Aplikasi Tanda Tangan Digital sebagai Penjamin Keaslian Konten pada Layanan *Streaming*

Muhammad Naufal Fakhrizal - 13518115
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: mnaufalfakhrizal@gmail.com

***Abstract*—Layanan *streaming* memungkinkan pengiriman media kontinu secara *real-time*. Untuk mengaplikasikan layanan-layanan kriptografi seperti otentikasi, menjamin integritas media, dan anti-penyangkalan, sebuah sistem tanda tangan digital dapat diaplikasikan pada layanan *streaming*. Sistem tanda tangan menggunakan algoritma kriptografi kunci publik RSA dan fungsi *hash* SHA-256. Karena layanan streaming mengirimkan media kepada sebuah *client* dalam bentuk potongan atau frame yang langsung ditayangkan ketika *client* menerima frame tersebut, untuk menjamin ketiga layanan kriptografi pada seluruh bagian media, tanda tangan diaplikasikan pada setiap *frame* dari media yang dikirim.**

Kata kunci—anti-penyangkalan, audio, hash, integritas, kriptografi, media, otentikasi, RSA, streaming, SHA256, tanda tangan digital

I. PENDAHULUAN

Layanan *streaming* yang populer di masa sekarang memberikan kenyamanan bagi pengguna untuk dapat menikmati media-media yang tersedia secara *online* secara langsung tanpa harus mengunduh keseluruhan file media. Tetapi, terdapat potensi kerentanan terhadap keamanan dalam *straming* media tersebut, seperti pengubahan konten media oleh pihak lain, atau pihak lain mencoba berpura-pura menjadi layanan *streaming* yang asli. Masalah-masalah ini dapat diselesaikan dengan kriptografi

Salah satu teknologi kriptografi adalah tanda tangan digital. Penggunaan tanda tangan digital pada sebuah pesan menjamin integritas dari pesan tersebut dan menjamin bahwa pihak yang mengirim pesan adalah pihak yang sebenarnya. Pada umumnya tanda tangan digital diberikan pada sebuah pesan utuh seperti file atau dokumen.

Pada sebuah layanan *streaming*, pesan dalam bentuk konten video atau audio dibagi menjadi beberapa *frame* atau potongan yang dikrim dari *server* ke client secara sekuensial. Hal ini dilakukan agar *client* dapat menayangkan video atau audio secara *real time* tanpa perlu menerima keseluruhan konten terlebih dahulu. Untuk menjamin layanan-layanan kriptografi pada konten pada layanan *streaming* menggunakan sistem tanda tangan digital, maka tanda tangan perlu diberikan pada setiap frame dari konten. Hal ini memberikan tantangan untuk mengimplementasi sebuah sistem tanda tangan yang dapat melakukan verifikasi tanda tangan pada tiap *frame* secara *real time*.

II. DASAR TEORI

A. Tanda Tangan Digital

Tanda tangan digital adalah sebuah nilai yang digunakan sebagai bukti digital terhadap keaslian suatu pesan yang dijamin oleh pemberi tanda tangan tersebut. Berbeda dengan tanda tangan fisik yang pada umumnya merupakan tulisan tangan, tanda tangan digital adalah sebuah bilangan besar yang dihasilkan menggunakan algoritma-algoritma kriptografi. Hal ini berguna untuk menyelesaikan perselisihan mengenai keaslian sebuah dokumen digital yang dapat dibuktikan kepada pihak penengah menggunakan tanda tangan digital.

Salah satu skema tanda tangan digital adalah dengan menggunakan kriptografi kunci publik. Pada skema ini, pengirim dokumen memiliki kunci privat yang hanya diketahui sendiri dan kunci publik yang disebarkan kepada umum. Untuk membuat tanda tangan digital dari sebuah dokumen, pengirim dokumen melakukan enkripsi terhadap dokumen menggunakan kunci privatnya. Kemudian, pengirim akan mengirim dokumen beserta tanda tangan digital yang telah dibuat kepada penerima.

