Implementasi Algoritma RSA pada Interkoneksi Jaringan IPv6 dan IPv4 dengan Mekanisme Tunneling Mode 6to4

I Dewa Gede W.[1], Indrarini Dyah I.[1] dan Tody Ariefianto W.[1]

ABSTRACT: *IPv6 to IPv4 transitions mode is a solution to change from IPv4 to IPv6, one of "IPv6 to IPv4" transitions is 6to4. But, there are security considerations to use 6to4 that the routers must accept and forward IPv4 packets from any other router. Thus, IPv6 to IPv4 transition mode on the public network may have some vulnerability, because of no threat detection like sniffing, and spoofing. On network security there are various algorithm for message encryptions and decrytions, one of them is RSA (Rivest Shamir Adleman) algorithm. RSA algorithm is asymmetryc crypthographic algorithm with different key on encryption and decryption. RSA still used on electronic commerce protocol and securing with a long key, demanding a long time to crack the key. On this research, implementation of RSA algorithm for securing data and communicating between client and server on 6to4 will be analyzed. Results from the research are, when SSH server send 10 MB data the server need 64.1172 ms on delay while FTP server need 39.6879 ms. But, the SSH server more secure than FTP server. Because, when logging in to SSH server the password, username, and data contains are encrypted while logging in to FTP server the password, username, and data contains can be sniffing by attacker. On CPU-uttility analisys, transfer using SSH causing the average CPU usage is 15.19355 %, while it's 1.333333% on FTP server (without encryption).*
KEYWORDS: *spoofing, sniffing, RSA, delay*

ABSTRAK: Mode transisi IPv6 ke IPv4 adalah salah satu solusi untuk melakukan perubahan IPv4 ke IPv6, salah satu modenya adalah 6to4. Namun ada pertimbangan keamanan dalam menggunakan 6to4 yaitu setiap router 6to4 harus menerima dan meneruskan paket dari *native* IPv6 maupun dari router 6to4 lainnya. Sehingga, mode transisi IPv6 ke IPv4 yang digunakan pada jaringan public mungkin meninggalkan celah karena tidak adanya deteksi terhadap suatu serangan seperti *sniffing,* dan *spoffing*. Dalam keamanan jaringan dikenal berbagai algoritma untuk melakukan enkripsi dan dekripsi pesan, salah satunya adalah RSA (Rivest Shamir Adleman). Algoritma RSA merupakan algoritma kriptografi *asimetry,* dimana kunci enkripsi tidak sama dengan kunci dekripsinya. RSA masih digunakan secara luas dalam protokol *electronic commerce*, dan dipercaya dalam mengamankan dengan menggunakan kunci yang cukup panjang. Dan untuk memecahkan kunci yang cukup panjang tersebut, memerlukan waktu yang cukup lama. Dalam Tugas Akhir ini dianalisa bagaimana penggunaan algoritma RSA untuk mengamankan data dan komunikasi antara *client* dan *server* pada jaringan 6to4 yang masih rentan terhadap pencurian data. Dari hasil analisa yang dilakukan, saat mengirimkan data sebesar 10 MB *delay* SSH *server* adalah 64,1172 ms sedangkan *delay* FTP *server* adalah 39,6879 ms. Namun, keamanan data teks yang dilewatkan pada SSH masih lebih baik dari FTP karena pada saat melakukan *long* ke *server* SSH, *password, username,* dan isi data sudah terenkripsi. berbeda halnya ketika menggunakan FTP, semua komunikasi dapat diketahui (di-*sniffing*) oleh *attacker*. Selain itu dengan adanya enkripsi dengan algoritma RSA menyebabkan utilitas CPU menjadi lebih besar yaitu 15,19355 % pada SSH *server*, sedangkan 1,333333 % pada FTP *server* (tanpa enkripsi).
KATA KUNCI: spoofing, sniffing, RSA, delay .

PENDAHULUAN

Perkembangan dunia internet semakin hari semakin meningkat dan jumlah penggunanya juga semakin banyak, hal ini berdampak pada semakin berkurangnya alamat IP yang masih menggunakan IPv4. Salah satu solusi untuk mengatasi hal tersebut adalah melakukan interkoneksi jaringan IPv6 dan IPv4 dengan *tunneling* 6to4, karena infrastruktur yang tersedia sekarang menyulitkan untuk merubah sekaligus IPv4 ke IPv6. Dalam RFC 3964 [6]dijelaskan bahwa ada 2 karakteristik 6to4 yang menjadi pertimbangan keamanan yaitu: (1) semua 6to4 router harus menerima dan mendekapsulasi paket yang diterima, (2) 6to4 router harus menerima setiap traffic yang dikirimkan oleh native ipv6. Kedua karakteristik 6to4 tersebut mungkin menjadi celah untuk menyerang jaringan 6to4, karena memungkinkan adanya serangan *spoffing, man in the middle attack*, serta *sniffing*. Maka diperlukan enkripsi agar data yang dilewatkan masih tetap terjaga kerahasiaanya Salah satu cara untuk melakukan pengamanan data adalah dengan melakukan enkripsi pada saat transfer data. Dalam dunia kriptografi dikenal berbagai algoritma yang digunakan untuk enkripsi/dekripsi sebuah pesan, salah satunya adalah algoritma RSA (Rivest Shamir Adleman).

