PERANCANGAN ALGORITMA ANGGI (AA) DENGAN MEMANFAATKAN *DIFFIE-HELLMAN* DAN *RONALD RIVEST* (RC4) UNTUK MEMBANGUN SISTEM KEAMANAN BERBASIS *PORT KNOCKING*

Anggi Sri Septiani Suhendar, Haruno Sajati, Yenni Astuti
Teknik Informatika STTA Yogyakarta
informatika@stta.ac.id

ABSTRACT

In managing a server, an administrator takes a variety of ways for ensuring the safety and cosiness of the system. One type of threat that attack on the server is done on a port that is in an open state, thus making people who do not have access rights to perform port scanning to infiltrate into the server. One solution to cope with the attack on the port can be done by using port knocking. Port knocking is a method that can be applied to the server with the workings of such a code to unlock the safe. Port knocking is done by inserting a predetermined port sequenced to open a specific port. At the port sequenced delivery process takes an additional authentication likes keyport. Anggi algorithm (AA) is an algorithm that is built by combination of Diffie-Hellman and RC4. The algorithm is used for encryption and descriptions a keyport from client side to the server side. In the process of sending a keyport to be modified by using the arithmetic calculation. Process of encryption and description keyport is done to prevent some attacks on the server by hackers with extracting such information after successfully breaking into a port. The results of implementation AA algorithm on the process of port knocking can secure the delivery process of keyport and improve the security of server.

Keywords: Port Knocking, Sequenced Port, AA Algorithms, Keyport

1. PENDAHULUAN

Berbagai jenis serangan yang dapat terjadi pada jaringan internet menjadikan alasan penting adanya sebuah sistem keamanan. Salah satu jenis ancaman pada *server* adalah serangan yang dilakukan pada suatu *port* yang berada dalam keadaan terbuka, sehingga membuat orang yang tidak mempunyai hak akses dapat melakukan *port scanning* untuk menyusup ke dalam *server*. Salah satu solusi untuk mengatasi serangan pada *port* dapat dilakukan dengan menggunakan metode *port knocking*.

Dalam proses melakukan *knocking* pada suatu *server* masih terdapat suatu masalah keamanan, yaitu ketika informasi *sequenced port* yang dikirimkan ke *server* kemungkinan masih dapat disadap oleh orang yang tidak bertanggung jawab yang biasa dikenal dengan istilah *Man In The Middle* (MITM). Oleh karena itu, dibutuhkan sebuah autentikasi tambahan dalam bentuk *keyport*. Definisi *keyport* merupakan sebuah *passkey* (kunci untuk pengamanan jaluk masuk) yang telah disetujui oleh sisi *client* dan sisi *server* yang menjadi ukuran legalitas bahwa *client* yang melakukan *request* koneksi ke *server* merupakan *client* yang memiliki hak akses. Dalam pengiriman data *keyport* akan dilakukan proses enkripsi menggunakan Algoritma Anggi (AA).

2. LANDASAN TEORI

2.1 Tinjauan Pustaka

Palmgren (2009) pada *Diffie-Helman Key Exchange A-Non-Matematichian's Explanation*, menyatakan bahwa Algoritma Diffie-Helman diperkenalkan oleh Whitfield Diffie dan Martin Hellman pada tahun 1976. Algoritma ini merupakan sebuah metode kriptografi yang menggunakan konsep aritmetika modulo.

Dalam jurnal penelitian yang dilakukan oleh Irwan Sembiring, Indrastati R. Widiasari, dan Danu Prasetyo (2009), membahas mengenai penerapan metode *port knocking* sebagai sistem keamanan pada *server*. Hasil dari penelitian tersebut menyatakan bahwa *port knocking* sangat berguna jika diterapkan sebagai sistem keamanan pada *server*. Dengan melihat *rule* pada *iptables* disisi *server* yang menerapkan *port knocking*, *server* dapat melakukan *monitoring* terhadap *client* yang melakukan usaha untuk mengakses sebuah *port* pada *server* tersebut.

2.2 Metode *Port Knocking*

Port knocking merupakan sebuah metode sistem keamanan yang di terapkan pada sebuah *server*. *Port knocking* bekerja menggunakan sistem ketukan dari *sequenced port* yang telah ditentukan. Jika urutan *sequenced port* benar, maka *server* memberikan ijin kepada *client* untuk mengakses *port* tersebut. Apabila urutan salah, *client* tidak dapat mengakses *port* tersebut.