Untuk melakukan verifikasi dari dokumen yang memiliki tanda tangan, penerima membutuhkan kunci publik dari pengirim dokumen untuk melakukan dekripsi dari tanda tangan, kemudian membandingkan hasilnya dengan dokumen. Apabila hasil perbandingan sama, maka tanda tangan dan dokumen tersebut asli dan benar dikirim oleh pihak pengirim.

Selain itu, sebuah fungsi *hash* dapat digunakan untuk mendapatkan nilai *hash* dari sebuah dokumen, yang kemudian dienkripsi menggunakan kunci *privat* pengirim untuk mendapatkan tanda tangan digital. Untuk melakukan verifikasi, penerima dapat menghitung nilai *hash* dari pesan yang diterima, kemudian membandingkannya dengan nilai *hash* yang diperoleh dari mendekripsi tanda tangan digital menggunakan kunci publik pengirim. Verifikasi berhasil ketika

kedua nilai *hash* yang diperoleh dengan cara berbeda bernilai sama.

Sebagai sebuah sistem kriprotgrafi, Tanda tangan digital menyediakan layana sebagai berikut:

1) Otentikasi (Authentication)

Penggunaan tanda tangan pada sebuah dokumen merupakan sebuah bukti otentik bahwa seseorang atau pihak telah membuat, membaca, atau menyetujui isi dari dokumen tersebut. Setiap pihak yang terlibat memiliki kunci publik yang diketahui bersama dan kunci privat yang hanya diketahui masing-masing. Jika verifikasi menggunakan suatu kunci publik dari seseorang berhasil, maka bisa dipastikan bahwa orang tersebut telah memberikan tanda tangan menggunakan kunci privat miliknya.

2) Integritas pesan (Integrity)

Penggunaan tanda tangan dengan hash dan kriptogradi kunci publik dapat digunakan untuk menjamin integritas dari dokumen, karena tanda tangan yang dihasilkan bergantung pada kunci privat dan isi dari dokuman. Jika dokumen yang diberi tanda tangan berubah, maka nilai *hash* dari dokumen akan berubah dan sehingga verifikasi tanda tangan digital akan menjadi gagal. Dengan begitu, dapat diketahui apakah dokumen yang diterima masih utuh atau tidak dengan memverifikasi tanda tangan digital dari dokumen tersebut.

3) Anti-penyangkalan (Non-Repudiation)

Penggunaan tanda tangan digital pada sebuah dokumen selain memberikan otentikasi terhadap orang yang memberikan tanda tangan tersebut, tetapi juga mencegah orang tersebut menyangkal bahwa dia telah memberikan tanda tangan pada dolumen tersebut. Hal ini dimungkinkan karena hanya orang tersebut yang mengetahui kunci privat miliknya pada saat memberikan tanda tangan, sehingga ketika dilakukan verifikasi tanda tangan menggunakan kunci publik dari orang tersebut, tanda tangan yang menggunakan kunci privat orang tersebut akan selalu terverifikasi dan dia tidak bisa menyangkal bahwa tanda tangan tersebut bukan miliknya.

B. Algoritma Enkripsi RSA

Salah satu algoritma yang umum digunakan dalam implementasi tanda tangan digital adalah algoritma enkripsi RSA. RSA merupakan algoritma enkripsi kunci publik yang dikembangkan oleh Ronald Rivest, Adi Shamir, dan Len Adleman.