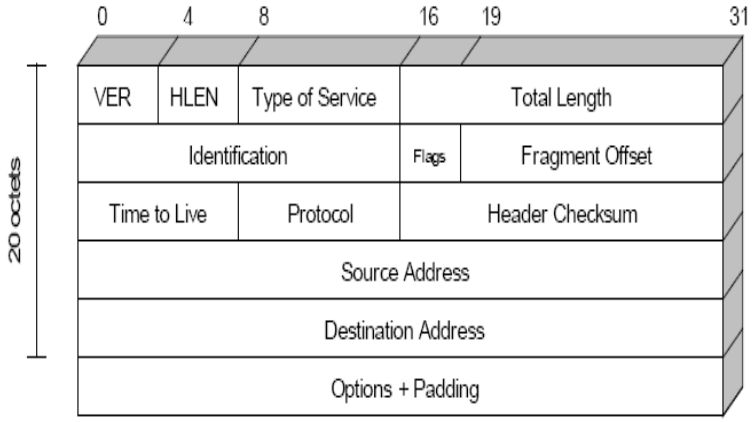
Manfaat dari penilitian ini adalah untuk mengetahui akibat yang ditimbulkan oleh penambahan algoritma RSA pada *tunnelling* 6to4 dengan meninjau keamanannya dan menganalisa *delay* yang disebabkan adanya enkripsi data dan mengetahui kekuatan dari algoritma RSA.

INTERNET PROTOKOL

IP merupakan suatu mekanisme transmisi yang digunakan oleh protokol-protokol TCP/IP, dimana IP bersifat *unreliable, connectionless* dan *datagram delivery service*. *Unreliable* berarti tidak ada jaminan bahwa paket akan sampai di tujuan. *Connectionless* artinya hubungan yang dilakukan tanpa melakukan perjanjian atau *handshake*. *Datagram delivery service* berarti pengiriman paket tidak dipengaruhi oleh paket yang lain. [3]

IPv4 merupakan pengalamatan IP yang masih banyak digunakan secara umum oleh banyak pelanggan diseluruh dunia. Namun dengan perkembangan kebutuhan terhadap layanan internet, semakin hari ketersediaannya juga semakin berkurang. *Header* IPv4 seperti ditunjukkan pada Gambar 1.

[1] Fakultas Teknik Universitas Telkom Bandung Jawa Barat



■ **Gambar** 1. *Header* IPv4

Header IP terdiri atas beberapa *field* sebagai berikut:

a. Version
Versi dari *header* IP yang digunakan

b. Header Length
Panjang *header* IP dalam format 32 bits.

c. Type of Service
Type dari servis yang menjelaskan bagaimana data harus ditangani.

d. Total Length
Merupakan panjang total dari *datagram* IP, mencakup *header* dan muatannya.

e. Identification
Nilai unik dari paket IP.

f. Flags
Menspesifikasikan apakah fragmentasi harus ada.

g. Fragment Offset
Menyediakan fragmentasi dan perakitan kembali jika paket terlalu besar untuk disimpan dalam sebuah frame.

h. Time to Live
Field ini menghentikan paket IP yang berputar di jaringan untuk mencari tujuannya.

i. Protocol
Port dari protokol upper-layer.

j. Header Checksum
CRC (cyclic redundance check) yang hanya berada pada header.

k. Source IP Address
Alamat IP 32-bits dari *host* pengirim.

