**PENGAMANAN PERANGKAT IOT DENGAN ENCRYPTED *PORT KNOCKING***

**UNIVERSITAS UDAYANA**

**KOMPETENSI JARINGAN**

**SKRIPSI**



**MARIA OKTA SAFIRA**

**NIM. 1608561055**

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS UDAYANA**

**BUKIT JIMBARAN**

**2020**

**SURAT PERNYATAAN KEASLIAN KARYA ILMIAH**

# **LEMBAR PENGESAHAN PROPOSAL TUGAS AKHIR**

Judul : Pengamanan Perangkat IoT dengan Encrypted *Port knocking*

Kompetensi : Jaringan Sensor Nirkabel

Nama : Maria Okta Safira

NIM : 1608561055

Tanggal Ujian :

|  |  |  |  |
| --- | --- | --- | --- |
| Disetujui Oleh : | | | |
| Ketua Penguji  I Dewa Made Bayu Atmaja Darmawan,S.Kom .,M.Cs NIP. 198901272012121001 | | Sekretaris Penguji  I Gede Arta Wibawa, S.T., M.Kom  NIP. 198310222008121001 | |
| Anggota Penguji  Dr. Anak Agung Istri Ngurah Eka Karyawati, S.Si.,M.Eng.  NIP. 197404071998022001 | Anggota Penguji  Dr. I Ketut Gede Suhartana, S.Kom., M.Kom  NIP. 197201102008121001 | | Anggota Penguji  Komang Ari Mogi, S.Kom., M.Cs.  NIP. 198409242008011007 |
| Mengetahui,  Komisi Seminar dan Tugas Akhir  Program Studi Teknik Informatika FMIPA UNUD    I Gusti Ngurah Anom Cahyadi Putra,ST.,M.Cs  NIP. 198403172019031005 | | | |

Judul : Pengamanan Perangkat IoT dengan Encrypted *Port knocking*

Nama : Maria Okta Safira

NIM : 1608561055

Pembimbing : 1. Dr. I Ketut Gede Suhartana, S.Kom., M.Kom

2. I Komang Ari Mogi, S.Kom., M.Kom

# **ABSTRAK**

Judul : Pengamanan Perangkat IoT dengan Encrypted *Port knocking*

Nama : Maria Okta Safira

NIM : 1608561055

Pembimbing : 1. Dr. I Ketut Gede Suhartana, S.Kom., M.Kom

2. I Komang Ari Mogi, S.Kom., M.Kom

# **ABSTRACT**

# **KATA PENGANTAR**

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa atas segala limpahan rahmat sehingga penulis dapat menyelesaikan proposal tugas akhir dengan judul Pengamanan Perangkat IoT Dengan Encrypted *Port knocking*.

Terimakasih penulis ucapkan kepada setiap pihak yang telah berkontribusi dalam membantu proses penyelesaian laporan ini, diantaranya:

1. Bapak Dr. I Ketut Gede Suhartana, S.Kom., M.Kom sebagai pembimbing 1 yang telah banyak membantu menyempurnakan proposal ini.
2. Bapak I Komang Ari Mogi, S.Kom., M.Kom sebagai pembimbing 2 yang telah banyak membantu menyempurnakan proposal ini.
3. Para dosen pengajar di program studi Teknik Informatika fakultas MIPA Universitas Udayana yang telah memberikan saran dan masukan dalam menyempurnakan proposal ini.
4. Kawan-kawan di Program Studi Teknik Informatika yang telah memberikan dukungan moral dalam penyelesaian penelitian ini.
5. Serta semua pihak yang telah terlibat dan banyak membantu sehingga proposal ini dapat terselesaikan.

Penulis menyadari bahwa laporan ini jauh dari kesempurnaan, oleh karena itu penulis mengharapkan kritik dan saran yang bersifat membangun dari semua pihak, guna menyempurnakan isi dari laporan ini. Akhir kata penulis sampaikan terimakasih dan semoga laporan ini dapat bermanfaat bagi pembaca.

Bukit Jimbaran, Februari 2020

Penulis

# **DAFTAR ISI**

[**LEMBAR PENGESAHAN PROPOSAL TUGAS AKHIR** iii](#_Toc42773586)

[**ABSTRAK** iv](#_Toc42773587)

[**ABSTRACT** v](#_Toc42773588)

[**KATA PENGANTAR** vi](#_Toc42773589)

[**DAFTAR ISI** vii](#_Toc42773590)

[**DAFTAR GAMBAR** viii](#_Toc42773591)

[**DAFTAR TABEL** x](#_Toc42773592)

[**DAFTAR LAMPIRAN** xi](#_Toc42773593)

[**BAB I** 1](#_Toc42773594)

[**PENDAHULUAN** 1](#_Toc42773595)

[**1.1.** **Latar Belakang** 1](#_Toc42773596)

[**1.2.** **Rumusan Masalah** 3](#_Toc42773597)

[**1.3.** **Tujuan** 3](#_Toc42773598)

[**1.4.** **Batasan Masalah** 3](#_Toc42773599)

[**1.5.** **Manfaat** 4](#_Toc42773600)

[**1.6.** **Metodelogi Penelitian** 4](#_Toc42773601)

[**1.6.1.** **Desain Penelitian** 4](#_Toc42773602)

[**1.6.2.** **Pengumpulan Data** 5](#_Toc42773603)

[**1.6.3.** **Metode *Port knocking*** 5](#_Toc42773604)

[**1.6.4.** **Metode RSA** 6](#_Toc42773605)

[**1.6.5.** **Skenario Pengujian** 7](#_Toc42773606)

[**BAB II** 8](#_Toc42773607)

[**TINJAUAN PUSTAKA** 8](#_Toc42773608)

[**2.1.** **Tinjauan Pustaka Jurnal** 8](#_Toc42773609)

[**2.2.** **Tinjauan Pustaka Teori** 12](#_Toc42773610)

[**2.2.1.** ***Internet of things*** 12](#_Toc42773611)

[**2.2.2.** **Port** 13](#_Toc42773612)

[**2.2.3.** ***Firewall*** 14](#_Toc42773613)

[**2.2.4.** ***Port knocking*** 15](#_Toc42773614)

[**2.2.5.** **Kriptografi** 17](#_Toc42773615)

[**2.2.6.** **RSA (Riverst Shamir Adleman)** 18](#_Toc42773616)

[**BAB III** 21](#_Toc42773617)

[**ANALISIS DAN PERANCANGAN SISTEM** 21](#_Toc42773618)

[**3.1.** **Analisis Permasalahan** 21](#_Toc42773619)

[**3.2.** **Analisis Kebutuhan Sistem** 21](#_Toc42773620)

[**3.3.** **Model Rancangan Eksperimental Penelitian** 22](#_Toc42773621)

[**3.3.1.** **Bagan Rancangan Sistem** 22](#_Toc42773622)

[**3.3.2.** **Diagram Alir (*Flowchart*)** 24](#_Toc42773623)

[**BAB IV** 29](#_Toc42773624)

[**HASIL DAN PEMBAHASAN** 29](#_Toc42773625)

[**4.1.** **Tahap Implementasi** 29](#_Toc42773626)

[**4.1.1.** **Implementasi VirtualBox** 29](#_Toc42773627)

[**4.1.2.** **Implementasi Sistem Operasi Ubuntu** 30](#_Toc42773628)

[**4.1.3.** **Implementasi Metode RSA** 31](#_Toc42773629)

[**4.1.4.** **Implementasi Program Server dan *Client*** 35](#_Toc42773630)

[**4.2.** **Konfigurasi** 35](#_Toc42773631)

[**4.2.1** **Konfigurasi Virtual Machine pada Server** 35](#_Toc42773632)

[**4.2.2** **Konfigurasi Virtual Machine pada *Client*** 36](#_Toc42773633)

[**4.3.** **Pengujian Metode RSA** 37](#_Toc42773634)

[**4.3.1.** **Pengujian Pembentukan Kunci** 38](#_Toc42773635)

[**4.3.2.** **Pengujian Enkripsi** 39](#_Toc42773636)

[**4.3.3.** **Pengujian Dekripsi** 39](#_Toc42773637)

[**4.4.** **Pengujian Keamanan Metode RSA** 40](#_Toc42773638)

[**4.5.** **Pengujian Sistem** 40](#_Toc42773639)

[**BAB V** 45](#_Toc42773640)

[**SIMPULAN DAN SARAN** 45](#_Toc42773641)

[**5.1.** **Kesimpulan** 45](#_Toc42773642)

[**5.2.** **Saran** 45](#_Toc42773643)

[**DAFTAR PUSTAKA** 46](#_Toc42773644)

# **DAFTAR GAMBAR**

# **DAFTAR TABEL**

# **DAFTAR LAMPIRAN**

# **BAB I**

# **PENDAHULUAN**

* 1. **Latar Belakang**

*Internet of things* atau yang biasa disebut IoT merupakan sebuah konsep yang bertujuan untuk memperluas manfaat dari konektivitas internet dari perangkat – perangkat fisik disekitarnya dan dapat bertukar informasi antara satu perangkat dengan perangkat lainnya. Contoh dari IoT adalah kulkas dan televisi dimana benda – benda ini telah tertanam sebuah sensor yang digunakan untuk berkomunikasi dan berinteraksi dengan orang lain melalui internet dan dapat dikendalikan dari jarak jauh. Pada era ini, teknologi IoT lebih banyak dimanfaatkan dalam konsep *smart house* atau bahkan yang lebih besar lagi yaitu *smart city*. Namun terdapat ancaman serius terhadap teknologi IoT yaitu masalah privasi dan keamanan data dari penggunaan perangkat yang terhubung dengan internet. Salah satu contoh serangan yang dapat mengancam keamanan data adalah *Distributed Denial of Service* (DDoS). DDoS adalah sebuah percobaan penyerangan dari beberapa sistem komputer yang menargetkan sebuah server agar jumlah *traffic* menjadi terlalu tinggi sehingga server tidak dapat mengatasi *request* tersebut. Serangan ini memanfaatkan internet untuk mengambil alih komputer. Mirai merupakan *malware* yang mampu mengubah sistem komputer yang menjalankan linux menjadi pengendali bot dari jarak jauh. Bot dapat melakukan berbagai tugas seperti memindai kerentanan perangkat lain, mengirim pesan email spam, atau melakukan berbagai jenis serangan. Target utama dari *malware* ini yaitu perangkat IoT online seperti cctv dan *router* rumahan. Mirai mampu melakukan beberapa jenis serangan DDoS antara lain *SYN-flooding*, *UDP flooding*, *HTTP GET attacks*, *HTTP POST attack*, dan serangan lainnya.

Contoh kasus penyerangan terhadap port IoT adalah mirai botnet yang menyerang pengguna internet di eropa. Mirai merupakan salah satu jenis *worm* yang sebelumnya berhasil menginfeksi jutaan CCTV dan perangkat IoT di seluruh dunia. *Malware* ini menyerang *router* rumahan yaitu *router* buatan *Zyxel* dan *Speedport* dengan port 7547 yang terbuka. Biasanya, portini dipakai oleh penyedia internet untuk mengatur dan memperbaiki masalah router dari jarak jauh. Kode yang digunakan untuk menyerang *router* rumahan ini didapatkan dari versi modifikasi mirai sebelumnya. Oleh karena itu perlu dilakukan pengamanan pada port IoT. Banyak serangan dilakukan melalui port yang terbuka menjadi salah satu ancaman bagi keamanan data dalam sistem jaringan komputer. Jika orang– orang yang tidak memiliki hak akses dapat mengendalikan port yang dimasuki maka hal ini dapat menjadi ancaman. Maka dari itu penggunaan *firewall* sangat dibutuhkan di dalam jaringan tersebut. Dalam *firewall* semua komunikasi yang keluar dan masuk dikontrol. Port yang tidak penting dapat diblokir dan port yang penting dan berbahaya juga dapat diblokir, sehingga hanya pihak yang diijinkan yang boleh mengakses port tersebut. Akan tetapi dalam beberapa kondisi pemblokiran yang dilakukan sering menjadi tidak fleksibel karena pada saat dibutuhkan untuk menjalin sebuah komunikasi *firewall* tidak mengijinkannya karena berada pada area yang tidak diijinkan. Untuk menghindari hal seperti ini digunakan sebuah metode yaitu metode *port knocking*. Kelebihan dari metode ini adalah meskipun semua port yang ada telah ditutup, tetapi user yang memiliki hak akses dan mengetahui knocking untuk membuka suatu port dapat menggunakan port yang telah ia buka.