Proses pembangkitan kunci RSA adalah sebagai berikut:

- Menentukan dua buah bilangan prima *p* dan *q* secara acak. Perkalian dua bilangan tersebut menghasilkan nilai *n* ($n = p \times q$) nilai *p* dan *q* adalah rahasia, sementara nilai *n* tidak rahasia. Keamanan dari algoritma RSA bersumber dari kesulitan memfaktorkan bilangan prima *p* dan *q* dari bilangan bulat besar *n.*
- Menentukan nilai *toitent* Euler dari *n*. Pada kasus ini, nilai *toitent n* adalah $\Phi(n) = (p-1)(q-1)$. Nilai ini bersifat rahasia.
- Menentukan nilai kunci enkripsi *e*, sehingga faktor persekutuan terbesar dari *e* dan *toitent n* adalah 1. Dengan kata lain, *e* dan Φ(n) adalah relatif prima. Nilai *e* tidak rahasia dan akan digunakan sebagai kunci publik.
- Menentukan nilai kunci dekripsi *d*, yang merupakan hasil modulo invers dari nilai *e* dalam modulus Φ(n) ($d = e^{-1} mod(\Phi(n))$). Nilai *d* adalah rahasia dan akan digunakan sebagai kunci privat.

Dari proses pembangkitan dihasilkan dua pasang kunci, yaitu kunci publik (*e, n*) untuk melakukan enkripsi dan kunci privat (*d, n*) untuk melakukan dekripsi. Kunci publik dapat disebarkan dan diketahui oleh pihak lain, sementara kunci privat hanya dimiliki oleh pembuat kunci.

Pada proses enkripsi pesan, plainteks direpresentasikan sebagai sebuah bilangan *m.* Cipherteks *c* dapat dihasilkan dengan menghitung

$$c = m^e \bmod n$$

Untuk melakukan dekripsi pesan,

$$e = c^d \bmod n$$

C. Algoritma Hash SHA-256

Algoritma SHA-256 termasuk ke dalam keluarga algoritma SHA-2 dan merupakan algoritma untuk menghitung nilai *hash* dari sebuah pesan. Algoritma *hash* merupakan algoritma satu arah mengambil informasi dari pesan yang memiliki panjang yang bervariasi untuk menghasilkan nilai *hash* atau *digest* yang berukuran tetap. Algoritma *hash* dapat dipasangkan dengan algoritma enkripsi kunci publik untuk membangun sebuah sistem tanda tangan.

Langkah-langkah algoritma SHA-256 adalah sebagai berikut:

Pertama, dilakukan penambahan *padding* di akhir pesan. *Padding* berisikan bit bernilai 1, kemudian diikuti oleh bit-bit bernilai 0, kemudian diikuti oleh nilai 64-bit yang menyimpan panjang pesan. Jumlah bit 0 disesuaikan sehingga panjang pesan ditambah *padding* merupakan kelipatan dari 512 bit. Kemudian, pesan dengan *padding dibagi* menjadi blok-blok berukuran 512 bit.

Delapan nilai *hash* antara dengan ukuran 32-bit H_0 sampai H_7 diinisialisasi dengan nilai yang telah ditentukan. Dan sebuah *array* konstan *K* yang menyimpan 64 bilangan 32-bit konstan

Pada setiap blok 512-bit dari pesan, dilakukan hal-hal berikut:

- Melakukan inisialisasi terhadap delapan register *a, b, c, d, e, f,* dan *g* dengan nilai H_0 sampai H_7
- Menyiapkan sebuah array W yang menyimpan 64 bilangan 32-bit
- Membagi blok menjadi 16 bilangan 32-bit dan menyimpannya di 16 elemen pertama array W.
- Melakukan '*message schedule*' dengan menghitung nilai 48 elemen berikutnya dari array W berdasarkan 16 elemen pertama.

- Melakukan kompresi sebanyak 64 putaran sebagai Menggunakan kedelapan register, array *K*, dan array *W*.
- Menjumlahkan nilai *a, b, c, d, e, f, g,* dan *h* kepada variabel H_0 sampai H_7

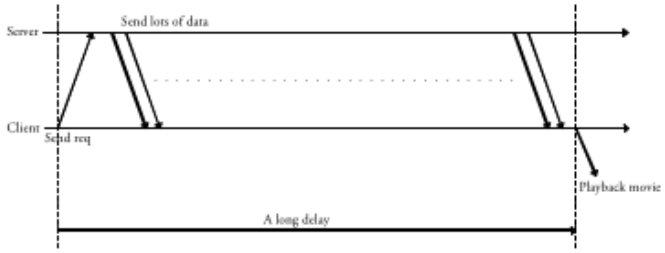
Setelah semua blok diproses, nilai H_0 sampai H_7 digabung untuk medapatkan nilai *hash* atau *digest* akhir berukuran 256-bit.