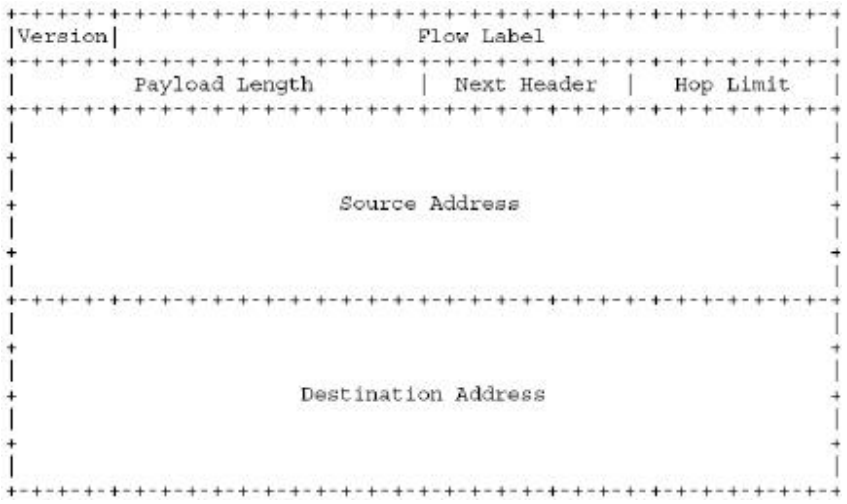
l. Destination IP Address
Alamat IP 32-bits dari *host* tujuan pengiriman paket.

m. Options
Digunakan untuk network testing, debug, keamanan, dan yang lainnya.

IPv6 memiliki beberapa fitur yang mampu mengantisipasi perkembangan aplikasi masa depan dan mengatasi kekurangan yang dimiliki pendahulunya, yaitu IPv4. IPv6 dirancang sebagai perbaikan dari IPv4. adapun format *header* dari IPv6 dapat dilihat pada Gambar 2.

Field-field pada *header* IPv6 dapat dijelaskan sebagai berikut :

1. *Version*
   Field 4 *bit* yang menunjukkan versi *Internet* Protokol, yaitu 6.
2. *Priority*
   Field 4 *bit* yang menunjukkan nilai prioritas. *Field* ini memungkinkan pengirim paket mengidentifikasi prioritas yang diinginkan untuk paket yang dikirimkan, relatif terhadap paket-paket lain dari pengirim yang sama.
3. *Flow Label*
   Field 24 *bit* yang digunakan oleh pengirim untuk memberi label pada paket-paket yang membutuhkan penanganan khusus dari router IPv6, seperti *quality of service* yang bukan default, misalnya service-service yang bersifat *real-time.*



■ **Gambar** 2. *Header IPv6 [5]*

4. *Payload Length*
   Field berisi 16 *bit* yang menunjukkan panjang *payload*, yaitu sisa paket yang mengikuti *header* IPng, dalam oktet.
5. *Next Header*
   Field 8 *bit* yang berfungsi mengidentifikasi *header* berikut yang mengikuti *header* IPv6 utama.
6. *Hop Limit*
   Field berisi 8 *bit unsigned integer*. Menunjukkan jumlah *link* maksimum yang akan dilewati paket sebelum dibuang. Paket akan dibuang bila *Hop Limit* berharga nol.
7. *Source Address*
   Field 128 *bit*, menunjukkan alamat pengirim paket.
8. *Destination Address*
   Field 128 *bit*, menunjukkan alamat penerima paket.

TUNNELING 6TO4[6]

Tunneling memungkinkan jaringan IPv6 yang terpisah untuk berkomunikasi melalui jaringan IPv4. Salah satu metode tunneling adalah 6to4. RFC 6732 mendefinisikan 6to4 tunneling, bersama dengan operasi anycast dijelaskan , secara luas digunakan dalam Sistem Operasi modern.

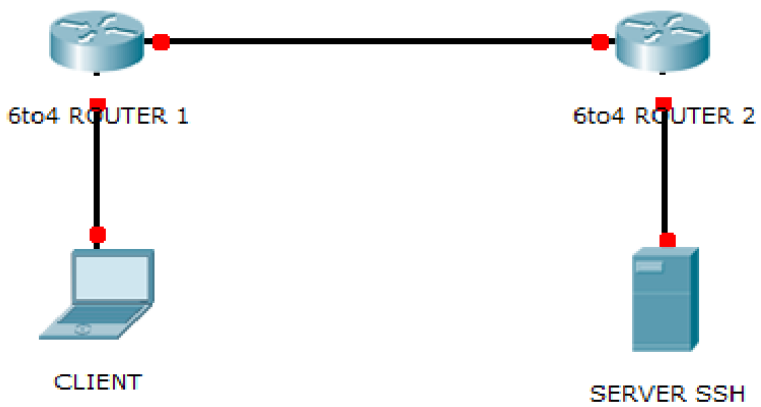
Jenis *tunneling 6to4* ini dapat digunakan pada individual *host* ataupun *local host* IPv6. Alokasi alamat blok IPv6 pada *tunneling* 6to4 diawali dengan alamat 2002(hex) dan diikuti oleh alamat IPv4 suatu *host* yang sudah diubah menjadi bilangan hexadesimal[4]

Ada 2 karakteristik dari mekanisme 6to4 yang menjadi pertimbangan dalam kemanan jaringan, yaitu:

1. Semua router 6to4 harus menerima dan mendekapsulasi paket IPv4 dari setiap router 6to4, dan dari 6to4 *relay* router.
2. 6to4 *relay* router harus menerima trafik dari node IPv6.

