i> | Port TCP |    |    |    | Port UDP |
|----|----------------------|----------|----|----|----|----------|
| 1  | 10000                | 21       | 23 | 25 | 80 | 53       |
| 2  | 15000                | 21       | 23 | 25 | 80 | 53       |
| 3  | 20000                | 21       | 23 | 25 | 80 | 53       |

Untuk menghitung akurasi, digunakan metode *confussion matrix* [10]. Penggunaan metode ini digunakan karena tergolong mudah untuk digunakan dan dapat menghasilkan nilai-nilai pengujian selain akurasi, seperti *true positive rate* dan *false positive rate*. *Confussion matrix* yang akan digunakan untuk pengujian akurasi adalah *confussion matrix* dengan ukuran 2x2. Berikut pada Gambar 2 ditunjukkan model *confussion matrix* beserta kemudian dijelaskan definisi masing-masing kelasnya.

|        |         | PREDICTED |        |
|--------|---------|-----------|--------|
|        |         | INTRUSI   | NORMAL |
| ACTUAL | INTRUSI | A         | B      |
|        | NORMAL  | C         | D      |

Gambar 2 Model *Confussion Matrix* untuk pengujian

### C. Hasil Pengujian

#### 1) Pengujian akurasi tanpa menambahkan proses *incremental learning*

Dengan berbekal *threshold* tersebut maka dilakukan pengujian terhadap data testing minggu ke-5, hari ke-1 tanpa proses *incremental learning*. Hasil pengujian dari 10000 paket data, yaitu terdapat 5328 *connection* yang terdiri dari 5308 paket normal dan 20 paket intrusi.

Setelah semua data diuji, maka data dapat diproses hasilnya dengan *confussion matrix*. Pada Tabel 3 disajikan klasifikasi jumlah masing-masing kelas berdasarkan pada hasil pengujian.

Tabel 3  
*Confussion matrix* uji coba 1

|        |         | PREDICTED |        |
|--------|---------|-----------|--------|
|        |         | INTRUSI   | NORMAL |
| ACTUAL | INTRUSI | 6         | 355    |
|        | NORMAL  | 14        | 4935   |

Berdasarkan pada jumlah diatas, maka didapatkan penilaian berdasarkan rumus-rumus yang terkait dengan *confussion matrix* yang disajikan pada Tabel 4.

Tabel 4  
Hasil penilaian percobaan 1

| No | Jenis Penilaian                 | Nilai  | Persentase |
|----|---------------------------------|--------|------------|
| 1  | Akurasi (AC)                    | 0.9307 | 93.07%     |
| 2  | <i>True positive rate</i> (TP)  | 0.0166 | 1.66%      |
| 3  | <i>False negative rate</i> (FN) | 0.9834 | 98.34%     |
| 4  | <i>False positive rate</i> (FP) | 0.0028 | 0.28%      |
| 5  | <i>True negative rate</i> (TN)  | 0.9972 | 99.72%     |
| 6  | Presisi (P)                     | 0.3    | 30.0%      |

#### 2) Pengujian akurasi dengan menambahkan proses *incremental learning*

Dengan berbekal *threshold* tersebut maka dilakukan pengujian terhadap data testing minggu ke-5, hari ke-1 dengan tambahan proses *incremental learning*. Hasil pengujian dari 10000 paket data, yaitu terdapat 5328 *connection* yang terdiri dari 1577 paket normal dan 3751 paket intrusi.

Setelah semua data diuji, maka data dapat diproses hasilnya dengan *confussion matrix*. Pada Tabel 5 disajikan klasifikasi jumlah masing-masing kelas berdasarkan pada hasil pengujian.

Tabel 5  
*Confussion matrix* uji coba 2

|        |         | PREDICTED |        |
|--------|---------|-----------|--------|
|        |         | INTRUSI   | NORMAL |
| ACTUAL | INTRUSI | 187       | 174    |
|        | NORMAL  | 3564      | 1403   |

Berdasarkan pada jumlah diatas, maka didapatkan penilaian berdasarkan rumus-rumus yang terkait dengan *confussion matrix* yang disajikan pada Tabel 6.