D. Streaming media

Streaming media adalah sebuah teknik untuk melakukan pengiriman media kontinu dari satu pihak ke pihak lain secara berkelanjutan dan terjadwal. Media kontinu mengacu pada jenis media digital yang memiliki aspek temporal (waktu) sehingga membutuhkan pengaturan waktu pada saat menayangkan media. Video merupakan salah satu contoh media kontinu, karena pada dasarnya video merupakan sekuens dari *frame* gambar yang ditampilkan secara sekuensial dengan frekuensi yang ditentukan, dalam satuan *frames per second*.

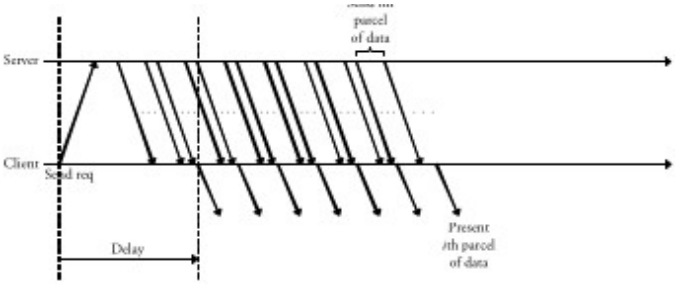
Untuk melaukan pengiriman media kontinu dari satu pihak ke pihak lain, diperlukan penjadwalan dalam pengiriman objek data yang menjadi bagian media, sehingga pihak penerima menerima setiap data bagian dari media pada waktu yang tepat dan menayangkan media sebagaimana mestinya.

Streaming merupakan alternatif dalam penyampaian media melalui jaringan selain dari metode *download*. Pada metode *download*, *client* mengirimkan *request* kepada *server* untuk mengunduh media. *Server* kemudian mengirimkan bagian-bagian dari media kepada *client*, yang kemudian disimpan oleh *client* dalam memori atau *cache*. Setelah keseluruhan data diterima oleh *client*, kumpulan bagian data tersebut kemudian disusun untuk mendapatkan file media utuh yang bisa ditayangkan.



Gambar 1. Proses transmisi data dan penayangan media pada model *download* (Lee, 2005).

Sementara itu, pada metode *streaming*, setiap kali data bagian dari media diterima oleh *client* dari server, bagian tersebut dapat langsung ditayangkan oleh *client* kepada pengguna sambil menunggu data selanjutnya diterima. Metode *streaming* memiliki keunggulan dibanding metode *download* yaitu pengguna dapat menonton atau mendengarkan media secara langsung dan tidak perlu menunggu keseluruhan media untuk diunduh sebelum media bisa ditayangkan.



Gambar 2. Proses transmisi data dan penayangan media pada model *streaming* (Lee, 2005).

Terdapat dua jenis pengiriman media kontinu berdasarkan kebutuhan dari

1) Real-time delivery

Pada sistem *real-time delivery*, terdapat kebutuhan untuk mengirimkan data dari pengirim ke penerima dalam batas *delay* tertentu. Sistem ini ditemukan dalam aplikasi *video conference* dan *internet phone* yang harus mendukung komunikasi antar pengguna secara *real-time*. Oleh karena itu, sistem lebih memprioritaskan data sampai ke penerima dengan *delay* yang kecil, sementara integritas data tidak terlalu diutamakan, sehingga dapat terjadi penurunan kualitas video dan audio pada aplikasi-aplikasi yang menggunakan *real-time delivery* pada saat kondisi jaringan buruk.