Dari karakteristik yang kedua, diperlukan kepercayaan antar node, dan tidak ada kendala pada isi paket IPv6. Jadi, ada kemungkinan alamat IPv4 dan IPv6 dipalsukan, dan ini mungkin menjadi awal dari berbagai ancaman seperti ancaman berbagai jenis *Denial of Service.*

Bentuk yang paling umum digunakan dalam operasi mekanisme 6to4 adalah model 6to4 to 6to4 yang dapat dilihat pada Gambar 3 berikut :



■ **Gambar** 3. *model 6to4 to 6to4*

Domain 6to4 selalu melakukan pertukaran trafik 6to4 secara langsung melalui *tunneling* 6to4, *endpoint* alamat V4ADDR berasal dari prefix 6to4 yaitu 2002:V4ADDR::/48. Ini diperlukan oleh setiap router 6to4 untuk memperhitungkan setiap router 6to4 lain yang ingin berkomunikasi.

Ancaman keamanan pada 6to4 secara umum ada tiga tipe yaitu :

1. *Denial of Service (DoS)*, merupakan node berbahaya yang dapat melumpuhkan komunikasi antar node yang sedang diserang.
2. *Reflection DoS*, merupakan node berbahaya yang mencerminkan lalu lintas tidak berbahaya ke suatu node tertentu yang dapat mencegah komunikasi antar node yang sedang diserang.
3. *Service theft*, merupakan node/operator yang mungkin membuat node/operator yang tidak sah untuk menggunakan layanan yang disediakan dalam jaringan

ALGORITMA RSA (RIVEST-SHAMIR-ADLEMAN) [1]

Model sistem penyandian RSA memiliki 2 buah kunci yang berbeda, yaitu enkripsi dan dekripsi. Enkripsi untuk menyandikan sebuah pesan (plaintext) adalah bilangan bulat utuh (*integer*) disimbolkan e dan kunci dekripsi menerjemahkan pesan tersandi (*chipertext*) adalah bilangan bulat utuh yang disimbolkan dengan d. Dari prinsip kerja RSA dapat dilihat bahwa keamanan sistem penyandian RSA bergantungpada kunci-kunci yang digunakan untuk enkripsi dan dekripsi, ukuran kunci yang digunakan pada sistem ini menentukan jumlah kombinasi kunci yang mungkin.

A. Sistem Pembangkitan Kunci

Sistem ini adalah fasilitas yang disediakan untuk membangkitkan kunci publik dan private yang diperlukan untuk melakukan proses enkripsi dan proses dekripsi. Secara keseluruhan sistem pembangkitan kunci dirancang sesuai dengan urutan dan tahapan proses yang diperlukan untuk membangkitkan kunci *public* dan *private* berdasarkan metode RSA.

Algoritmanya:

1. Pilih bilangan prima p dan q
2. Hitung n = p * q
3. Hitung $\Phi(n) = (p-1) * (q-1)$
4. Pilih sembarang bilangan b, $1< b < \Phi(n)$, dengan gcd $(b, \Phi(n)) = 1$
5. Hitung invers dari b yaitu $a = b-1 \bmod \Phi(n)$
6. Kunci publik = (n,b) dan kunci private (a).

B. Tahap Enkripsi RSA (pihak pengirim)

Algoritmanya :

1. Ambil kunci publik (n,b)
2. Pilih plainteks m, dengan $0 \leq m \leq n-1$
3. Hitung $c = m^b \bmod n$
4. Diperoleh chiperteks c, dan dikirim ke b

C. Tahap Dekripsi RSA (pihak penerima)

Algoritmanya :

1. Ambil kunci publik (a)
2. Hitung $m = c^a \bmod n$

JENIS ANCAMAN KEAMANAN JARINGAN [2]

A. IP spoofing

Spoofing adalah seni untuk menjelma menjadi sesuatu yang lain.