Namun *port knocking* masih memiliki kelemahan dalam pengamanan port yaitu penyerang akan sangat mudah mengetahui ketukan rahasia yang *valid* dengan melakukan *scanning* maupun *sniffing* karena format *port number* berupa *plain text*, adanya *DOS-Knocking* yaitu kondisi ketika penyerang mengirim paket secara terus-menerus dengan *random fake network address* kepada server yang menyebabkan meningkatnya penggunaan memori secara signifikan dan dapat mengakibatkan server *overload*. Karena metode ini masih memiliki kelemahan, maka diperlukan keamanan ganda pada port IoT yaitu dengan kriptografi. Kriptografi merupakan sebuah ilmu yang mempelajari cara mengamankan dan menjaga suatu data. Dalam kriptografi terdapat empat aspek penting dalam keamanan informasi antara lain kerahasiaan, integritas data, autentikasi, dan *non-repudiation*. Aspek – aspek keamanan inilah menjadi dasar bahwa perlu dilakukannya pengamanan dengan melakukan enkripsi.

Dalam penelitian ini, penulis akan menggunakan metode *port knocking* dalam pengamanan port IoT. Untuk mengastasi kelemahan dari metode ini, digunakan kriptografi sebagai pengamanan ganda pada port IoT. Metode ini disebut *encrypted* *port knocking*.

* 1. **Rumusan Masalah**

Berdasarkan dari latar belakang tersebut, dapat dirumuskan beberapa rumusan masalah yaitu:

1. Bagaimana meningkatkan keamanan *port knocking* dengan enkripsi?
2. Bagaimana rancangan *port knocking* dengan enkripsi?
   1. **Tujuan**

Berdasarkan dari latar belakang tersebut, dapat dirumuskan beberapa rumusan masalah yaitu:

1. Mengetahui peningkatan keamanan *port knocking*.

* 1. **Batasan Masalah**

Terdapat beberapa hal yang digunakan sebagai batasan permasalahan dalam penelitian ini yaitu :

1. Penelitian ini menggunakan metode RSA.
2. Penelitian ini dilakukan dengan simulasi menggunakan komputer.
3. Penelitian ini menggunakan port 22 sebagai port IoT.
4. Diasumsikan *public key* telah ditukar.
   1. **Manfaat**

Beberapa manfaat yang didapatkan dari penelitian ini yaitu:

* 1. Bagi penulis

Dari penelitian ini peneliti dapat menerapkan pengetahuan di bidang keamanan data pada IoT dan menjawab keingintahuan penulis mengenai masalah yang ditelitinya.

* 1. Bagi pihak lain

Hasil penelitian ini nantinya diharapkan dapat bermanfaat, menjadi referensi untuk menambah wawasan mengenai pengamanan data pada IoT.

* 1. **Metodelogi Penelitian**

Pada metodologi penelitian ini akan dijelaskan langkah – langkah dalam penelitian ini. Subbab bahasan yang akan dijelaskan meliputi desain penelitian, pengumpulan data, metode yang digunakan, serta pengujian.

* + 1. **Desain Penelitian**

Dalam mengatasi permasalahan mengenai pengamanan perangkat IoT, penulis menggunakan metode *port knocking* untuk pengamanan port dan melakukakan enkripsi guna meningkatkan keamanan terhadap port tersebut. Pada proses enkripsi digunakan metode RSA (*Riverst Shamir Adleman*). Langkah pertama yang dilakukan pada penelitian ini adalah melakukan pembangkitan *random port* dengan menggunakan *port knocking*. Sebelum *client* dapat mengakses perangkat IoT, maka akan dilakukan pembangkitan *random port* pada perangkat IoT kemudian hasil dari pembangkitan *random port* tersebut akan dikirimkan ke *client* sesuai dengan *rule por*t yang digunakan agar port tersebut dapat diketuk oleh *client* dan membuka *port* tujuan. Langkah kedua adalah proses enkripsi, pesan knock yang digunakan oleh client untuk memulai proses knocking akan dienkripsi oleh client. Kemudian hasil enkripsi dari pesan knock tersebut akan dikirimkan ke penerima yaitu server dan pada server dilakukan dekripsi cipher text yaitu pesan “knock” menjadi plain text atau pesan asli. Selanjutnya server akan melakukan pengecekan apakah pesan yang didekripsi benar, jika benar maka proses knocking akan dilakukan.

* + 1. **Pengumpulan Data**

Data yang digunakan pada penelitian ini diperoleh dari proses pembangkitan bilangan *random* yang digunakan untuk menentukan port dan jumlah port. Data ini digunakan sebagai rule untuk melakukan *knocking* pada port dengan metode *port knocking*. Hasil dari *random port* akan memberikan *rule port* yang berbeda pada saat jalannya *port knocking*.

Tabel contoh pembangkitan *random port*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 213 | 324 | 56 | 99 | 221 | 30281 | 77 | 6273 | 28743 | 18 |

Berdasarkan tabel tersebut, diperoleh *random port* dimana port tersebut digunakan sebagai syarat pengetukan sebelum port tujuan dapat terbuka. Port tujuan akan terbuka jika proses pengetukan pada port *random* tersebut telah selesai dilakukan.

* + 1. **Metode *Port knocking***

Penelitian ini menggunakan metode *port knocking* dimana dalam metode ini dilakukan pembangkitan bilangan *random* yang digunakan untuk menentukan port dan jumlah port. Pembangkitan bilangan *random* untuk port dilakukan dengan batasan tertentu yaitu 1 sampai 65000, sedangkan pembangkitan bilangan *random* untuk jumlah port dilakukan dengan batasan tertentu yaitu 1 sampai 1000. Server akan melakukan pembangkitan bilangan *random* dan mengirim hasil tersebut ke *client*. Kemudian, hasil dari pembangkitan bilangan *random* yaitu port dan jumlah port akan digunakan sebagai *rule* atau aturan yang digunakan *client* untuk proses *knocking* sebelum port tujuan dapat terbuka. Mekanisme metode *port knocking* memiliki beberapa tahap yaitu:

* + - * 1. Langkah pertama, client akan menjalankan program untuk port knocking. Client akan mengirim paket yang berisi pesan “get” yang berfungsi untuk menjalankan proses port knocking.
        2. Server akan menerima request dari client dan memulai pembangkitan bilangan acak untuk menentukan jumlah port dan nomor port yang harus di ketuk.
        3. Setelah dilakukan pembangkitan bilangan acak dan mendapatkan nomor port yang harus diketuk, nomor port tersebut akan dikirimkan oleh server menuju client satu persatu.
        4. *Client* menerima nomor port yang dikirimkan oleh server kemudian akan melakukan proses knocking. Untuk memulai proses knocking, *client* akan mengirim paket berupa enkripsi pesan “*knock*” yang berfungsi untuk menjalankan proses *port knocking*. Enkripsi pesan ini akan dikirimkan ke server.
        5. Selanjutnya server akan melakukan dekripsi terhadap cipher text pesan “knock” dan melakukan pengecekan terhadap pesan yang didekripsi. Jika bernilai benar, server akan menerima request *client* dan knocking sukses.
        6. Proses *knocking* yang dilakukan oleh *client* akan terus dilakukan sampai seluruh nomor port yang digunakan sebagai *rule* telah dikirimkan.
    1. **Metode RSA**

Pada penelitian ini, digunakan pula metode RSA (*Riverst Shamir Adleman*) untuk pengamanan pada perangkat IoT. Metode RSA memiliki 3 tahap yaitu pembentukan kunci, enkripsi dan dekripsi. Pada proses pembentukan kunci dilakukan pembangkitan bilangan *random* prima untuk menentukan nilai dari *public key* dan *private key*. Batasan nilai yang digunakan dalam pembangkitan bilangan *random* prima untuk pembentukan kunci adalah 0 sampai 1000. Kunci yang diperoleh akan digunakan pada saat proses enkripsi dan dekripsi. Pada proses enkripsi dilakukan konversi pesan dari *plaintext* (pesan asli) ke *ciphertext*. Sedangkan pada proses dekripsi dilakukan konversi pesan dari *ciphertext* ke *plaintext* (pesan asli). Enkripsi pesan dilakukan oleh pengirim pesan dengan menggunakan *public key* sedangkan dekripsi pesan dilakukan oleh penerima pesan dengan menggunakan *private key*. Berikut adalah langkah – langkah pengamanan dengan menggunakan metode RSA:

* + - * 1. Langkah pertama adalah client melakukan pembangkitan bilangan *random* prima untuk menentukan nilai p dan q.
        2. Kemudian, dilakukan perhitungan sesuai aturan untuk menentukan *public key* dan *private key*.
        3. Pesan “knock” yang digunakan oleh client untuk memulai knocking terhadap nomor port yang dikirimkan oleh server akan dienkripsi.
        4. Server akan melakukan dekripsi terhadap cipher text pesan “knock”, kemudian dilakukan pengecekan terhadap pesan yang didekripsi. Jika bernilai benar maka server akan menerima request *client* dan knocking sukses.
    1. **Skenario Pengujian**

Dalam penelitian ini dilakukan pengujian untuk mengetahui bahwa sistem yang dibuat dapat berjalan dengan baik sesuai dengan yang direncanakan sehingga mendapatkan hasil dari penelitian. Adapun skenario pengujian dilakukan dalam 2 tahap, yaitu:

Pengujian Algoritma Kriptografi

Pengujian ini dilakukan untuk menguji algoritma kriptografi yang digunakan. Pada penelitian ini, digunakan metode RSA dalam pengamanan data. Dalam metode RSA terdapat proses enkripsi dan dekripsi yang digunakan untuk mengamankan sebuah data. Enkripsi dan dekripsi inilah yang akan diuji untuk mengetahui keberhasilan dari penggunaan algoritma kriptografi.

Uji Coba Sistem

Pada pengujian ini akan dilakukan uji coba terhadap sistem yang dibuat dengan skenario komunikasi perangkat IoT. Untuk uji coba sistem akan menggunakan simulasi dengan virtual machine. Virtual machine yang digunakan pada penelitian ini yaitu VirtualBox dengan sistem operasi ubuntu.

# **BAB II**

# **TINJAUAN PUSTAKA**

* 1. **Tinjauan Pustaka Jurnal**

Dalam melaksanakan penelitian ini terdapat beberapa tinjauan pustaka junal yang digunakan sebagai pendukung terhadap penelitian yang dilakukan. Berikut tinjauan pustaka jurnal yang sudah diambil:

1. **Penerapan Sistem Pengamanan Port Pada Layanan Jaringan Menggunakan *Port knocking* (Suchendra, 2017)**

Penelitian ini membahas mengenai keamanan akses dan port dalam layanan jaringan. Permasalahan yang diperoleh adalah port yang terbuka atau akses yang tidak disertai dengan autentikasi dan otorisasi dapat mengakibatkan mudahnya *user* yang tidak memiliki kepentingan dapat mengakses sistem tersebut. Pengamanan port layanan jaringan yang terbuka ini menggunakan metode *port knocking* sebagai pengamanan untuk melakukan autentikasi sebelum mengakses server. Selain metode *port knocking*, digunakan juga metode *IP filter* dan fitur *packet timeout* pada sebuah *router*. Metode *port knocking* sangat diperlukan dalam pengamanan port karena diperlukannya proses autentikasi atau *knocking* untuk membuka port yang tertutup dengan cara mengakses beberapa port komunikasi sehingga port tujuan dapat terbuka. Metode *port knocking* dikombinasikan dengan *fitur* *packet timeout* dimana fitur ini akan memberi kesempatan waktu dalam proses autentikasi *port knocking*.