2) Soft-real-time delivery

Pada sistem *soft-real-time* delivery, tidak ada kebutuhan untuk memastikan data terkirim dari pengirim ke penerima dalam *delay* yang singkat. Pada sistem ini, integritas dan kualitas dari data yang dikirim lebih diprioritaskan dibandingkan dengan ketepatan waktu dari pengiriman data. Sistem ini ditemukan dalam aplikasi *streaming* video, musik, dan sebagainya yang melibatkan pengiriman data media dari *server* kepada *client*. Pada sistem ini, *client* akan menerima beberapa *frame* dari media terlebih dahulu sebelum menayangkan media untuk memberikan penayangan yang halus kepada pengguna.

III. RANCANGAN SISTEM

Sistem yang akan dirancang adalah sebuah sistem untuk melakukan *streaming* dengan *soft-real-time delivery* terhadap media audio yang diberi tanda tangan digital. Sistem ini memiliki arsitektur *client-server* sederhana. Pada sistem ini terdapat satu *server* yang memiliki kunci privat dan media yang akan dikirimkan ke beberapa *client* dalam kumpulan *frame*. *Client* memiliki kunci publik dari *server* yang dapat diperoleh dari layanan sertifikasi pihak ketiga yang terpercaya.

A. Struktur Paket

Struktur paket yang digunakan dalam komunikasi antara server dan client pada sistem ini terdiri atas *header* dan *body*.

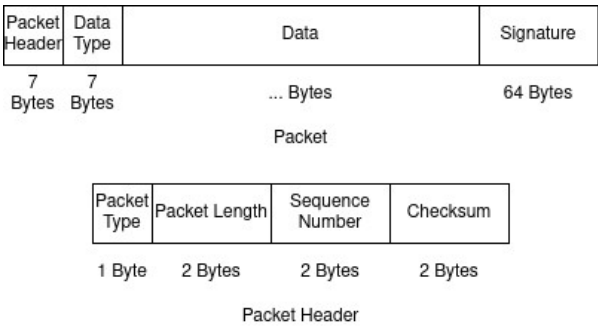
Header dari paket berukuran 7 byte, dengan detail sebagai berikut:

- *Packet type* berukuran 1 byte menyatakan jenis paket

- *Packet length* berukuran 2 byte menyatakan ukuran *body* pada paket
- *Sequence number* berukuran 2 byte menyatakan nomor urut dari paket. Hal ini berguna untuk menentukan urutan potongan-potongan data dari media yang akan disebarkan
- *Checksum* berukuran 2 byte menyatakan *checksum* dari isi paket.

Body dari paket memiliki ukuran yang ditentukan oleh *packet length*, dengan detail sebagai berikut:

- *Data type* menyatakan jenis data dari konten paket.
- *Data* menyatakan isi dari paket, seperti metadata atau *raw data* dari media yang disebarkan.
- *Signature* berukuran 64 byte atau 512-bit yang menyatakan tanda tangan digital dari data.