Tabel 6  
Hasil penilaian percobaan 2

| No | Jenis Penilaian                 | Nilai  | Persentase |
|----|---------------------------------|--------|------------|
| 1  | Akurasi (AC)                    | 0.2984 | 29.84%     |
| 2  | <i>True positive rate</i> (TP)  | 0.518  | 51.8%      |
| 3  | <i>False negative rate</i> (FN) | 0.482  | 48.2%      |
| 4  | <i>False positive rate</i> (FP) | 0.7175 | 71.75%     |
| 5  | <i>True negative rate</i> (TN)  | 0.2825 | 28.25%     |
| 6  | Presisi (P)                     | 0.0499 | 4.99%      |

## IV. KESIMPULAN

### A. Kesimpulan

Dari hasil uji coba yang telah dilakukan dapat diambil beberapa kesimpulan sebagai berikut:

1. Metode *Mahalanobis Distance* tidak dapat digunakan untuk mengklasifikasikan antara paket data normal dan paket data yang berupa intrusi untuk protokol HTTP. Jarak yang dihasilkan pada saat *training* menggunakan paket data normal maupun paket data yang berupa intrusi yaitu bernilai 0. Sehingga paket data normal maupun paket data intrusi tidak dapat dibedakan.

2. Sistem yang dibuat untuk pendeteksi intrusi menggunakan metode *Mahalanobis Distance* tanpa proses *incremental learning* dapat mendeteksi intrusi dengan persentase kebenaran sekitar 93%, namun dengan tambahan proses *incremental learning* hanya dapat mendeteksi intrusi dengan persentase kebenaran sekitar 20%. Dari hasil tersebut, dengan tambahan proses *incremental learning* mengurangi tingkat akurasi pendeteksian intrusi.

### B. Saran

Adapun saran-saran yang diberikan untuk pengembangan sistem ini selanjutnya adalah karena membedakan paket data normal dengan paket data serangan menggunakan metode *mahalanobis distance* dengan proses *incremental learning* kurang akurat dibandingkan tanpa proses *incremental learning*. Hal ini dikarenakan dengan menambahkan proses *incremental learning*, rata-rata dan standar deviasi pada model diperbaharui tetapi *threshold* yang digunakan untuk mendeteksi intrusi tidak diperbaharui, sehingga *threshold* yang ada tidak akurat untuk mendeteksi intrusi. Perlu ada implementasi metode lain sehingga dapat membantu meningkatkan keakuratan pendeteksian intrusi.

### DAFTAR PUSTAKA

- [1] SANS Institute, "Understanding Intrusion Detection System," *SANS Institute Reading Room*, pp. 1-9, 2001.
- [2] "Intrusion detection system," [Online]. Available: [https://en.wikipedia.org/wiki/Intrusion\\_detection\\_system](https://en.wikipedia.org/wiki/Intrusion_detection_system). [Diakses 22 June 2016].
- [3] S. J. S. Ke Wang, "Anomalous *Payload*-based Network Intrusion Detection".
- [4] A. Hanafi, "Pengenalan Bahasa Suku Bangsa Indonesia Berbasis Teks Menggunakan Metode *N-gram*." IT TELKOM, 2009.
- [5] "*Mahalanobis distance*," [Online]. Available: [https://en.wikipedia.org/wiki/Mahalanobis\\_distance](https://en.wikipedia.org/wiki/Mahalanobis_distance). [Diakses 22 June 2016].
- [6] D. E. Knuth, "The Art of Computer Programming," *Fundamental Algorithms*. Addison Wesley, vol. 1, 1973.
- [7] K. Fuji, "a Java library for capturing and sending network packets," Jpcap, 15 May 2007. [Online]. Available: <http://jpcap.gitspot.com/>. [Diakses 23 May 2016].
- [8] V. Galleys, "Cross Validation," 2006. [Online]. Available: [http://www.cse.iitb.ac.id/~tarung/smt/papers\\_ppt/ency-cross-validation.pdf](http://www.cse.iitb.ac.id/~tarung/smt/papers_ppt/ency-cross-validation.pdf). [Diakses 24 June 2016].
- [9] MIT Lincoln Laboratory, "MIT Lincoln Laboratory: Cyber system & technolog: DARPA Intrusion Detection," MIT Lincoln Laboratory, [Online]. Available: <https://www.ll.mit.edu/mission/communications/cyber/CSTcorporation/eval/docs/index.html>. [Diakses 23 Mei 2016].
- [10] Kohavi, "Confusion Matrix," 1999. [Online]. Available: [http://www2.cs.uregina.ca/~dbd/cs831/notes/confusion\\_matrix/confusion\\_matrix.html](http://www2.cs.uregina.ca/~dbd/cs831/notes/confusion_matrix/confusion_matrix.html). [Diakses 24 June 2016].