Persamaan dari penelitian ini adalah penggunaan metode *port knocking* yang digunakan untuk pengamanan port. Sedangkan perbedaanya adalah penambahan proses enkripsi dan dekripsi dengan menggunakan metode RSA. Penelitian yang dilakukan penulis akan menerapkan kriptografi untuk mengamankan nomor port yang dituju.

1. **Aplikasi Pengendalian Port dengan Utilitas *Port knocking* untuk Optimalisasi Sistem Keamanan Jaringan Komputer** (Muzawi, 2016)

Penelitian ini membahas mengenai lemahnya sistem keamanan jaringan pada STMIKAmik-Riau dimana *firewall* tidak mampu membedakan *user* yang dapat dipercaya. Cara kerja *firewall* adalah dengan menutup semua port tanpa memperdulikan apapun meskipun *user* memiliki hak untuk mengakses port tersebut. Oleh karena itu digunakan sebuah metode yaitu metode *port knocking* dimana metode ini memiliki keamanan yang baik dan memiliki kemampuan untuk mengizinkan user yang berhak untuk mengakses server. Pengimplementasian *port knocking* bertujuan untuk memberikan keamanan berlapis pada server dan memfilter *ip address* mana saja yang diperkenankan terkoneki terhadap server. Berdasarkan penelitian tersebut diperoleh hasil yaitu aplikasi pengendalian port dengan utilitas *port knocking* telah berhasil diuji dan menghasilkan kondisi dapat membuka dan menutup port tujuan yaitu port 22.

Persamaan dari penelitian ini adalah penggunaan metode *port knocking* yang digunakan untuk pengamanan port. Perbedaan dengan penelitian yang dilakukan oleh penulis adalah jumlah port dan *rule port* yang diketuk berbeda karena terdapat proses pembangkitan bilangan *random* sehingga *rule port* yang diperoleh akan bervariasi dan tidak tetap

1. **Implementasi Algoritma Kriptografi RSA Pada Surat Elektronik (E-Mail)** (Wahyadyatmika, 2014)

Penelitian ini membahas tentang suatu aplikasi yang mampu menjaga kerahasiaan dan keamanan proses pendistribusian informasi yaitu aplikasi email *client*. Aplikasi ini dirancang dengan menggunakan algoritma kriptografi RSA. Algoritma RSA membantu pengguna untuk melakukan proses enkripsi pesan sebelum dikirimkan dan melakukan proses dekripsi saat penerimaan pesan. Algoritma ini memiliki tiga proses yaitu proses pembentukan kunci untuk memperoleh pasangan *public key* dan *private key* kemudian proses enkripsi dan proses dekripsi terhadap data/informasi yang akan dtransmisikan. Berdasarkan penelitian yang dilakukan, aplikasi yang dibuat menunjukkan bahwa algoritma kriptografi RSA sangat baik untuk mengatasi masalah manajemen distribusi kunci yaitu dengan menyimpan pasangan kunci pada basis data sedangkan kunci yang di distribusikan hanya *public key*.

Persamaan dari penelitian yang penulis lakukan adalah pengamanan data/informasi dengan menggunakan algoritma kriptografi RSA. Sedangkan perbedaan antara penelitian ini dengan penelitian yang penulis lakukan adalah batasan nilai dalam pembangkitan bilangan *random* prima yang dilakukan oleh sistem untuk memperoleh nilai p dan q yang akan digunakan dalam proses pembangkitan kunci. Batasan nilai tersebut adalah dari 0 – 1000.

1. **Implementasi Kriptografi Modern Dengan Metode RSA Pada Data Citra Digital** (Deskiva, 2018)

Pada penelitian ini membahas permasalahan mengenai aplikasi yang dapat digunakan untuk mengamankan data citra *digital*. Metode yang digunakan yaitu metode asimetris dengan algoritma RSA. Algoritma RSA digunakan dalam mengamankan data citra digital dimana data yang diamankan berupa gambar. Pada aplikasi ini terdapat proses pembangkitan kunci, enkripsi pada data citra digital dan proses dekripsi pada gambar tersebut. Berdasarkan penelitian ini diperoleh hasil yaitu algoritma kriptografi RSA dapat diimplementasikan pada penyandian data citra digital dengan mengubah citra menjadi bentuk nilai *pixel-pixel*. Selain itu proses enkripsi dan dekripsi membutuhkan waktu sesuai dengan ukuran *pixel* gambar dimana semakin besar jumlah *pixel* gambar tersebut maka akan semakin lama proses enkripsi dan dekripsi selesai.

Persamaan dari penelitian yang penulis lakukan adalah pengamanan data/informasi dengan menggunakan algoritma kriptografi RSA. Perbedaan penelitian ini dengan penelitian yang penulis lakukan yaitu terdapat pada proses pembangkitan kunci dimana pada penelitian yang penulis lakukan untuk mencari nilai p dan q, sistem melakukan pembangkitan bilangan *random* prima bukan dengan cara manual. Pembangkitan bilangan *random* prima ini akan mendapatkan nilai prima untuk p dan q. Selain itu nilai e dan d juga ditentukan oleh sistem.

1. **Pengujian Sistem Enkripsi-Dekripsi Dengan Metode RSA Untuk Pengamanan Dokumen** (Supriyono, 2008)

Penelitian ini membahas tentang suatu sistem yang dapat mengamankan proses penyimpanan dan pengiriman dokumen. Dokumen dalam bentuk teks akan dienkripsi sehingga dokumen tidak dapat dibaca oleh siapapun karena dokumen atau teks tersebut telah diubah menjadi bentuk *ciphertext*. Agar dokumen dapat dibaca oleh penerima maka *cipher*text tersebut harus diubah menjadi dokumen asli (*plaintext*), proses ini disebut dekripsi. Sistem pengamanan dokumen ini menggunakan metode RSA. Dalam pembuatan sistem ini terdapat beberapa proses pengamanan dengan menggunakan metode RSA yaitu pembangkitan kunci, enkripsi, dan dekripsi. Berdasarkan penelitian ini diperoleh hasil yaitu sistem dapat menyimpan dan mengirimkan dokumen baik pengiriman melalui internet maupun intranet dalam bentuk susunan huruf yang terenkripsi dan mengembalikan ke bentuk dokumen semula dengan cara dekripsi.

Persamaan dari penelitian yang penulis lakukan adalah pengamanan data/informasi dengan menggunakan algoritma kriptografi RSA. Perbedaan penelitian ini dengan penelitian yang penulis lakukan yaitu pada batasan nilai saat proses pembangkitan kunci dimana nilai yang ditetapkan oleh penulis untuk menentukan nilai p dan q menggunakan pembangkitan bilangan *random* prima dengan Batasan nilai yaitu 0 sampai 1000.

1. **Implementasi Asymmetric Encryption RSA Pada Port Knocking Ubuntu Server Menggunakan Knockd Dan Python** (Amir, 2020)

Pada penelitian ini membahas mengenai beberapa kelemahan dari autentikasi metode port knocking yang membuat metode ini masih rentan terhadap banyak serangan. Beberapa serangan tersebut adalah TCP replay dan bruteforce. Oleh karena itu, penulis mengusulkan metode autentikasi dan end to end connection untuk menambah keamanan pada metode port knocking. Autentikasi yang dilakukan oleh client pada server bertujuan untuk validasi client yang melakukan pengetukan sehingga dapat menghindari dari serangan bruteforce. Kemudian ditambahkan end to end koneksi yang bertujuan untuk meningkatkan keamanan port knocking. Selain itu dengan adanya penambahan tunneling antara client dan server memungkinkan menambah untuk keamanan port knocking itu sendiri. Berdasarkan penelitian ini diperoleh hasil yaitu *port knocking* memiliki potensi pengembangan yang cukup baik terutama dibagian pengamanan *sequence* dan timeout serta autentikasi yang terjadi setelah melakukan *knocking*, dalam penelitian ini metode asymmetric enctyption RSA digunakan untuk autentikasi dari *server* ke *client* dan sebaliknya, tidak hanya itu proses enkrip dan dekrip dilakukan dengan waktu yang paling cepat, sehingga proses *knocking* tidak membutuhkan waktu yang lama

Persamaan dari penelitian yang penulis lakukan adalah pengamanan data/informasi dengan menggabungkan metode port knocking dan algoritma kriptografi RSA. Perbedaan penelitian dengan yang penulis lakukan adalah penggunaan VPN tunnel agar client dan server berada dalam satu jaringan private. Untuk melakukan jaringan private harus menggunakan username dan password yang ditentukan oleh server yang mana dibuat pada saat konvigurasi VPN pada server. Sedangkan pada penelitian yang penulis lakukan tidak menggunakan VPN tunnel dalam pegerjaan penelitian ini.

* 1. **Tinjauan Pustaka Teori**
     1. ***Internet of things***

*Internet of things* atau yang dikenal dengan IoT pertama kali diperkenalkan oleh Kevin Ashton pada tahun 1999 dan mulai terkenal melalui *Auto-ID Center* di [MIT](http://id.wikipedia.org/wiki/MIT). Meski telah diperkenalkan sejak 15 tahun yang lalu, hingga kini belum ada sebuah konsensus global mengenai definisi IoT. Menurut *Coordinator and support action for global RFID-related activities and standadisation* menyatakan *Internet of things* (IoT) sebagai sebuah infrastruktur koneksi jaringan *global*, yang mengkoneksikan benda fisik dan *virtual* melalui eksploitasi *data* *capture* dan teknologi komunikasi. Selain itu, menurut IEEE (*Institute of Electrical and Electronics Engineers*) *Internet of things* (IoT) didefinisikan sebagai sebuah jaringan dengan masing-masing benda yang ternanam dengan sensor yang terhubung kedalam jaringan internet. Dari beberapa pernyataan dapat disimpulkan secara umum konsep IoT diartikan sebagai sebuah kemampuan untuk menghubungkan objek-objek cerdas dan memungkinkannya untuk berinteraksi dengan objek lain, lingkungan maupun dengan peralatan komputasi cerdas lainnya melalui jaringan internet. Adapun contoh dari IoT adalah monitor glukosa yang terkoneksi pada pasien diabetes, alat ini bertujuan untuk memudahkan dokter dalam menerima data pasien secara *real time*, memonitor kondisi pasien dan menyesuaikan dosis obat.

* + 1. **Port**

Dalam protokol jaringan [TCP/IP](https://id.wikipedia.org/wiki/TCP/IP), sebuah port adalah mekanisme yang mengizinkan sebuah komputer untuk mendukung beberapa sesi koneksi dengan komputer lainnya dan program di dalam jaringan. Port dapat mengidentifikasikan aplikasi dan layanan yang menggunakan koneksi di dalam jaringan TCP/IP. Selain itu, port juga mengidentifikasikan sebuah proses tertentu di mana sebuah server dapat memberikan sebuah layanan kepada klien atau bagaimana sebuah klien dapat mengakses sebuah layanan yang ada dalam server. Port dapat dikenali dengan angka [16-bit](https://id.wikipedia.org/w/index.php?title=16-bit&action=edit&redlink=1) (dua *byte*) yang disebut dengan *Port Number* dan diklasifikasikan dengan jenis protokol *transport* apa yang digunakan, ke dalam Port TCP dan Port UDP. Karena memiliki angka 16-bit, maka total maksimum jumlah port untuk setiap protokol *transport* yang digunakan adalah 65536 buah. Berdasarkan penomorannya, port UDP dan TCP dibagi menjadi tiga jenis yaitu:

* *Well-Known Port*

Ditetapkan oleh  [*Internet Assigned Number Authority* (IANA)](https://id.wikipedia.org/wiki/IANA) dimana *port number* yang termasuk ke dalam jenis ini selalu merepresentasikan layanan jaringan yang sama. *Range well-known port* diperlebar menjadi berkisar dari 0 hingga 1023.

* *Registered Port*

Merupakan port yang digunakan oleh vendor komputer untuk mendukung aplikasi dan sistem operasi yang dibuat. *Registered port* tidak dialokasikan secara permanen oleh IANA sehingga *port number* dapat digunakan lebih dari satu vendor. *Range registered port* berkisar dari 1024 hingga 49151.