Gambar 3. Struktur paket

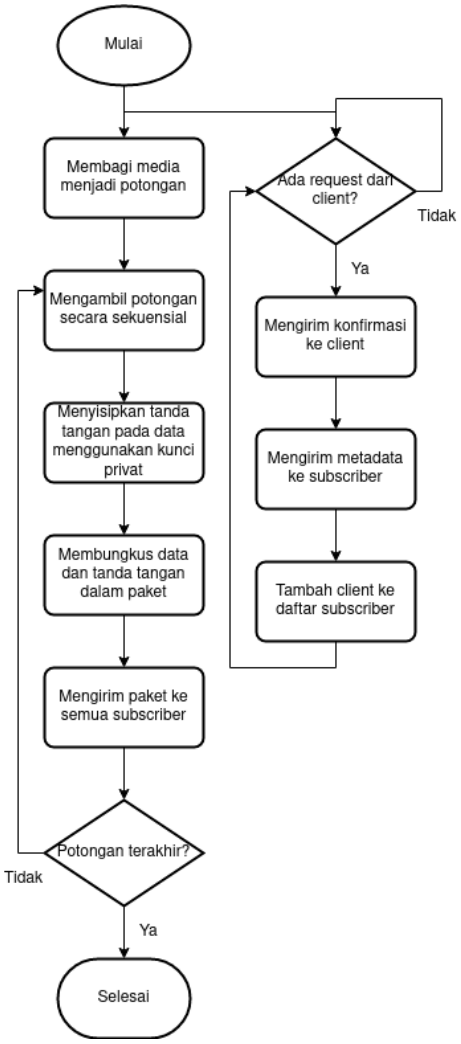
B. *Rancangan Server*

Server menjalankan dua buah *thread*. Satu *thread* menerima *request* dari *client* untuk melakukan *subscribe* terhadap layanan *streaming* dari *server* tersebut. Sementara itu, satu *thread* lain akan mengirimkan media kepada *client* yang terdaftar.

Proses penerimaan *request* dari *client* dilakukan pada sebuah *thread* yang mendengarkan paket-paket yang datang kepada *server*, dan menetukan apakah paket-paket tersebut merupakan *request* dari *client*. Jika ya, maka *server* akan mengirim konfirmasi kepada *client* dan mencatat alamat *client* dalam sebuah daftar *subsctiber*. Selain itu, *server* juga mengirim metadata dari media yang akan ditayangkan kepada *client.*

Pada *thread* utama, *server* akan membagi media menjadi *frame* atau potongan-potongan. Setiap *frame* akan diberi tanda tangan menggunakan kunci privat *server*. Kemudian, *frame* dan tanda tangan yang berkoresponden dibungkus dalam satu buah paket. *Server* kemudian mengirim paket satu per satu kepada seluruh *client* yang terdaftar secara sekuensial, dengan interval pengiriman setiap paket disesuaikan dengan panjang paket. Misalnya, jika *frame* pada sebuah paket berkorespondensi dengan konten media sepanjang 100 milidetik, maka *server* akan mengirim paket kepada *client* setiap 100 milidetik.

Flowchart dari algoritma *server* dapat dilihat pada gambar berikut:



Gambar 4. Flowchart *Server*

C. *Rancangan Client*

Client bertugas menerima dan menayangkan konten media yang dikirim dari *server*.

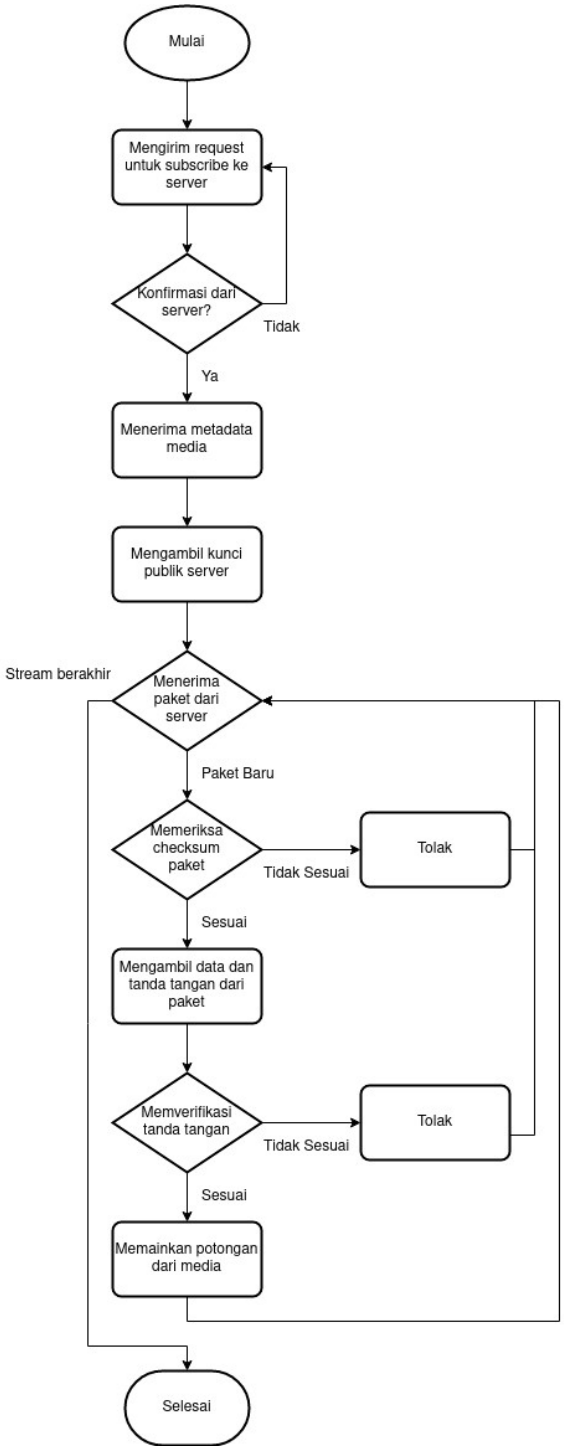
Pertama, *Client* mengirim *request* untuk melakukan *subscribe* ke *server*. *Client* akan terus melakukan *request* sampai *server* mengirimkan konfirmasi kepada *client*. Setelah itu, *client* menerima metadata dari media yang akan ditayangkan. Pada media audio, metadata dapat terdiri dari jumlah *channel*, *sampling rate*, panjang *frame*, dan sebagainya.