2.3 Algoritma *Ronald Rivest* (RC4)

Sistem algoritma RC4 dikembangkan oleh Ronald Rivest pada tahun 1984. RC4 menerapkan sistem sandi *stream* yang bertujuan agar sistem sandi mampu melakukan penyandian unit data terkecil secara *realtime*. RC4 menghasilkan *pseudorandom stream* bit. Seperti halnya *stream cipher* lainnya, algoritma RC4 dapat digunakan untuk mengenkripsi sebuah *plainteks* menjadi *chipertext* dengan menggunakan *bit-wise* Xor *(Exclusive-or)*. Proses dekripsinya dilakukan dengan cara yang sama.

2.4 Algoritma Anggi (AA)

Algoritma AA didesain menjadi sebuah sistem keamanan pada proses pengiriman *keyport* dari sisi *client* ke sisi *server* yang menerapkan metode *port knocking*. Konsep algoritma AA dibuat berdasarkan algoritma enkripsi-deskripsi yang telah ada sebelumnya, yaitu kombinasi antara Algoritma Diffie-Helman dan Algoritma RC4. Dengan Algoritma AA, proses pengiriman *keyport* akan dilakukan modifikasi menggunakan perhitungan aritmetika.

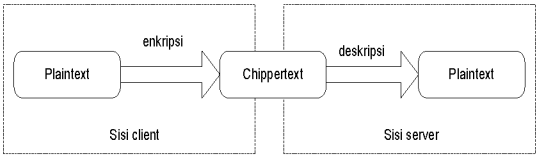
3. PERANCANGAN ALGORITMA

Algoritma AA merupakan sebuah algoritma baru yang diciptakan oleh Anggi. Ide dasar pembuatan algoritma ini dengan mengkombinasikan cara kerja algoritma yang telah ada sebelumnya, yaitu Algoritma Diffie-Hellman dan Algoritma RC4 untuk melakukan proses enkripsi dan deskripsi *keyport*. Pada proses pengiriman *keyport* dilakukan modifikasi menggunakan perhitungan aritmetika. Proses enkripsi dan deskripsi *keyport* dilakukan sebagai usaha untuk mencegah serangan pada *server* seperti *port scanning* yang kemungkinan seorang *hacker* juga dapat melakukan penyadapan informasi setelah berhasil menyusup ke suatu *port*. Proses perhitungan dilakukan dengan mencari nilai modulus seperti yang dilakukan pada proses *key exchange* pada Algoritma Diffie Hellman.

Tabel 1 Proses Perhitungan *Key Exchange*

Inisialisasi parameter x	x	
Inisialisasi parameter y	y	
	Client (a)	*Server (b)*
Memilih bilangan acak *(random secret)*	A	b
Proses pengiriman kunci dengan *random secret* menghasilkan *public key*	*Client* mengirim (y^a mod x)	*Server* mengirim (y^b mod x)
Proses pemangkatan *key exchange* dengan *random secret* menghasilkan *private key*	*Client* menghitung: $(y^b \bmod x)^a \bmod x$	*Server* menghitung: $(y^a \bmod x)^b \bmod x$

Untuk menghindari adanya pihak yang melakukan penyadapan informasi *keyport* yang dikirimkan oleh *client* ke *server*, selain dengan menerapkan pertukaran kunci, *keyport* juga akan dienkripsi terlebih dahulu pada sisi *client* kemudian dikirimkan ke *server* dalam bentuk *chipertext*. Proses Algoritma AA dalam melakukan proses enkripsi-deskripsi *keyport* dijelaskan pada Gambar 1.