* *Dynamically Assigned Port*

Merupakan port yang digunakan untuk melayani *request* dari pengguna sesuai dengan kebutuhan. *Range  Dynamically Assigned Port* berkisar dari 1024 hingga 65536.

* + 1. ***Firewall***

*Firewall* adalah sebuah sistem atau perangkat yang mengizinkan lalu lintas jaringan yang dianggap aman untuk melaluinya dan mencegah lalu lintas jaringan yang tidak aman. *Firewall* memiliki tugas untuk melakukan pemblokiran terhadap port-port komunikasi yang terbuka bebas dalam sebuah jaringan komputer. Pada *firewall* semua komunikasi yang keluar dan masuk dikontrol. Port yang tidak penting dapat diblokir (ditutup), port yang penting namun berbahaya juga dapat diblokir sehingga hanya pihak yang diijinkan saja yang boleh masuk melalui port tersebut. Cara tersebut merupakan sistem pengamanan jaringan komputer yang paling efektif dan banyak digunakan. Akan tetapi terkadang pemblokiran yang dilakukan sering menjadi tidak fleksibel, ketika dibutuhkan untuk menjalin komunikasi dengan apa yang ada di dalam jaringan, *firewall* tidak mengijinkannya karena berada pada area yang tidak diijinkan. *Firewall* berfungi untuk mencatat semua kejadian di jaringan dan melaporkan kepada administrator, melakukan autentikasi terhadap akses, melindungi sumber daya dalam jaringan privat, serta mengontrol dan mengatur lalu lintas jaringan. Umumnya, sebuah *firewall* diterapkan dalam sebuah mesin terdedikasi, yang berjalan pada pintu gerbang (*gateway*) antara jaringan lokal dan jaringan lainnya. Parameter proteksi dari *firewall* adalah *IP address*, *domain name*, *protocol*, dan port. *Firewall* terbagi menjadi dua jenis yaitu

* *Personal* *Firewall*

Digunakan untuk melindungi sebuah komputer yang terhubung ke jaringan dari akses yang tidak dikehendaki. Personal *Firewall* merupakan sebuah kumpulan program yang bertujuan untuk mengamankan komputer secara total, dengan ditambahkannya beberapa fitur pengaman tambahan semacam perangkat proteksi terhadap virus, *anti-spyware*, *anti-spam*, dan lainnya. *Personal* *Firewall* secara umum hanya memiliki dua fitur utama, yaitu *Packet Filter* *Firewall* dan *Stateful* *Firewall*.

* *Network Firewall*

Digunakan untuk melindungi jaringan secara keseluruhan dari berbagai serangan. *Network* *Firewall* dapat berupa sebuah perangkat terdedikasi dan sebagai sebuah perangkat lunak yang diinstalasikan dalam sebuah server. *Network* *Firewall* memiliki beberapa fitur utama yaitu *packet filter firewall*, *stateful firewall, Circuit* *Level Gateway*, *Application Level Gateway*, dan juga *NAT Firewall*. *Firewall* jenis ini tidak terlihat oleh pengguna dan menggunakan teknologi *routing* untuk menentukan paket mana yang diizinkan, dan mana paket yang akan ditolak

Cara kerja *Firewall* adalah mengecek *header* dari paket data yang masuk ke jaringan kemudian menentukan apakah paket data ini bisa diteruskan atau tidak sesuai dengan aturan jaringan. Jika tidak maka paket data tersebut akan diblokir namun jika diijinkan maka paket data ini akan diteruskan sesuai mekanisme jaringan tersebut. Pemblokiran dan penolakan pada paket data terjadi apabila tidak memiliki akses, berasal dari jaringan internal ke internet contohnya pengguna internet mengakses situs terlarang, memiliki konten yang tidak diijinkan seperti file yang memiliki virus.

* + 1. ***Port knocking***

*Port knocking* adalah metode yang digunakan untuk membuka akses ke port tertentu yang telah di *block* oleh *firewall* pada perangkat jaringan dengan cara mengirimkan paket atau koneksi tertentu. Koneksi yang dikirim dapat berupa *protocol TCP* dan *UDP*. Port hanya akan terbuka jika menggunakan *sequence of request* untuk nomor port yang telah ditentukan. Dengan menggunakan metode ini, maka perangkat jaringan akan lebih aman karena terdapat proses *filtering* terhadap port – port yang ada. Pada aplikasinya, jika *user* tidak memiliki akses ke port maka pada saat dilakukannya *port scanning* terhadap port – port yang ada maka port – port tersebut tidak akan dapat terlihat atau port tersebut terlihat tertutup. Mekanisme metode *port knocking* ini memiliki beberapa tahap, sebagai berikut:

* + - 1. Pada tahap pertama, klien melakukan koneksi ke komputer server ke salah satu port di komputer server, misal port 22, namun koneksi tersebut di blok oleh *firewall* komputer server
      2. Klien melakukan koneksi ke port – port *sequences* yang telah didefinisikan dalam file konfigurasi *daemon* *port knocking* ke komputer server dengan mengirimkan paket SYN didalamnya. Selama fase ini, klien tidak akan mendapatkan respon apa – apa
      3. Selanjutnya *Daemon* *Port knocking* mencatat adanya percobaan koneksi dan kemudian melakukan autentikasi terhadap percobaan tersebut. Apabila autentikasi sesuai dengan yang didefinisikan pada *daemon* *port knocking* dalam hal ini adalah *port sequences* yang didefinisikan, maka *daemon* *port knocking* akan melakukan *overwrite* terhadap *rule* yang telah didefinisikan didalam *firewall* agar membuka port yang ingin dituju oleh klien.
      4. Setelah melakukan autentikasi klien dapat melakukan koneksi ke port yang dituju menggunakan aplikasi seperti pada umumnya.
      5. Setelah selesai, klien memutuskan koneksi dengan port dan kemudian mengirimkan paket SYN kembali agar daemon *port knocking* menulis ulang *rule* pada *firewall* agar tidak bisa dilakukan koneksi kembali ke port 22.
    1. **Kriptografi**

Kriptograﬁ (bahasa yunani "*Cryptos*" yaitu rahasia dan "*graphein*" yaitu tulisan) merupakan tulisan rahasia. Kriptograﬁ merupakan sebuah ilmu yang mempelajari cara mengamankan dan menjaga suatu data. Terdapat Istilah istilah yang umum digunakan dalam kriptograﬁ yaitu *plaintext* (M) merupakan pesan yang akan dikirim (berupa data asli), *ciphertext* (C) merupakan pesan tersandi dan hasil dari enkripsi, enkripsi merupakan proses pengubahan *plaintext* menjadi *ciphertext*, dekripsi merupakan proses yang mengubah *ciphertext* menjadi *plaintext* yang awalnya berupa data yang berbeda bentuk dari data awal menjadi data awal/asli, serta kunci merupakan suatu bilangan yang dirahasiakan dan digunakan dalam proses enkripsi serta dekripsi. Dalam kriptografi terdapat 4 aspek keamanan informasi yaitu:

* *Confidentiality* (Kerahasiaan)

Kerahasiaan adalah layanan yang digunakan untuk menjaga informasi dari setiap pihak yang tidak berwenang untuk mengaksesnya. Dengan demikian informasi hanya akan dapat diakses oleh pihak-pihak yang berhak saja.

* Integritas Data

Integritas berhubungan dengan penjagaan dari perubahan data secara tidak sah. Untuk menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi terjadinya manipulasi data. Manipulasi data yang dimaksud di sini meliputi penyisipan, penghapusan, maupun penggantian data.

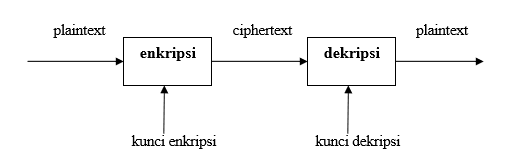
* *Authentication*

*Authentication* berhubungan dengan identifikasi/pengenalan baik secara kesatuan sistem maupun informasi itu sendiri. Informasi yang dikirimkan harus diautentikasi keaslian isi data, waktu pengiriman dan lainnya.

* *Non-repudiation*

*Non-repudiation* adalah usaha untuk mencegah terjadinya penyangkalan terhadap pengiriman suatu informasi oleh yang mengirim atau yang membuat.

Pada dasarnya algoritma kriptograﬁ dibedakan menjadi dua jenis jika dilihat berdasarkan jenis kunci yang digunakan antara lain algoritma simetris dan algoritma asimetris. Algoritma asimetris menggunakan kunci yang berbeda pada saat proses enkripsi dan proses dekripsi. Dalam algoritma simetris terdapat dua jenis kunci yaitu *public key* dan kunci rahasia. Sedangkan algoritma simetris menggunakan kunci yang sama untuk proses enkripsi serta proses dekripsi. Dalam kriptograﬁ sendiri terdapat dua hal penting yaitu proses enkripsi dan proses dekripsi. Proses enkripsi merupakan proses dimana informasi atau data yang dikirim diubah menjadi bentuk yang tidak diketahui dengan menerapkan algoritma tertentu. Data yang bisa dimengerti biasa disebut dengan *plaintext* dan data yang bentuknya berbeda dan tidak dapat dimengerti biasa disebut *cipher*teks. Sedangkan proses dekripsi adalah proses dimana data atau informasi yang sebelumnya telah diubah menjadi bentuk yang tidak diketahui diubah kembali menjadi data atau informasi awal yang disampaikan. Berikut adalah diagram proses dari enkripsi dan dekripsi.



Gambar Proses enkripsi dan dekripsi

* + 1. **RSA (Riverst Shamir Adleman)**

Metode RSA diciptakan oleh 3 orang peneliti dari *Massachussets Institute of Technology* (MIT) yaitu Ron Rivest, Adi Shamir dan Leonard Adleman pada tahun 1977. Dalam metode ini terdapat 3 tahap dalam melakukan enkripsi antara lain proses pembentukan kunci, enkripsi, dan dekripsi.

1. **Pembangkitan Kunci**

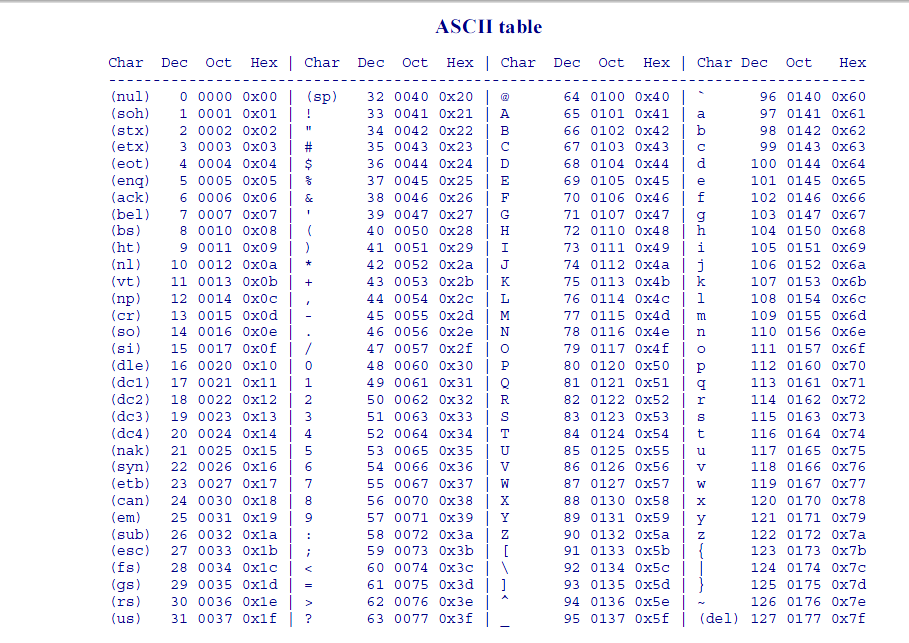
Pada proses pembentukan kunci akan mengkasilkan 2 buah kunci yang berbeda untuk digunakan pada saat enkripsi dan dekripsi. Kunci yang dihasilkan adalah *public key* dan *private key*. Dalam metode RSA terdapat beberapa besaran yang digunakan antara lain:

1. p dan q bilangan prima (rahasia)
2. n = p×q (tidak rahasia)
3. φ(n) = (p−1)×(q−1) (rahasia)
4. e (kunci enkripsi) (tidak rahasia)
5. d (kunci dekripsi) (rahasia)
6. m (*plaintext*) (rahasia)
7. c (*ciphertext*) (tidak rahasia)

Adapun langkah – langkah perhitungan terkait pembangkitan kunci pada metode RSA antara lain:

1. Menentukan 2 bilangan prima, dengan nama p dan q.
2. Menghitung nilai modulus (n) dengan rumus n = p×q.
3. Menghitung nilai *totient* atau *phi*(φ) dari n dengan rumus φ(n) = (p - 1) (q - 1)
4. Menentukan nilai e dimana nilai e ini merupakan bilangan prima dan nilai e harus sesuai dengan syarat 1< e < φ(n). Untuk pembuktian terhadap nilai e dapat dilakukan perhitungan dengan rumus gcd (e, φ(n)) = 1.
5. Mencari nilai *deciphering exponent* (d) dengan rumus d = (1 + (k×φ(n))/e
6. Setelah menemukan nilai n, e, dan d maka didapatkan pasangan kunci yaitu pasangan *public key* dan pasangan kunci rahasia. Pasangan *public key* (n,e) dan pasangan *private key*(n,d).
7. **Enkripsi**

Pada proses ini dilakukan konversi pesan dari *plaintext* ke dalam kode ASCII dengan menggunakan tabel ASCII untuk melihat kode sesuai dengan *plaintext* yang ada. Selanjutnya adalah mencari nilai C dengan menggunakan rumus . Kunci yang digunakan pada proses ini yaitu *public key* (n,e). Dari rumus perhitungan tersebut maka akan ditemukan nilai dari c (*ciphertext*). Berikut adalah gambar dari tabel ASCII



Gambar tabel ASCII

1. **Dekripsi**

Pada proses ini akan mengubah *ciphertext* yang telah diperoleh sebelumnya menjadi data awal yaitu berupa *plaintex*t. Proses dekripsi dilakukan dengan menggunakan rumus Setelah ditemukan nilai m dengan menggunakan rumus perhitungan untuk dekripsi maka akan ditemukan pula *plaintext* dari pesan yang dikirim

# **BAB III**

# **ANALISIS DAN PERANCANGAN SISTEM**

* 1. **Analisis Permasalahan**

Internet of things atau yang dikenal sebagai IoT merupakan sebuah konsep yang bertujuan untuk memperluas manfaat konektivitas internet dari perangkat – perangkat fisik disekitarnya dan dapat bertukar informasi antara satu perangkat dengan perangkat lainnya. Pada era ini, teknologi IoT lebih banyak dimanfaatkan dalam konsep *smart house* atau bahkan yang lebih besar yaitu *smart city*. Namun terdapat ancaman serius terhadap teknologi IoT yaitu permasalahan privasi dan keamanan data dari penggunaan perangkat yang terhubung dengan internet. Salah satu contoh kasus penyerangan terhadap keamanan data pada perangkat IoT adalah mirai botnet. Mirai dapat menyerang perangkat IoT online seperti cctv dan *router* rumahan. Cara kerja malware ini adalah dengan menyerang port yang digunakan oleh penyedia internet untuk mengatur dan memperbaiki masalah router dari jarak jauh. Oleh karena itu perlu dilakukan pengamanan pada port IoT. Banyak serangan dilakukan melalui port yang terbuka menjadi salah satu ancaman bagi keamanan data dalam sistem jaringan komputer. Jika orang– orang yang tidak memiliki hak akses dapat mengendalikan port yang dimasuki maka hal ini dapat menjadi sebuah ancaman.

* 1. **Analisis Kebutuhan Sistem**

Berdasarkan dari analisis permasalahan, dapat ditemukan analisis kebutuhan sistem yang dibangun adalah melakukan pengamanan pada perangkat IoT dengan *encrypted port knocking*. *Encrypted port knocking* merupakan sebuahmetode pengamanan ganda pada port IoT dengan mengunakan metode *port knocking* dalam pengamanan port IoT dan kriptografi. Pembangkitan bilangan acak pada *port knocking* dilakukan untuk memperoleh jumlah port dan nomor port. Pada kriptografi, digunakan metode *Riverst Shamir Adleman* (RSA). Metode RSA ini digunakan untuk melakukan enkripsi dan dekripsi terhadap pesan “knock” yang digunakan client untuk memulai knocking pada nomor port yang dikirim oleh server. Sistem pengujian menggunakan 2 buah komputer yaitu server dan *client*. Komputer server digunakan sebagai perangkat IoT. Komputer server dan *client* akan disimulasikan menggunakan *Virtual Machine* dengan spesifikasi seperti berikut:

Tabel Analisis Kebutuhan Sistem

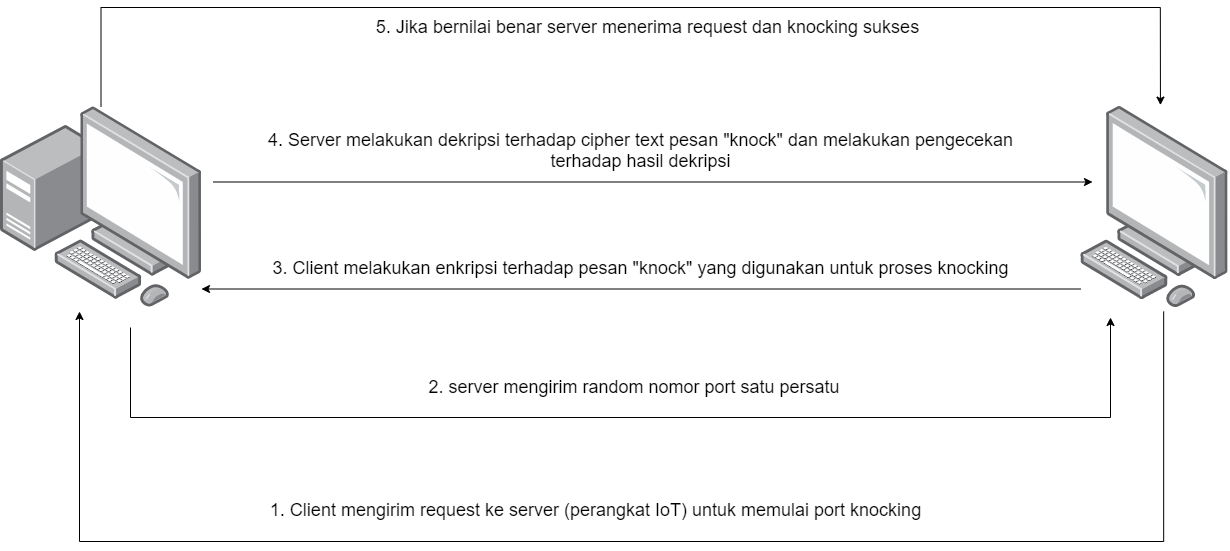
|  |  |  |  |
| --- | --- | --- | --- |
| No | Spesifikasi | Sistem Operasi | Keterangan |
| 1 | *Virtual Machine*, RAM 1GB, Kapasitas HDD 10GB | Ubuntu Dekstop 16.04.3 LTS | Bertindak sebagai Server (Perangkat IoT) |
| 2 | *Virtual Machine*, RAM 1GB, Kapasitas HDD 10GB | Ubuntu Dekstop 16.04.3 LTS | Bertindak sebagai *Client* |

* 1. **Model Rancangan Eksperimental Penelitian**

Pada penelitian ini, penulis menggunakan bagan rancangan sistem dan diagram alir (*flowchart*) untuk menggambarkan alur dari program yang dibuat.

* + 1. **Bagan Rancangan Sistem**

Berikut adalah gambar dari bagan rancangan sistem yang akan dibuat



Gambar Bagan Rancangan Sistem

Gambar 4.1 merupakan sebuah bagan rancangan sistem yang akan dibuat pada penelitian ini. Pada bagan tersebut dapat dilihat terdapat 2 buah komputer yang akan digunakan pada saat pengujian sistem yaitu server (perangkat IoT) dan *Client*. Server dan *Client* memiliki tugasnya masing – masing, antara lain:

1. *Client*

*Client* memiliki fungsi untuk mengirim *request* ke server. *Request* yang dikirim oleh *client* digunakan untuk memperoleh jumlah port dan *encrypted port*. Setelah server mengirimkan nomor port, maka client dapat melakukan knocking terhadap nomor port tersebut. Untuk memulai knocking, client harus mengirimkan request knock agar dapat melakukan knocking terhadap port tersebut. Pesan “knock” akan dienkripsi dan hasil enkripsi akan dikirimkan ke server. Jika server telah menerima request knocking yang dilakukan oleh client, maka knocking sukses. Proses knocking akan terus dilakukan hingga seluruh nomor port yang dibangkitkan pada server telah dikirimkan semuanya menuju client dan port tujuan terbuka.

1. Server

Server merupakan perangkat IoT yang memiliki fungsi untuk menerima *request* yang dilakukan oleh *client*. *Request* yang diterima oleh server digunakan untuk memulai proses *port knocking*. Server akan memulai proses pembangkitan bilangan acak untuk menentukan jumlah port dan nomor port. Server akan menerima cipher text pesan “knock” dari client dan melakukan dekripsi terhadap pesan tersebut. Pesan “knock” merupakan pesan request yang digunakan oleh client untuk melakukan knocking. Kemudian dilakukan pengecekan terhadap hasil dekripsi yang diperoleh. Jika hasil dekripsi bernilai sama dengan pesan “knock” maka proses knocking dapat dilakukan.

Berikut adalah proses pengetukan pada port:

1. *Client* melakukan koneksi terhadap server (perangkat IoT).
2. Setelah terkoneksi, *client* mengirim *request* menuju server untuk memperoleh jumlah port dan random nomor port.
3. Server menerima *request* dari *client* kemudian melakukan proses pembangkitan bilangan acak. Dari pembangkitan bilangan acak diperoleh jumlah port dan nomor port yang harus dituju.
4. Selanjutnya server akan mengirim random nomor port satu persatu menuju client.
5. Client akan melakukan pengetukan terhadap nomor port yang diperoleh dari server. Untuk memulai knocking atau pengetukan, client harus mengirimkan request knock agar dapat melakukan knocking terhadap port tersebut. Pesan “knock” akan dienkripsi dan dikirimkan ke server.
6. Server akan menerima cipher text pesan “knock” dari client dan melakukan dekripsi terhadap pesan tersebut.
7. Kemudian dilakukan pengecekan terhadap hasil dekripsi yang diperoleh. Jika hasil dekripsi bernilai sama dengan pesan “knock” maka proses knocking dapat dilakukan.
8. *Client* akan malakukan *knocking* sampai seluruh nomor port yang dikirimkan oleh server habis.
9. Proses *port knocking* selesai, port tujuan terbuka.
   * 1. **Diagram Alir (*Flowchart*)**

Untuk lebih menjelaskan mengenai alur program pada penelitian ini, berikut merupakan *flowchart* untuk program:

1. *Flowchart* *Port knocking* pada Server



Gambar *Flowchart* *port knocking* pada server

Berikut adalah penjelasan dari *flowchart* server:

1. Pertama server akan menutup *ip* *client* tertentu untuk membuka port.
2. Server akan berada pada kondisi *listening* untuk siap menerima input dari *client*.
3. Server menerima *request* dari *client* dan diuji apakah *request* yang diterima adalah untuk *port knocking* atau HTTP/HTTPS *request*. Jika HTTP/HTTPS *request* yang diterima maka HTTP/HTTPS *response* dikirimkan oleh server ke *client* dan kembali ke keadaan *listen*.
4. Jika iya maka server menginisialisasi n\_port = random()
5. Lalu memproses i = 0.
6. Lalu masuk ke dalam proses *generated random port*, apa i = n\_port, jika tidak maka random disimpan pada port *array* di index i dan terus berulang sampai *array* yang ditentukan
7. Status pada server *not valid* karena masih belum terjadi proses pengetukan
8. Proses i = 0
9. Masuk ke kondisi pengetukan port i = n\_port , jika tidak maka port pada *array* di index i dikirimkan ke *client*
10. Kemudian port akan dienkripsi menjadi bentuk *cipher*
11. *Cipher* *port* dan *generated token* akan dikirimkan ke *client*
12. Server kembali ke keadaan *listen*
13. Server mengambil data di *client*
14. Server akan melakukan dekripsi port menjadi bentuk *plain port*
15. Jika ketukan benar maka proses pengetukan akan berulang sebanyak port yang ada pada *array* di index i.
16. Jika tidak maka server mengirimkan status *not valid* ke *client* dan kembali ke keadaan *listen* port 80
17. Jika iya i = n\_port maka buka port tujuan (*setting firewall*)
18. *Flowchart* proses tutup port pada server



Gambar *Flowchart* menutup port pada server

Penjelasan *flowchart* proses menutup port pada server

1. Pada saat menutup port 22, Server dalam keadaan *listen*ing
2. Kemudian server menerima *request* untuk menutup port dari *client*
3. Jika server menerima *request* untuk menutup maka port 22 akan ditutup. Namun jika tidak maka akan kembali dalam keadaan *listen*ing.
4. *Flowchart* pengetukan oleh *client*



Gambar *Flowchart* pengetukan oleh *client*

Berikut adalah penjelasan *flowchart* diatas:

1. Pada proses pertama *client* akan me*request* menuju ke server
2. Jika iya *client* menerima *generated random port*
3. Jika tidak apa HTTP/HTTPS *request* ?
4. Jika iya kirim HTTP/HTTPS *response* dan kembali ke *client* *request*.
5. Jika tidak maka langsung kembali ke *client* *request*
6. *Client* melakukan *encrypted generated port*
7. Port di dekripsi menjadi bentuk *plain port*
8. *Client* melakukan melakukan proses pengetukan
9. Jika ketukan benar maka *client* akan lanjut mengetuk port, jika tidak maka proses pengetukan selesai.
10. Jika port masih dikirim maka proses *request* port akan terus berulang sampai port yang dikirim oleh server habis dan jika tidak maka port IoT terbuka
11. *Flowchart* menutup port pada *client*



Gambar *Flowchart* menutup port pada *client*

Berikut adalah penjelasan dari flowchat penutupan port:

1. *Client* ingin menutup port IoT yang terbuka yaitu port 22
2. Jika *client* ingin menutup maka *client* akan me*request* untuk menutup port 22 (dilakukan pada setting firewall)
3. Port 22 tertutup.

# **BAB IV**

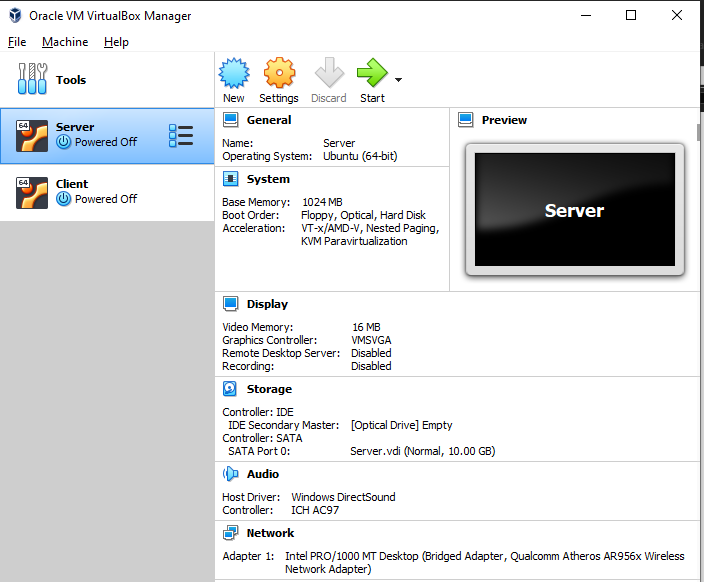
# **HASIL DAN PEMBAHASAN**

* 1. **Tahap Implementasi**

Pada sub bab ini dilakukan instalasi terhadap aplikasi yang mendukung serta implementasi dari program – program yang digunakan untuk penelitian ini.

1. **Implementasi VirtualBox**

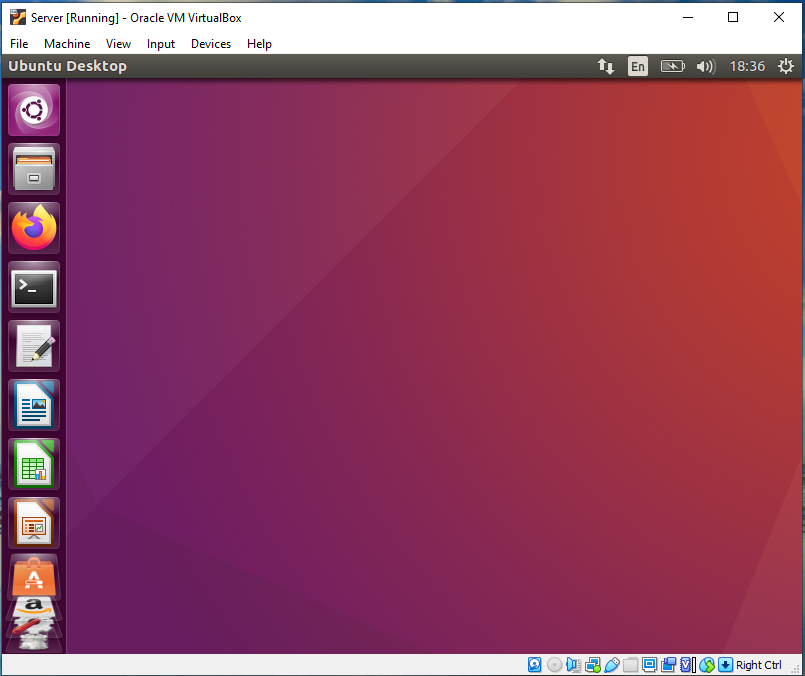
Langkah pertama adalah melakukan instalasi terhadap virtual machine yang akan digunakan. Virtual machine yang digunakan pada penelitian ini adalah VirtualBox. Virtual machine berfungsi sebagai media untuk menyimpan sistem operasi yang digunakan pada penelitian ini yaitu ubuntu. Dalam VirtualBox terdapat 2 sistem operasi ubuntu yaitu server dan *client*.



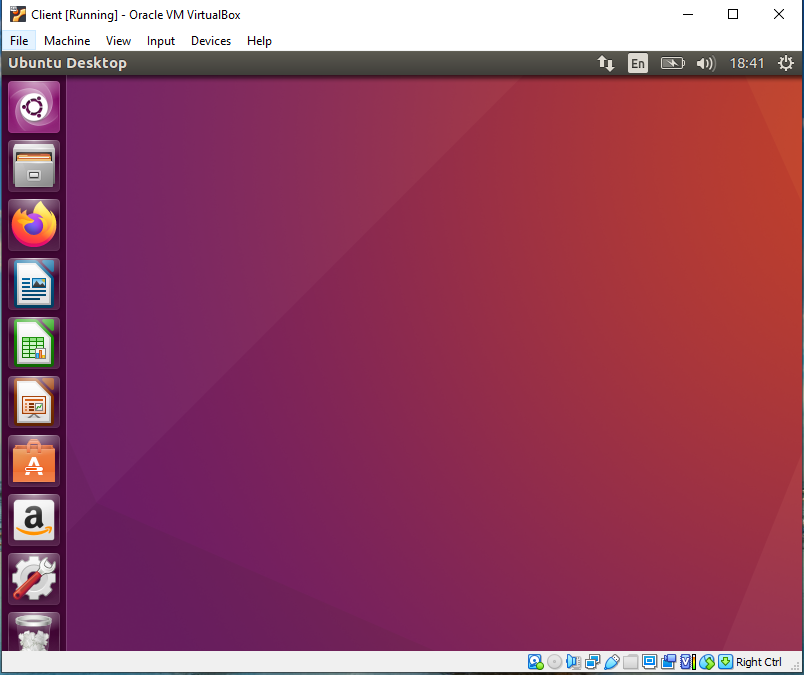
Gambar Virtual Machine

1. **Implementasi Sistem Operasi Ubuntu**

Sistem operasi yang digunakan pada implementasi sistem adalah ubuntu 16.04.3 LTS. Sistem operasi ubuntu ini digunakan pada server (perangkat IoT) dan *client*.



Gambar ubuntu 16.04.3 LTS pada server



Gambar ubuntu 16.04.3 LTS pada *client*

1. **Implementasi Metode RSA**

Proses Encrypted Port yang dilakukan pada penelitian ini menggunakan metode RSA. Pemrograman yang digunakan dalam pengerjaan encrypted port knocking menggunakan bahasa c dan diimplementasikan pada operasi sistem ubuntu. Pada implementasi metode ini, dibagi menjadi 3 bagian yaitu pembangkitan kunci, enkripsi, dan dekripsi.

1. **Implementasi Pembangkitan Kunci**

Pada tahap implementasi ini, untuk memperoleh nilai prima digunakan pembangkitan bilangan random prima. Pembangkitan bilangan random prima ini difungsikan untuk memperoleh nila p dan q yang digunakan dalam perhitungan untuk mencari kunci. Hasil dari proses pembangkitan kunci akan menghasilkan 2 buah kunci public (n,e) dan kunci private (n,d). Source code dibawah ini merupakan code untuk proses pembangkitan kunci yang nantinya kunci – kunci yang diperoleh akan digunakan pada tahap enkripsi dan dekripsi.

Tabel Source Code pembangkitan bilangan random prima

void random\_prima(int p, int q)

{

int i,number[1000]={0},temp,j,status=0, prime[1000]={0},r;

for(i=2;i<=1000;i++)

{

for(j=i+1;j<=1000;j++)

{

if(j % i == 0 && number[j] == 0)

{

number[j] = 1;

}

}

}

temp = 0;

for(i=2;i<=1000;i++)

{

if(number[i] == 0)

{

prime[temp++] = i;

}

}

srand(time(NULL));

do

{

r = rand()%1000;

}

while(prime[r] == 0);

p = prime[r];

do

{

r = rand()%1000;

}

while(prime[r] == 0 || prime[r] == p);

{

q = prime[r];

}

printf("Bilangan Prima 1 : %d",p);

printf("\nBilangan Prima 2 : %d",q);

n=p\*q; // menghitung nilai modulus (n)

printf("\nn = %d \n",n);

t=(p-1)\*(q-1); //mencari nilai totient/phi(n)

}

Tabel Source Code mencari nilai e dan d

void ce()

{

int k;

k=0;

for (i=2;i<t;i++)

{

if(t%i==0)

continue;

flag=prime(i);

if(flag==1&&i!=p&&i!=q)

{

e[k]=i; //mencari nilai e

flag=cd(e[k]);

if(flag>0)

{

d[k]=flag; //mencari nilai deciphering exponent (d)

k++;

}

if(k==99)

break;

}

}

}

1. **Implementasi Enkripsi**

Source code dibawah ini merupakan code untuk melakukan enkripsi. Pada tahap enkripsi ini digunakan kunci public yang telah didapatkan pada proses pembangkitan kunci. Pesan yang dienkripsi akan dikonversi kedalam kode ASCII untuk memperoleh nilai cipherteks. Berikut adalah code dari implementasi enkripsi.