B. Man in the Middle Attack

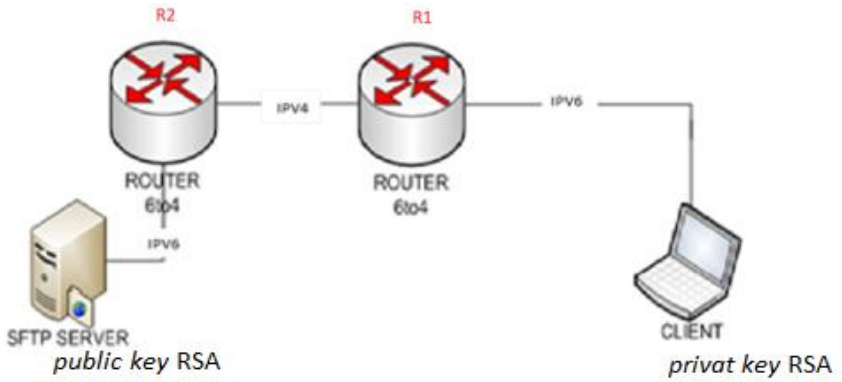
Serangan keamanan jaringan Man-in-the-middle (serangan pembajakan) terjadi saat user perusak dapat memposisikan diantara dua titik link komunikasi.

C. Sniffing

Suatu serangan keamanan jaringan dalam bentuk *Sniffer* (atau dikenal sebagai snooping attack) merupakan kegiatan user perusak yang ingin mendapatkan informasi tentang jaringan atau traffic lewat jaringan tersebut.

MODEL SISTEM

Secara garis besar, sistem yang dibuat dapat dilihat pada pemodelan sistem pada Gambar 4.



■ **Gambar** 4. topologi jaringan yang akan dibangun

Keterangan :

i. SFTP *server* digunakan untuk mejalankan tugas-tugas file (*upload* dan *download*), memperbolehkan akses ke direktori dan dapat menyelesaikan beberapa operasi seperti pindah lokasi ke tempat yang berbeda. File-file yang ada di sini dapat diakses oleh user. Software yang digunakan adalah OpenSSH.

ii. Router berfungsi untuk menyediakan konektifitas IPv6 dan IPv4, dan mengirim paket dari satu router 6to4 ke router 6to4 yang lain.

iii. *Client* berfungsi untuk menyiapkan data yang dimasukkan oleh pengguna dengan menggunakan teknologi pemrosesan tertentu dan mengirimkannya kepada komponen *server*. Software yang digunakan adalah WinSCP.

iv. Algoritma RSA (*public key* dan *private key*) berfungsi untuk mengenkripsi dan mendekripsi data yang dikirimkan oleh *client* ke *server* atau sebaliknya.

Cara Kerja Sistem

Cara kerja system secara keseluruhan :

i. *Tunneling* mode 6to4 akan menghubungkan IPv4 pada jaringan *client* dengan IPv6 pada jaringan *server*.

ii. Saat *client* mengirimkan data ke *server*, maka data tersebut akan di-enkripsi terlebih dahulu menggunakan algoritma RSA. Setelah itu baru dilewatkan pada jaringan.

iii. Pada sisi *server*, *server* sudah siap untuk mendekripsikan data yang dikirimkan oleh *client* dengan kunci public yang dimiliki oleh *client*. Pada saat pengiriman dari *server* ke *client*, *client* menggunakan kunci privat untuk mendekripsi data dari *server*.

PENGUJIAN

A. Melewatkan paket dengan ukuran 10 MB

■ **Tabel** 1. Perbedaan data yang dikirimkan SSH dan FTP

NO	JARINGAN	DATA RATA-RATA YANG DITERIMA (bytes)	UKURAN DATA SEBENARNYA (bytes)	Δ (bytes)
1	SSH	11140379,67	10485760	654619,6666
2	FTP	10966690,9	10485760	480930,9

■ **Tabel** 2. Perbedaan penambahan *header*

NO	JARINGAN	*Header* wireshark (bytes)	Jumlah Paket rata-rata	Penambahan *Header* oleh wireshark (bytes) (jumlah paket x *header*)
1	SSH	54	8023,266667	433265,4
2	FTP	54	7437,6	401630.4

- Pada jaringan SSH :

Penambahan bytes karena SSH = Δ (bytes) - Penambahan *Header* oleh wireshark (bytes)
= 654619,6666 - 433265,4
= 221354,2666 bytes = 221,354 kbytes

- Pada jaringan FTP :

Penambahan bytes karena FTP = Δ (bytes) - Penambahan *Header* oleh wireshark (bytes)
= 480930,9- 401630.4
= 79300.5 bytes = 79,3 kbytes

Jaringan SSH memiliki penambahan bytes lebih besar dari jaringan FTP, hal ini diakibatkan karena adanya enkripsi pada setiap paket pada transfer SSH.