Setelah menerima metadata, *client* memasuki *loop* untuk menerima paket yang berisi *frame* dan tanda tangan dari media yang akan ditayangkan. Ketika menerima paket baru dari server, *client* akan memeriksa *checksum* dari paket untuk memeriksa integritas dari paket. Jika *checksum* tidak sesuai dengan isi paket, maka *client* akan menolak paket tersebut.

Setelah memeriksa *checksum* dari paket, *client* kemudian mengekstra data dan tanda tangan dari paket tersebut, dan memverifikasi tanda tangan tersebut menggunakan kunci publik dari *server*. Jika ternyata verifikasi gagal, hal ini menyatakan bahwa data telah dimodifikasi oleh pihak ketiga, atau kunci publik yang digunakan tidak sesuai dengan kunci *privat* dari server. Oleh karena itu, *client* akan menolak paket tersebut.

Jika pemeriksaan *checksum* dan tanda tangan sukses, *client* akan mengantrikan data tersebut ke dalam sebuah *media player* yang akan menayangkan media dalam data tersebut, sementara *client* menunggu dan menerima paket berikutnya.

Flowchart dari algoritma *client* dapat dilihat pada gambar berikut:



Gambar 5. Flowchart *Client*

IV. Analisis

Untuk menguji sistem, sistem dijalankan dengan beberapa skenario. Pengujian menggunakan dua pasang kunci publik dan kunci privat berukuran 512-bit.

Kunci Privat 1

(8712054966994351156497408959887266277174075574201668808504111755381043792869472402695210620645765348446641075051240719734796231850599109307512856343544 7,
2598313699809172355106364658237197481934171099038818358444655947678237934758589512141320044775023173187523076122452575899800059331159911145949625268322543)

Kunci Publik 1

(1517256152630589409498647197438751852320599359932431848272459363309506916998897285605220738219277404330305864895796098030365886428560489329082794653654743,
2598313699809172355106364658237197481934171099038818358444655947678237934758589512141320044775023173187523076122452575899800059331159911145949625268322543)

Kunci Privat 2

(6061246791297760866706400933466475312838294212145781083135938529374373735865345908363341485260367297383067131647159140841165746957612502350331283222380 97,
8138284673453572343294570720006816787231016191968236770576468783255937371094995345968217360759256623981266204835459632730858570875478266931272246464052 63)

Kunci Publik 2

(2551895372003454080404914974995033730250163566305034095421508068315403875546605895850127636652526802852303303222319409762788590593498653184970288995328 33,
8138284673453572343294570720006816787231016191968236770576468783255937371094995345968217360759256623981266204835459632730858570875478266931272246464052 63)

Pada skenario pertama, *server* dan *client* menggunakan kunci privat dan publik yang bersesuaian, yaitu kunci privat 1 dan kunci publik 1, dan *server* mengirimkan data media yang tidak dimodifikasi. Pada skenario ini, *client* berhasil memverifikasi semua tanda tangan yang disertakan pada setiap paket yang dikirimkan oleh *server*.