Tabel Source Code enkripsi

void encrypt()

{

int pt,ct,key=e[0],k,len;

i=0;

len=strlen(pesan);

while(i!=len)

{

pt=m[i]; //menampung nilai message ke variabel pt

pt=pt-96;

k=1;

for (j=0;j<key;j++)

{

k=k\*pt;

k=k%n;

}

temp[i]=k;

ct=k+96;

en[i]=ct;

i++;

}

en[i]=-1;

printf("\n----------------------------------------\n");

FILE \*filea;

filea=fopen("enkripsi.txt","w");

fprintf(filea,"%s",en);

fclose(filea);

}

1. **Implementasi Dekripsi**

Pada implementasi dekripsi, pesan berupa cipherteks yang diperoleh akan didekripsi menjadi bentuk pesan asli (plainteks). Dalam proses dekripsi digunakan kunci private yang didapatkan pada proses pembangkitan kunci.

Tabel Source Code enkripsi

void decrypt()

{

long int pt,ct,key=d[0],k;

i=0;

while(en[i]!=-1)

{

ct=temp[i];

k=1;

for (j=0;j<key;j++)

{

k=k\*ct;

k=k%n;

}

pt=k+96;

m[i]=pt;

i++;

}

m[i]=-1;

printf("\nNILAI DEKRIPSI\n");

for (i=0;m[i]!=-1;i++)

{

printf("%c",m[i]);

}

printf("\n");

}

1. **Implementasi Program Server dan *Client***

Dalam implementasi client dan server, terdapat 2 buah program yang digunakan pada penelitian ini. Program tersebut dijalankan pada 2 sistem operasi yang berbeda, yaitu server dan client.

* + - 1. **Program Client**

Program client digunakan untuk membuka port IoT tujuan dan melakukan request ke server. Client akan melakukan request terhadap server untuk memperoleh jumlah port dan random nomor port. Setelah request diterima oleh server, maka random nomor port akan dikirim satu persatu menuju client. Client akan menerima random port tersebut dan akan melakukan knocking. Untuk memulai proses knocking, Client harus melakukan request ke server yaitu dengan pesan “knock”. Pesan “knock” akan dienkripsi dan dikirimkan ke server. Proses pengetukan akan terjadi apabila server menerima request tersebut. Port IoT dapat terbuka apabila client telah selesai melakukan proses knocking.

* + - 1. **Program Server**

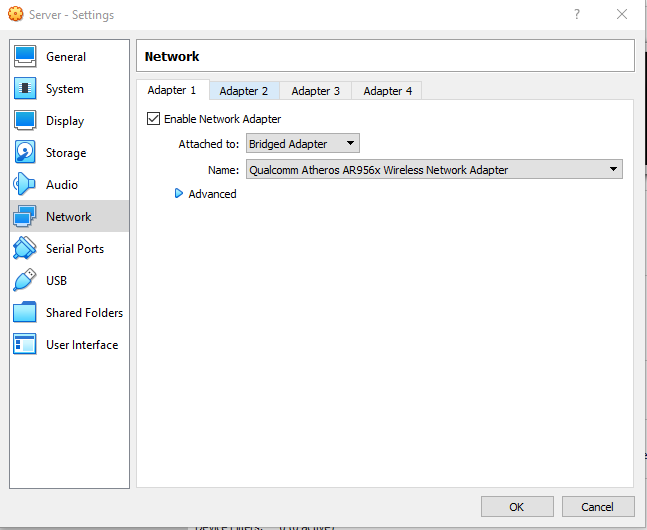
Pada penelitian ini, program server digunakan untuk menerima request dari client. Selanjutnya server akan merespon dengan melakukan pembangkitan bilangan acak sehingga menghasilkan nomor port dan jumlah port. Server akan menerima cipher text pesan “knock” dari client dan melakukan dekripsi terhadap pesan tersebut. Pesan “knock” merupakan pesan request yang digunakan oleh client untuk melakukan knocking. Kemudian dilakukan pengecekan terhadap hasil dekripsi yang diperoleh. Jika hasil dekripsi bernilai sama dengan pesan “knock” maka proses knocking dapat dilakukan.

* 1. **Konfigurasi**

Pada tahap ini dilakukan konfigurasi pada server dan client. Konfigurasi dilakukan dengan menyesuaikan terhadap kebutuhan server dan client.

1. **Konfigurasi Virtual Machine pada Server**

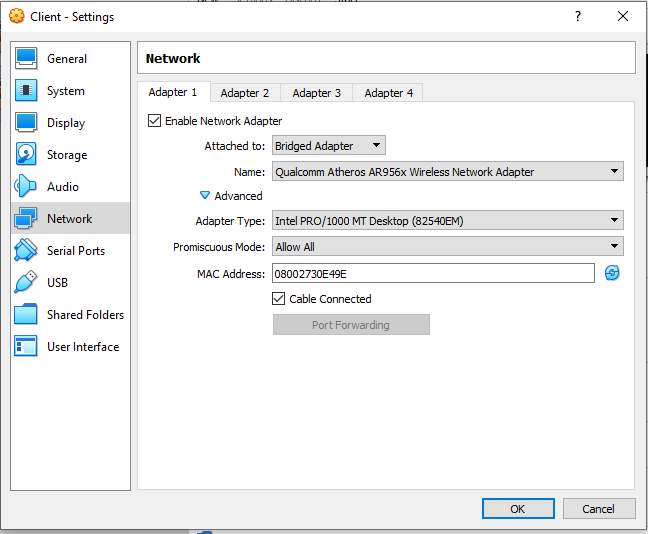
Pada server dilakukan konfigurasi dengan menggunakan network interface dan mengatur ke dalam mode bridged adapter. Tujuan pengaturan mode ini yaitu agar server dapat terhubung ke client sehingga terbentuk koneksi antara server dan client. Koneksi diperlukan karena server dan client berada pada komputer yang berbeda.



Gambar Konfigurasi Virtual Machine pada Server

1. **Konfigurasi Virtual Machine pada *Client***

Pada client, pengaturan konfigurasi yang digunakan yaitu network interface dengan menggunakan mode Bridge Adaptor. Pengaturan mode tersebut berfungsi agar server dan client dapat terkoneksi satu sama lain walaupun berada pada 2 komputer yang berbeda.



Gambar Konfigurasi Virtual Machine pada Client

* 1. **Pengujian Metode RSA**

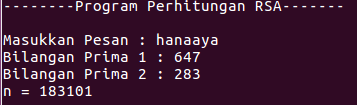
Pengujian pada metode RSA dilakukan untuk menguji hasil akhir enkripsi yaitu chiperteks dan hasil akhir dari dekripsi yaitu plaintext. Tujuan dilakukannya pengujian ini adalah untuk membuktikan bahwa pesan yang dieknkripsi (cipherteks) dapat kembali menjadi pesan asli (plainteks). Adapun data yang digunakan untuk pengujian ini dapat dilihat pada tabel

Tabel Data Pengujian

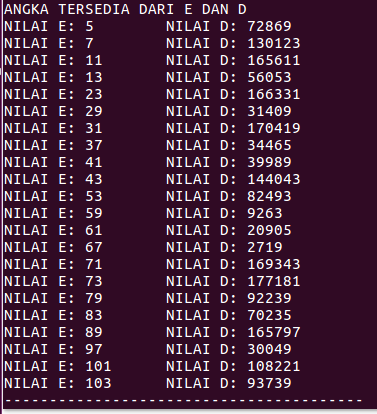
|  |  |
| --- | --- |
| Keterangan | Nilai |
| Pesan asli (Plainteks) | hanaaya |
| Kunci Public (n,e) | (183101,5) |
| Kunci Private (n,d) | (183101,72869) |

1. **Pengujian Pembentukan Kunci**

Langkah pertama yang harus dilakukan dalam pengujian metode RSA adalah proses pembangkitan kunci. Pembangkitan kunci dilakukan untuk memperoleh kunci yang akan digunakan pada saat enkripsi dan dekripsi pesan. Pada proses ini akan diperoleh 2 buah kunci yaitu kunci public (n,e) dan kunci private (n,d). Berikut adalah hasil pembentukan kunci yang dilakukan oleh sistem.



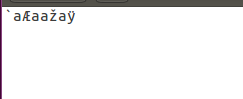
Gambar (a) Proses Pembentukan Kunci Program RSA



Gambar (b) Proses Pembentukan Kunci Program RSA

1. **Pengujian Enkripsi**

Langkah selanjutnya adalah melakukan proses enkripsi. Pada proses ini, dilakukan konversi pesan dari plainteks ke dalam bentuk cipherteks dengan menggunakan kode ASCII. Setelah memperoleh kode ASCII dari masing – masing karakter yang ada pada plainteks, dilakukan perhitungan untuk mencari nilai C dengan menggunakan rumus . Nilai e dan n diperoleh dari proses pembangkitan kunci yang dilakukan sebelumnya yaitu kunci public (n,e). Nilai e diambil dari array e[0] dan digunakan pada kunci public. Dari rumus perhitungan tersebut maka akan ditemukan nilai dari C (cipherteks). Cipherteks yang dihasilkan oleh sistem diexport ke dalam file enkripsi.txt. Berikut adalah gambar hasil enkrispi yang dilakukan oleh sistem.



Gambar hasil enkripsi oleh sistem

1. **Pengujian Dekripsi**

Langkah terakhir dalam pengujian metode RSA adalah proses dekripsi. Pada proses ini, cipherteks yang telah diperoleh pada proses enkripsi akan diubah menjadi pesan asli (plainteks). Rumus perhitungan yang digunakan untuk proses dekripsi adalah. Nilai n dan nilai d yang digunakan pada dekripsi pesan diperoleh dari proses pembangkitan kunci yaitu kunci private (n,d). Nilai deciphering exponent diambil dari array d[0] dan digunakan sebagai nilai d pada kunci private. Dari rumus perhitungan tersebut akan ditemukan nilai M yaitu nilai dari karakter plainteks. Kemudian nilai M akan diubah sesuai dengan tabel ASCII sehingga diperoleh pesan asli (plainteks). Berikut adalah gambar hasil dekripsi yang dilakukan oleh sistem.



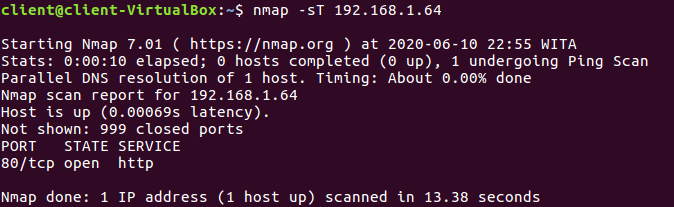
Gambar hasil dekripsi oleh sistem

* 1. **Pengujian Keamanan Metode RSA**
  2. **Pengujian Sistem**

Pengujian Sistem pada penelitian ini dilakukan untuk mengetahui bahwa sistem yang dibuat dapat berjalan dengan baik sesuai dengan yang diharapkan. Pengujian ini dilakukan pada 2 kondisi yaitu pada komputer yang menggunakan port knocking dan pada komputer yang menggunakan penggabungan port knocking dan kriptografi yaitu metode RSA. Hasil dari pengujian ini adalah untuk mengetahui keberhasilan dalam pengamanan data antara komputer yang menggunakan port knocking dengan rsa dan komputer yang hanya menggunakan port knocking. Pengujian sistem dilakukan berdasarkan langkah – langkah berikut yaitu:

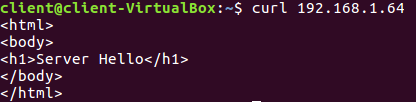
1. Pengujian Komputer dengan Port Knocking

Pada program server, akan menerima request dari client dengan berkomunikasi melalui port 80. Dalam proses port knocking ini, komunkasi yang terjadi antara server dan client akan melalui port tersebut. Oleh karena itu dilakukan scanning port untuk mengetahui status port.



Gambar proses scanning port dengan nmap

Selanjutnya adalah melakukan curl command terhadap server. Curl merupakan singkatan dari client URL, yang berfungsi untuk mengecek konektivitas ke URL dan juga sebagai tool transfer data. Curl Command pada pengujian ini digunakan untuk mengecek konektivitas yang terjadi antara server dan client.



Gambar proses curl server pada client

Kemudian pada komputer client, dibuat sebuah file untuk menampung isi pesan yang digunakan sebagai pesan request pada proses port knocking. Lalu pesan tersebut akan ditulis ke dalam file yang telah dibuat.