Untuk menambah *sistem* keamanan *keyport*, sebelum proses enkripsi pada *keyport* akan dihitung menggunakan Persamaan 3.1.

$$Keyport = keyport2/(2*5*\sqrt{9}) \quad (3.1)$$

Saat proses deskripsi dilakukan proses menggunakan Persamaan 3.2

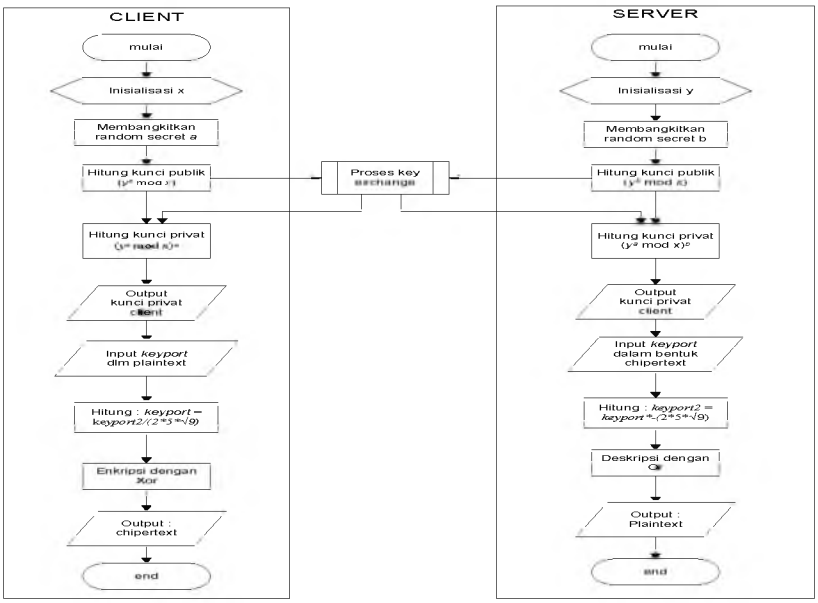
$$Keyport2 = keyport* -(2*5*\sqrt{9}) \quad (3.2)$$

Contoh perhitungan dari Persamaan 3.1 dan Persamaan 3.2 dijelaskan pada Tabel 2.

Tabel 2 Perhitungan *Keyport*

Perhitungan *Keyport2*	*Keyport*
Enkripsi = 3000	Enkripsi = 1000
Deskripsi = -3000	Deskripsi = -1000

Pada Tabel 2, nominal yang diinputkan *client* untuk melakukan proses ketukan adalah *keyport2* yaitu 3000, pada proses pengiriman, *keyport* diolah dengan rumus di atas dan hasil perhitungan dibandingkan dengan *keyport* yang telah didefinisikan sebelumnya pada aplikasi. Angka pada Persamaan 3.1 dan 3.2 didesain melalui tanggal lahir si pembuat algoritma. *Flowchart* Algoritma AA dalam proses enkripsi dan deskripsi *keyport* ditunjukkan pada Gambar 2.



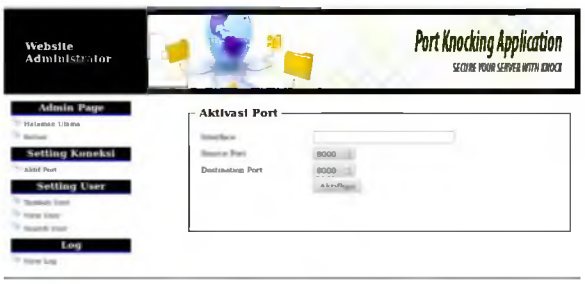
Gambar 2 *Flowchart* Algoritma AA

4. IMPLEMENTASI DAN ANALISA HASIL

4.1 Penjelasan Aplikasi

Penjelasan aplikasi terdiri dari dua bagian, yaitu pada sisi admin dan sisi *client*. Pada sisi admin terdiri dari menu login, menu utama, menu *setting* koneksi, menu *setting user*, dan log. Pada sisi *client* yang mengakses terdiri dari menu utama, menu proses *open port*, menu proses *close port*, menu log, dan menu *about*.

Pada halaman *setting* koneksi sisi admin digunakan untuk mengaktifkan program *daemon* dan me-nonaktifkan layanan *port*. Halaman *setting* koneksi pada sisi admin ditunjukkan pada Gambar 5.



Gambar 5 Halaman *Setting* Koneksi pada Sisi Admin

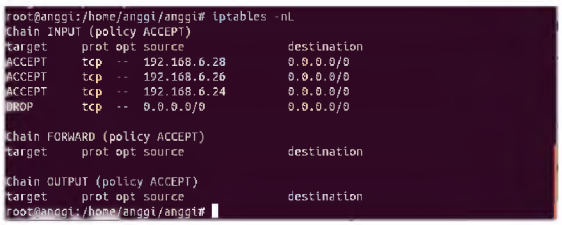
Pada sisi *client*, terdapat halaman untuk proses *open port* digunakan untuk mengakses sebuah layanan *port* pada *server*. Pada halaman ini, *client* harus memasukkan IP *server*, *sequenced port*, dan *keyport*. Halaman proses *open port* ditunjukkan pada Gambar 6.