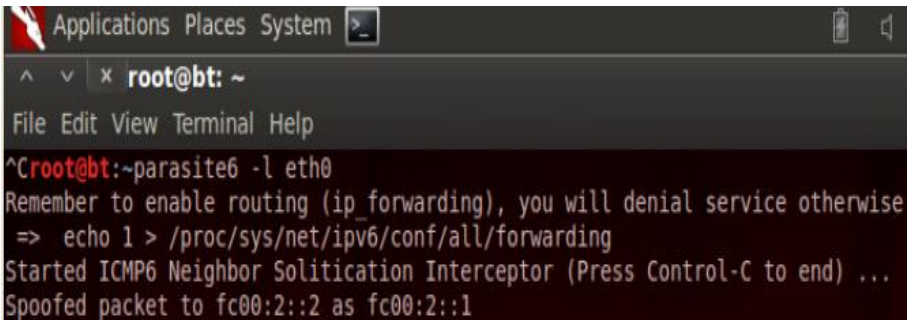
B. Perbandingan *header* paket pada jaringan SSH dan FTP

Pada saat transfer menggunakan FTP, *client* mengirimkan paket SYN untuk mengawali *three way handshake*. Setalah melakukan *three way handshake* barulah paket dikirimkan. Hal ini akan dilakukan sistem setiap akan melakukan transfer file. Dan setelah selesai mengirimkan paket, maka *client* akan mengirimkan ACK dan FIN untuk mengakhiri hubungan. Inilah perbedaannya pada saat melakukan transfer dengan menggunakan SSH.

Pada saat melakukan transfer file di jaringan SSH, *client* tidak melakukan *three way handshake*. *Three way handshake* hanya dilakukan pada saat *login* pertama kali ke *server*. Dan pemutusan hubungan dengan mengirimkan flags FIN dikirimkan saat melakukan proses *dissconeccting*. Hal ini terjadi karena SSH menggunakan *tunnel* sehingga selama tidak ada pemutusan hubungan oleh *client* atau *server* maka akan terus terjadi komunikasi antara *client* dan *server*. Sebaliknya pada saat transfer file FTP, setelah FIN diterima oleh *client* atau *server* maka tidak ada lagi komunikasi antara *client* dan *server*.

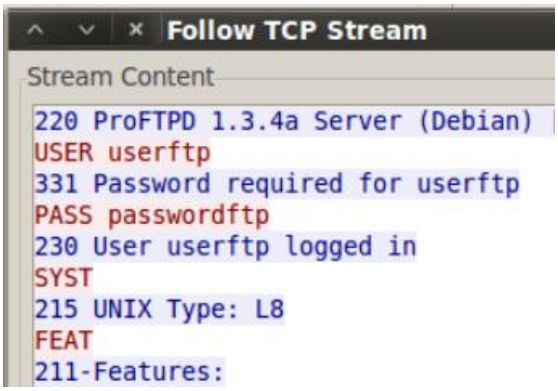
C. Analisa pengaruh *Sniffing* terhadap keamanan *password* dan isi data.

Sebelum melakukan *sniffing*, *attacker* terlebih dahulu melakukan *spoofing*. *Spoofing* agar *attacker* dapat melakukan *Man in The Middle Attack*, sehingga nantinya seluruh komunikasi antara *client* dan *server* akan melalui *attacker*. Gambar 5 menunjukkan *spoofing* oleh *attacker*.



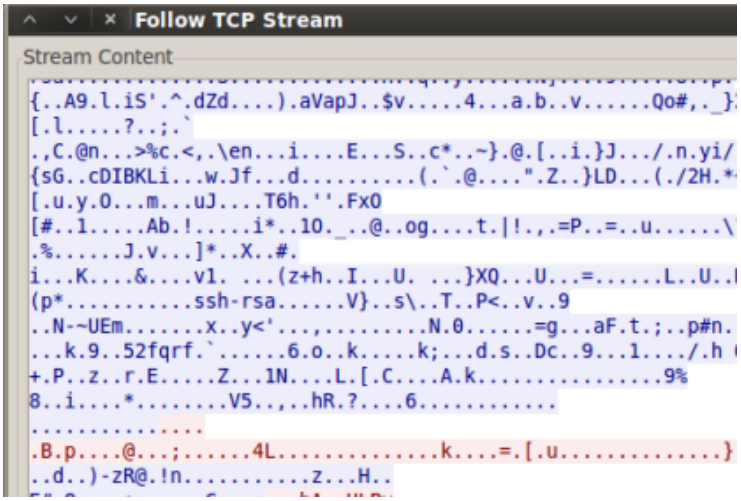
■ **Gambar** 5. *spoofing* oleh *attacker*

Dari hasil *sniffing* yang dilakukan dengan menggunakan wireshark pada saat *client* melakukan *login*. Saat *client* melakukan *login* ke *server* FTP, setelah sistem melakukan *three way handshake* terlihat paket yang menunjukkan permintaan *server* untuk memasukan *username* dan *password*. Kedua paket ini dapat dibaca isinya oleh wireshark. Hasil *capture* proses *sniffing* ini dapat dilihat pada Gambar 6 berikut:



■ **Gambar** 6. hasil *sniffing* pada FTP

Dengan adanya *sniffing* ini maka pihak lain (*attacker*) yang memperoleh *username* dan *password* akan bebas menggunakan layanan yang dimiliki oleh *client* dan bebas melakukan exploitasi terhadap sumber daya yang dimiliki oleh *client* yang disimpan pada *server* FTP.