Dari aspek kinerja pada *client*, *client* mampu untuk membaca dan memverifikasi tanda tangan yang bersamaan dengan data media dan menayangkan bagian media tersebut

dalam kecepatan yang sama dengan penerimaan paket yang datang dari *server*. Oleh karena itu,

Pada skenario kedua, *server* dan *client* menggunakan kunci privat dan publik yang tidak bersesuaian, yaitu kunci privat 1 dan kunci publik 2. Pada skenario ini, *client* gagal memverifikasi tanda tangan yang disertakan pada paket potongan data media yang dikirimkan oleh *server*, dan menolak untuk menayangkan media tersebut.

Pada skenario ketiga, *server* dan *client* menggunakan kunci private dan publik yang bersesuaian yaitu kunci privat 1 dan kunci publik 1, tetapi terdapat perubahan isi dari paket pada saat transmisi paket dari server kepada *client.* Pada skenario ini, *client* gagal memverifikasi tanda tangan yang disertakan pada paket yang dikirim oleh *server*, dan menolak untuk menayangkan media tersebut.

Dari ketiga skenario tersebut, sistem tanda tangan yang diimplementasikan memenuhi kriteria otentikasi (authentication) dan integritas (*integrity*). Untuk kriteria anti-penyangkalan (*non-repudiation*),

V. KESIMPULAN

Teknologi tanda tangan digital dapat digunakan untuk memberikan layanan otentikasi, integritas pesan, dan anti-penyangkalan terhadap layanan *streaming* media dengan memberikan tanda tangan digital pada setiap potongan data dari media yang ditayangkan. Hal ini dapat membuka peluang untuk dikembangkannya teknologi tanda tangan yang lebih baik untuk *use case* pada layanan *streaming*.

ACKNOWLEDGMENT

Penulis memenjatkan puji syukur kepada Allah SWT yang seatas izin-Nya, penulisan makalah ini dapat terlaksana dengan baik. Ucapan terimakasih penulis sampaikan kepada Bapak Dr. Rinaldi Munir M.T. sebagai dosen pengampu mata kuliah IF 4020 Kriptografi Semester I Tahun 2021/2022 atas semua ilmu dan pengetahuan yang telah diberikan dalam mata kuliah ini. Selain itu, ucapan terima kasih juga disampaikan kepada pihak-pihak lain yang membantu kelancaran penulisan makalah ini.

REFERENCES

[1] Gilbert H., Handschuh H. (2004) Security Analysis of SHA-256 and Sisters. In: Matsui M., Zuccherato R.J. (eds) Selected Areas in Cryptography. SAC 2003. Lecture Notes in Computer Science, vol 3006. Springer, Berlin, Heidelberg

[2] Lee, Jack Y. B. (2005). Scalable Continuous Media Streaming Systems: Architectures, Design, Analysis, and Implementation. Chichestet: John Wiley & Sons, Ltd.

[3] Pfitzmann, Birgit. (1996). Digital Signature Schemes: General Framework and Fail-Stop Signatures. Berlin: Springer-Verlag

[4] Munir, Rinaldi. (2021). Kuliah IF4020 Kriptografi: Algoritma RSA. Bandung.

[5] Munir, Rinaldi. (2021). Kuliah IF4020 Kriptografi: Tanda Tangan Digital. Bandung.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 20 Desember 2021

Muhammad Naufal Fakhrizal (13518115)