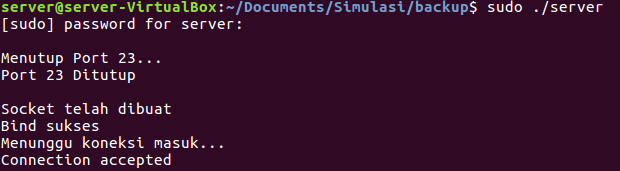


Gambar proses pembuatan file pada client



Gambar proses menuliskan pesan get pada client

Selanjutnya adalah pengujian dengan menggunakan tool netcat. Pada pengujian ini diilakukan percobaan untuk masuk ke server melalui port 80. Sebelum melakukan pengujian, server harus dalam keadaan listening agar client dapat melakukan request.



Gambar Server dalam Kondisi Listening

Setelah server siap untuk menerima request dari client, maka langkah selanjutnya adalah client melakukan request agar proses port knocking dapat berjalan. Proses tersebut dilakukan dengan menggunakan tools netcat melalui port 80.

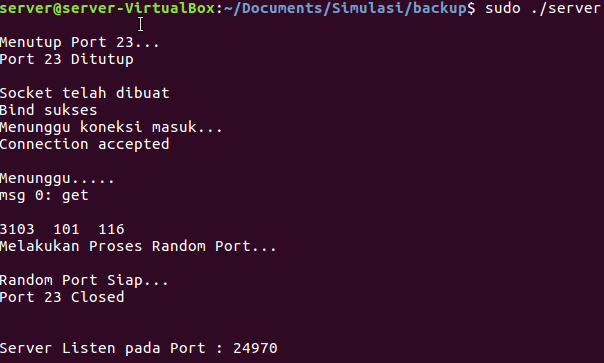


Client melakukan koneksi ke server melalui port 80

Client akan memasukan pesan yaitu “get” yang merupakan perintah untuk melakukan proses port knocking. Ketika server menerima request dari client, maka server akan melakukan random dan mengirimkan hasil random port tersebut ke client.



Client melakukan request ke server



Server mengirim random port

Setelah server mengirimkan random port ke client, maka server akan kembali pada kondisi listening. Proses listening ini menunjukan bahwa server siap untuk menerima pengetukan yang dilakukan oleh client terhadap port yang telah dikirim sebelumnya.

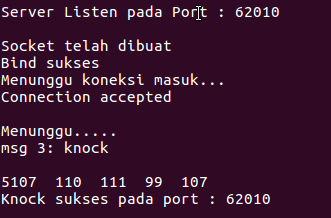


Client melakukan knocking terhadap random port yang diperolehnya

Jika pengetukan yang dilakukan client berhasil, selanjutnya server akan mengirimkan nomor port lain yang harus diketuk oleh client.



Client berhasil melakukan pengetukan

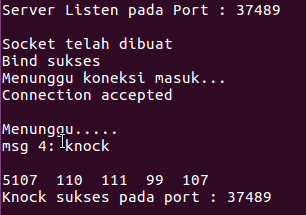


Server mengirimkan random port

Kemudian saat client berhasil melakukan pengetukan, maka server akan kembali mengirimkan nomor port selanjutnya yang harus diketuk



Client berhasil melakukan pengetukan



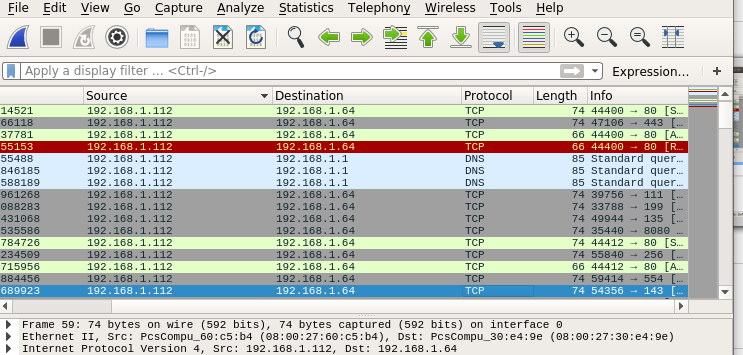
Server mengirimkan random port

Selanjutnya, ketika client berhasil melakukan knocking terhadap random port yang dikirim oleh server maka server akan mengirimkan random port selanjutnya yang harus diketuk oleh client. Proses knocking atau pengetukan yang terjadi antara server dan client akan terus berjalan hingga seluruh random port yang diperoleh telah habis dikirim menuju client.



Client berhasil melakukan pengetukan

Untuk megetahui proses port knocking yang dilakukan berjalan dengan baik, digunakan tools lain untuk menguji sistem ini. Tools tersebut adalah Wireshark. Wireshark dapat digunakan untuk melakukan analisis paket data pada jaringan secara real time dan melihat detail traffic. Berikut adalah hasil capture paket data yang dilakukan dengan tools ini untuk menguji sistem dapat berjalan dengan baik.



Gambar hasil capture paket oleh wireshark

# **BAB V**

# **SIMPULAN DAN SARAN**

* 1. **Kesimpulan**

Berdasarkan penelitian yang telah dilakukan diperoleh beberapa kesimpulan, antara lain yaitu:

* 1. **Saran**

# **DAFTAR PUSTAKA**

1. Suchendra, D. R., Rahman, A. F., & Ismail, S. J. I. (2017). PENERAPAN SISTEM PENGAMANAN PORT PADA LAYANAN JARINGAN MENGGUNAKAN *PORT KNOCKING*. *Jurnal Komputer Bisnis*, *10*(2).
2. Muzawi, R. (2016). Aplikasi Pengendalian Port dengan Utilitas *Port knocking* untuk Optimalisasi Sistem Keamanan Jaringan Komputer. *SATIN-Sains dan Teknologi Informasi*, *2*(1), 52-58.
3. Wahyadyatmika, A. P., Isnanto, R. R., & Somantri, M. (2014). Implementasi Algoritma Kriptografi RSA pada Surat Elektronik (E-Mail). *TRANSIENT*, *3*(4), 442-450.
4. Deskiva, Z. Z. (2018). Implementasi Kriptografi Modern dengan Metode Rsa pada Data Citra Digital. *Publikasi Ilmiah Teknologi Informasi Neumann*, *3*(1), 44-49.
5. Supriyono, S. (2008, November). PENGUJIAN SISTEM ENKRIPSI-DEKRIPSI DENGAN METODE RSA UNTUK PENGAMANAN DOKUMEN. In *Jurnal Forum Nuklir* (Vol. 2, No. 2, pp. 179-194).
6. Setiadi, D., & Muhaemin, M. N. A. (2018). Penerapan Internet Of Things (IoT) Pada Sistem Monitoring Irigasi (smart irigasi). *Infotronik: Jurnal Teknologi Informasi dan Elektronika*, *3*(2), 95-102.
7. Amir, Z., Syaifuddin, S., & Risqiwati, D. (2020). Implementasi Asymmetric Encryption RSA Pada Port Knocking Ubuntu Server Menggunakan Knockd Dan Python. *Jurnal Repositor*, *2*(6), 787-794.
8. Meutia, E. D. (2015). Internet of things–Keamanan dan Privasi. In *Seminar Nasional dan Expo Teknik Elektro* (pp. 85-89).
9. Ginting, A., Isnanto, R. R., & Windasari, I. P. (2015). Implementasi Algoritma Kriptografi RSA untuk Enkripsi dan Dekripsi Email. *Jurnal Teknologi dan Sistem Komputer*, *3*(2), 253-258.
10. Reeves, M. (2014). Port Knocking with Single Packet Authentication using Asymmetric Key Cryptography. *Journal of Information Assurance & Security*, *9*(1).
11. Stallings, W. (2006). *Cryptography and network security, 4/E*. Pearson Education India.

**LAMPIRAN**

**Lampiran 1. Source Code Program**

* + - 1. Source Code Program enkripsi dan dekripsi dengan metode RSA

#include<stdio.h>

#include<ncurses.h>

#include<stdlib.h>

#include<math.h>

#include<string.h>

#include <time.h>

long int flag,e[100],d[100],temp[100],j,i,pr,k;

char m[100], en[100], pesan[100];

int p, q, n, t;

int prime(long int);

long int cd(long int);

void ce();

void random\_prima();

void encrypt();

void decrypt();

int main()

{

printf("--------Program Perhitungan RSA-------\n\n");

//---------------message yg akan dienkripsi--------------------

printf("Masukkan Pesan : ");

scanf("%s",pesan);

for (i=0;pesan[i]!='\0';i++)

{

m[i]=pesan[i]; // menampung nilai yang sudah ada dlam pesan

}

random\_prima(); //memanggil f

ce(); //cek nilai enkripsi

printf("\nANGKA TERSEDIA DARI E DAN D");

for (i=0;i<j-1;i++)

{

printf("\nNILAI E: %ld\t NILAI D: %ld",e[i],d[i]);

}

encrypt();

decrypt();

getch();

}

int prime(long int pr)

{

int i;

j=sqrt(pr);

for (i=2;i<=j;i++)

{

if(pr%i==0)

return 0;

}

return 1;

}

void ce()

{

int k;

k=0;

for (i=2;i<t;i++)

{

if(t%i==0)

continue;

flag=prime(i);

if(flag==1&&i!=p&&i!=q)

{

e[k]=i; //nilai enkripsi

flag=cd(e[k]);

if(flag>0)

{

d[k]=flag; //nilai dekripsi

k++;

}

if(k==99)

break;

}

}

}

long int cd(long int x)

{

long int k=1;

while(1)

{

k=k+t;

if(k%x==0)

return(k/x);

}

}

//------------------enkripsi pesan----------------------

void encrypt()

{

int pt,ct,key=e[0],k,len;

i=0;

len=strlen(pesan); //menghitung panjang karakter dalam suatu string

while(i!=len) // program akan selesai jika nilai len sudah berakhir

{

pt=m[i]; //menampung nilai message ke variabel pt

pt=pt-96;

k=1;

for (j=0;j<key;j++)

{

k=k\*pt;

k=k%n;

}

temp[i]=k; //menampung nilai k

ct=k+96;

en[i]=ct;

i++;

}

en[i]=-1;

printf("\n----------------------------------------\n");

FILE \*filea;

filea=fopen("enkripsi.txt","w");

/\*for (i=0;en[i]!=-1;i++)

{

printf("%c",en[i]);

//fprintf(filea,"%c",en[i]);

}\*/

fprintf(filea,"%s",en);

fclose(filea);

}

//------------------dekripsi pesan----------------------

void decrypt()

{

long int pt,ct,key=d[0],k;

i=0;

while(en[i]!=-1)

{

ct=temp[i];

k=1;

for (j=0;j<key;j++)

{

k=k\*ct;

k=k%n;

}

pt=k+96;

m[i]=pt;

i++;

}

m[i]=-1;

printf("\nNILAI DEKRIPSI\n");

for (i=0;m[i]!=-1;i++)

{

printf("%c",m[i]);

}

printf("\n");

}

void random\_prima(int p, int q)

{

int i,number[1000]={0},temp,j,status=0, prime[1000]={0},r;

for(i=2;i<=1000;i++)

{

for(j=i+1;j<=1000;j++)

{

if(j % i == 0 && number[j] == 0)

{

number[j] = 1;

}

}

}

temp = 0;

for(i=2;i<=1000;i++)

{

if(number[i] == 0)

{

prime[temp++] = i;

}

}

srand(time(NULL));

do

{

r = rand()%1000;

}

while(prime[r] == 0);

p = prime[r];

do

{

r = rand()%1000;

}

while(prime[r] == 0 || prime[r] == p);

{

q = prime[r];

}

printf("Bilangan Prima 1 : %d",p);

printf("\nBilangan Prima 2 : %d",q);

n=p\*q; // perkalian pembangkitan kunci

printf("\nn = %d \n",n);

t=(p-1)\*(q-1); //nilai untuk mencari relatif prima kunci

}

* + - 1. Source Code Program Server
      2. Source Code Program *Client*