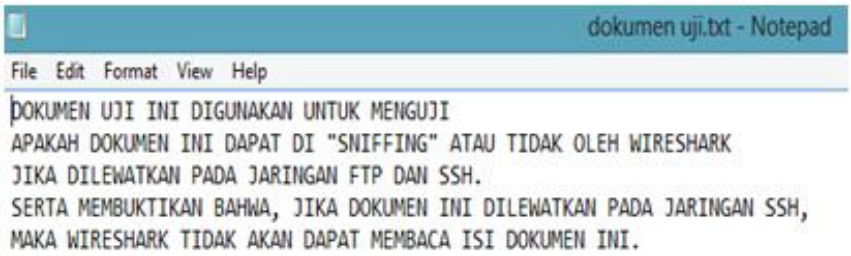


■ **Gambar** 7. hasil *sniffing* pada SSH

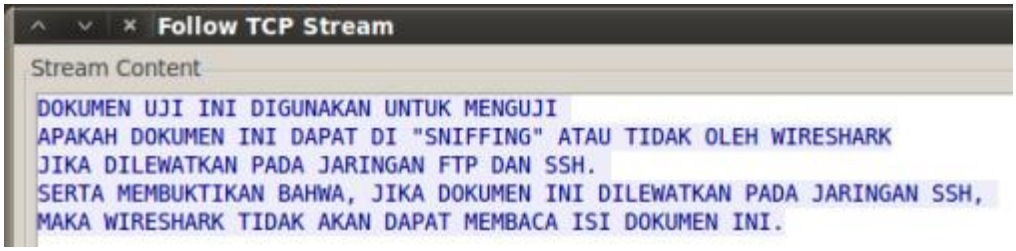
Dengan adanya enkripsi saat *login* maka *username* dan *password* dari *client* tidak dapat dibaca oleh *attacker* saat melakukan *sniffing*. Gambar 7 adalah hasil *sniffing* pada SSH.

Pengujian keamanan data

Pengujian berikutnya dilakukan untuk menguji kerahasiaan data yang dikirimkan oleh *client* ke *server*. Pengujian ini dilakukan dengan mengirimkan sebuah file .txt. Hasil data uji hasil *sniffing* data pada FTP dan hasil *sniffing* data pada SSH dapat dilihat pada Gambar 8, Gambar 9 dan Gambar 10.



■ **Gambar** 8. data uji



■ Gambar 9. hasil *sniffing* data pada FTP

Jika dilakukan Follow TCP stream pada paket nomor 176 maka data yang dikirimkan oleh *client* dapat dibaca oleh *attacker* seperti yang terlihat pada gambar 4.20.

Pada saat dilakukan *sniffing* pada jaringan SSH, pengiriman paket tetap dapat diketahui oleh *attacker* namun *attacker* tidak dapat mengetahui isi dari paket yang dikirimkan oleh *client*.

D. Pengujian utilitas *server* SSH

Saat dilakukan pengiriman data 200 MB, FTP memerlukan waktu 18 detik dengan rata-rata utilitas CPU adalah sebesar 1,333333 % sedangkan utilitas rata-rata untuk SSH adalah sebesar 15,19355 %. Perbedaan ini diakibatkan oleh adanya enkripsi yang dilakukan dengan menggunakan algoritma RSA sehingga mengakibatkan beban CPU pada *server* menjadi lebih besar. Pada gambar 4.23 dapat dilihat grafik perbedaan utilitas CPU. Pada gambar tersebut terlihat beban CPU selalu lebih besar ketika terjadi transfer pada SSH.

■ **Gambar** 10. hasil *sniffing* data pada SSH

E. Analisa kekuatan kunci algoritma RSA

Algoritma RSA adalah sebuah algoritma yang memiliki kunci yang cukup panjang. Sekarang, ukuran kunci 1024 bit sudah cukup kuat untuk semua aplikasi. Ada dua kemungkinan cara untuk memecahkan kunci algoritma RSA. Pertama adalah dengan melakukan *brute force attack* (mencoba semua kemungkinan *private key*).