Gambar 6 Halaman Proses *Open Port* pada Sisi *Client*

4.2 Uji Fungsi Aplikasi

4.2.1 Uji Coba pada Jaringan LAN

Pengujian pada jaringan LAN menggunakan lima buah komputer yang sudah diberi alamat IP komputer agar dapat saling terhubung. satu buah komputer dijadikan *server*, dan empat buah komputer dijadikan sebagai *client*. Hasil dari *client* yang berhasil melakukan *knocking* pada proses uji coba menggunakan jaringan LAN menggunakan empat *client*. Dengan format tiga *client* merupakan *client* yang sah dan satu *client* merupakan seorang penyusup ditunjukkan pada Gambar 7.



Gambar 7 Hasil Uji Coba *Knocking* pada Jaringan LAN

Tabel hasil uji coba pada jaringan LAN saat *client* mengakses *port* SSH dijelaskan pada Tabel 3.

Tabel 3 Hasil Uji Coba Akses *Port* SSH melalui Jaringan LAN

NO	IP *client*	*Username*	Waktu akses ke SSH
1	192.168.6.28	Anggi	00:00:05
2	192.168.6.24	Anggi	00:00:05
3	192.168.6.26	Wawan	00:00:05

Dari hasil uji coba menggunakan jaringan LAN, diperoleh hasil rata-rata waktu *load* untuk mengakses web adalah 3 detik dan waktu untuk akses ke *port* SSH setelah proses *knocking* berhasil adalah 5 detik.

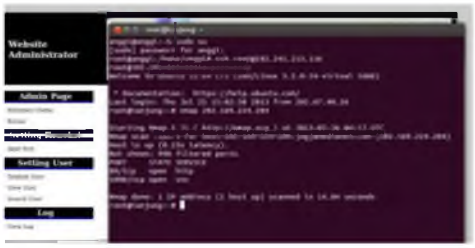
4.2.2 Uji Coba pada Jaringan Internet

Uji coba ini memanfaatkan sebuah *server* untuk menyimpan *hosting file* dan menanamkan *database server*. Pada pengujian dengan jaringan internet, apabila *client* berhasil

melakukan proses *knocking*, *client* dapat mengakses *port* pada *server* dan masuk ke *port* yang ingin dituju. *Port* yang dapat diakses oleh *client* ditunjukkan pada Gambar 8.

Gambar 8 *Port* yang dapat diakses oleh *Client* setelah Proses *Knocking* pada Uji coba dengan Jaringan Internet

Saat *client* belum melakukan proses *knocking*, maka saat melakukan *port scanning*, *client* tidak dapat mengakses seluruh *port* yang aktif pada *server*. Hal tersebut ditunjukkan pada Gambar 9.



Gambar 9 *Port Scanning* ketika *Port* dalam Keadaan Tertutup pada Uji Coba dengan Jaringan Internet

Tabel hasil uji coba pada jaringan internet saat *client* mengakses *port* SSH dijelaskan pada Tabel 4.

Tabel 4 Hasil Uji Coba Akses *Port* SSH pada Jaringan Internet

NO	IP *client*	*Username*	Waktu Akses ke SSH
1	203.78.123.244	Anggi	00:00:07
2	202.67.41.68	Anggi	00:00:10

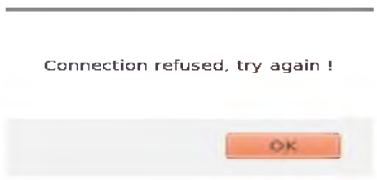
Dari hasil uji coba menggunakan jaringan internet, diperoleh hasil rata-rata waktu *load* untuk mengakses web adalah 7 detik dan waktu untuk akses ke *port* SSH setelah proses *knocking* berhasil adalah 8 detik.

4.2.3 Penerapan Algoritma AA pada Aplikasi

Algoritma AA mengamankan proses enkripsi dan deskripsi *keyport*, ketika *server* menolak *request* dari *client* maka pada *logging* di sistem akan menampilkan pesan bahwa *server* menerima *bad password* dari *client*. Proses tersebut ditunjukkan pada Gambar 10.