- Panjang kunci = 1024 bit
- Kemungkinan kunci : $2^{1024} = 1{,}79 \times 10^{308}$
- Dengan asumsi waktu dekripsi 1 µs untuk satu kunci, maka :
  Waktu yang dibutuhkan = $1{,}79 \cdot 10^{308} \times 1\mu s$
  $= 1{,}79 \cdot 10^{308} \times 10^{-6}$ detik = **$5{,}7 \cdot 10^{294}$ tahun**

Kedua adalah dengan metode *cryptanalysis*. *Cryptanalysis* pada RSA difokuskan pada bagaimana memfaktorkan nilai *n* menjadi dua bilangan prima**Error! Reference source not found.**. Jenis serangan ini bisa dilakukan tanpa mengtahui *ciphertext*. Walaupun demikian, belum ada algoritma pemfaktoran yang mampu melakukan pemfaktoran bilangan *integer* secara cepat. Sehingga memerlukan waktu yang lama untuk melakukan serangan ini.

Metode matematis seperti serangan GCD (*greatest common modulus*) dan *common modulus* dapat memecahkan RSA. Untuk metode GCD, penyerang harus mendapatkan dua buah *ciphertext* dari dua buah pesan (m_1 dan m_2, dengan syarat $m_2 = m_1 + \Delta$). Dan untuk serangan *common modulus* dapat dilakukan apabila diketahui dua buah *ciphertext* hasil enkripsi pesan m dengan dua buah kunci enkripsi yang berbeda, selain itu nilai n dari yang digunakan untuk kedua enkripsi tersebut harus sama. Metode GCD dan *common modulus* secara matematis dapat memecahkan RSA, namun dengan kondisi tertentu, terbatas pada pesan-pesan yang memenuhi kondisi bersangkutan, dan memerlukan waktu yang cukup lama.

KESIMPULAN

1. Data yang dikirimkan dari *server* SSH memiliki *delay* yang lebih besar dibandingkan ketika data diterima dari *server* FTP. Dimana, *delay* untuk pengiriman SSH adalah 64,1172 ms sedangkan pada saat pengiriman dengan *server* FTP adalah 39,6879 ms. Hal ini terjadi karena jumlah total bytes yang dikirimkan dari *server* SSH lebih besar dibandingkan dengan *server* FTP, serta *delay* akibat penggunaan *tunnelling* SSH dan enkripsi dengan menggunakan algoritma RSA.
2. Jalur komunikasi antara *client* dan *server* SSH akan terus diduduki sampai salah satu dari *client* atau *server* memutuskan hubungan. sedangkan pada saat menggunakan FTP, setiap *server* atau *client* selesai mengirimkan data maka akan terjadi penutupan hubungan, atau dengan kata lain jalur sudah tidak digunakan lagi.
3. Ketika terjadi serangan *man in the middle attack*, seluruh data yang dikirimkan oleh *client* ke *server* maupun sebaliknya akan melewati PC/laptop *attacker* sehingga semua proses komunikasi dapat diketahui oleh *attacker*.
4. Dengan adanya enkripsi dengan RSA pada komunikasi SSH, *attacker* hanya dapat mengetahui adanya komunikasi antara *client* dan *server* namun tidak dapat mengetahui isi dari komunikasi tersebut.

5. Pada saat komunikasi menggunakan FTP. *Attacker* dapat mengetahui *username, password,* dan isi data yang dikirimkan.
6. Dengan adanya enkripsi dengan algoritma RSA utilitas CPU menjadi lebih besar yaitu rata-rata 15,19355 % pada SSH *server*, sedangkan 1,333333% pada FTP *server* yang tidak menggunakan enkripsi.

DAFTAR PUSTAKA

[1]. Erika, Winda dkk.2012. *Implementasi Algoritma RSA untuk Perlindungan Data pada Server FTP*. Bandung : Politeknik Telkom.
[2]. Hariono , Ir. H. Ali.2009. http://www.sysneta.com/ancaman-keamanan-jaringan (diakses tanggal 22 November 2012)
[3]. Lammie, Todd.2005.*CCNA Cisco Certified Network Associate Study Guide*. Jakarta : PT. Elex Media Komputindo
[4]. Paramayudha, Gilang R.2010.*Analisa Perbandingan Performansi Jaringan IPv4, IPv6 dan Tunneling 6to4 untuk Aplikasi File Transfer Protokol (FTP) pada Media wired dan wireless di sisi Client*. Depok : Universitas Indonesia.
[5]. Riyadi, Roni.2008. *Analisis Proaktif dan Reaktif Handover pada Jaringan Mobile IPv6.* Bandung : IT Telkom.
[6]. Savola, P., Patel, C. 2004. *Security Consideration for 6to4*. IETF.
[7]. http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2005-2006/ Makalah/Makalah2005-11.pdf (diakses tanggal 2 oktober 2013)