Gambar 10 Uji Coba *Keyport*

Sementara itu apabila proses knocking gagal dilakukan, maka akan muncul *Message Box* seperti yang ditunjukkan pada Gambar 11.



Gambar 11 *Message Box* Proses *Knocking* Gagal dilakukan

5. ANALISA

Pada uji coba pada jaringan LAN menggunakan lima buah komputer, satu buah komputer dijadikan *server* dan empat buah komputer bertindak sebagai *client*. Hasil uji coba memperoleh rata-rata waktu yang dibutuhkan untuk me-*load* halaman web adalah 3 detik dan waktu untuk akses ke *port* SSH setelah proses *knocking* berhasil adalah 5 detik.

Pengujian pada jaringan internet dilakukan dengan menggunakan dua buah komputer sebagai *client*, dan memanfaatkan sebuah *server dedicated* untuk menjadi *server* yang menerapkan *port knocking*. Hasil uji coba diperoleh hasil rata-rata waktu *load* untuk mengakses web adalah 7 detik dan waktu untuk akses ke *port* SSH setelah proses *knocking* berhasil adalah 8 detik. Waktu yang dibutuhkan untuk menerapkan aplikasi via jaringan LAN lebih cepat dibandingkan pada jaringan internet. Hal tersebut disebabkan uji coba pada jaringan internet sangat tergantung pada kestabilan koneksi jaringan internet yang dipakai.

Pengujian terakhir, yaitu pengujian pada Algoritma AA yang digunakan dalam proses enkripsi dan deskripsi *keyport*. Hasil uji coba diperoleh apabila *keyport* tidak sesuai, maka akan muncul peringatan bahwa proses *knocking* gagal dilakukan dan pada *logging* akan menerima pesan "menerima *bad password*", apabila proses berhasil maka *server* akan menerima akses *client* yang melakukan proses *knocking*.

6. KESIMPULAN DAN SARAN

6.1 Kesimpulan

1. Aplikasi yang dirancang dalam tugas ini dapat digunakan untuk meningkatkan keamanan pada *server* dengan cara menerapkan aturan *rule iptables* pada sisi *INPUT* dalam penerapan metode *port knocking*.
2. Penerapkan metode *port knocking* memungkinkan seorang admin atau *client* yang memiliki hak akses masih dapat membuka *port* meskipun *sever* telah menutup semua layanan *port*.
3. Algoritma AA pada metode *port knocking* dapat mengamankan pengiriman *keyport* dari sisi *client* ke sisi *server* dengan adanya *key-exchange* dan proses enkripsi *keyport*.

6.2 Saran

1. Aplikasi ini dapat dikembangkan dalam platform yang berbeda atau dalam bentuk *mobile*.
2. Dapat mengembangkan aplikasi ini dengan membuat sebuah *keyport* dan *sequenced port* yang berbeda untuk setiap *client* yang memiliki hak akses.

DAFTAR PUSTAKA

Dede, Sopandi. 2006. *Instalasi dan Konfigurasi Jaringan Komputer*. Informatika. Bandung.

Kurniawan, Agus. 2012. *Network Forensics Panduan Analisis & Investigasi Paket Data Jaringan menggunakan Wireshark.*. Andi Offset. Yogyakarta.

Palmgren, Keith. 2006. *Diffie-Helman Key Exchange A-Non-Matematichian's Explanation*. NetIP, Inc. United States. *(https://learningnetwork.cisco.com/servlet/JiveServlet/download/294583-67040/WP_Palmgren_DH.pdf,* diakses pada 12 Februari 2013).

Sadikin, Rifki. 2012. *Kriptografi Untuk Keamanan Jaringan*. Andi Offset. Yogyakarta.

Sembring, Irwan dkk. November 2009. *Analisa dan Implementasi Sistem Keamanan Jaringan Komputer dengan Iptables sebagai Firewall menggunakan Metode Port Knocking*. Jurnal Informatika Volume V Nomor II, November 2009, hal 1-15, ISSN 1693-7279. Salatiga.

Stallings, William. 2006. *Cryptography and Network Security*. Fourth Edition, Pearson Education. New Jersey.