Aplikasi Sinkronisasi *File* dengan Metode *Peer-to-peer*

Rifky Hamdani[1], Rinaldi Munir[2]

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

[1]rifkyhamdani@students.itb.ac.id

[2]rinaldi@informatika.org

Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia

Abstrak—**Sinkronisasi *file* adalah suatu proses untuk menyamakan *file* yang dimiliki oleh seorang pengguna dengan pengguna yang lain baik dalam komputer yang sama ataupun berbeda. Sinkronisasi *file* secara konvensional menggunakan server pusat untuk menyimpan duplikat dari *file-file* terbaru yang dimiliki oleh pengguna. Oleh karena itu, diperlukan tempat penyimpanan data yang semakin besar dengan bertambahnya *file* pada pengguna. Oleh karena itu, diperlukan metode lain untuk mengurangi kebutuhan tempat penyimpanan data pada proses sinkronisasi *file*.**

Tujuan utama dari tugas akhir ini adalah menghasilkan aplikasi sinkronisasi *file* yang dapat berfungsi, tanpa menyimpan *file* tersebut pada server dengan memanfaatkan metode *peer-to-peer*. Solusi yang ditawarkan adalah membuat aplikasi yang terdiri dari *tracker* dan *peer*. Langkah berikutnya adalah merancang protokol komunikasi antara *tracker* dengan *peer* dan *peer* dengan *peer*.

Aplikasi *tracker* dan *peer* diimplementasikan sesuai dengan rancangan yang telah dibuat. Implementasi aplikasi tersebut menggunakan bahasa pemrograman JAVA. Setelah aplikasi selesai diimplementasikan, dilakukan pengujian terhadap fungsi-fungsi yang dimiliki oleh aplikasi tersebut.

Kesimpulan utama yang didapatkan dalam tugas akhir ini diantaranya aplikasi berhasil dibangun menggunakan arsitektur *client-server*, sinkronisasi *file* dapat dilakukan tanpa menyimpan *file* pada server dengan menggunakan metode *peer-to-peer*. Dengan menggunakan aplikasi tersebut penggunaan tempat penyimpanan data pada server dapat diminimalkan.

Kata kunci: **sinkronisasi *file, peer-to-peer*.**

I. PENDAHULUAN

Aplikasi sinkronisasi *file* adalah sebuah perangkat lunak yang mengembalikan perubahan sesuai dengan struktur *directory* yang disalin [2]. Sinkronisasi *file* secara umum dapat didefinisikan sebagai proses untuk memastikan *file* yang terdapat dalam dua komputer atau lebih telah diperbaharui melalui aturan dan acuan tertentu tertentu. Sinkronisasi *file* dapat dilakukan secara manual maupun otomatis. Sinkronisasi *file* secara manual misalnya memindahkan *file* dari satu komputer ke komputer lain melalui media portable ataupun melalui *email*. Sinkronisasi *file* secara otomatis dapat dilakukan dengan menggunakan perangkat lunak yang secara otomatis melakukan sinkronisasi *file* apabila terjadi perubahan.

Peer-to-peer adalah istilah dalam jaringan komputer yang berarti tiap komputer dapat berperan sebagai client ataupun server sehingga tidak diperlukan adanya server pusat. Protokol yang memanfaatkan metode *peer-to-peer* contohnya adalah BitTorrent. BitTorrent adalah sebuah protokol untuk pendistribusian *file*. Protokol ini mengenali isi dari URL dan didesain menyatu dengan *web*. Kelebihannya dari HTTP sederhana adalah ketika terjadi lebih dari satu pengunduhan pada *file* yang sama pada saat bersamaan, pengunduh dapat saling mengunggah satu sama lain yang membuat sumber *file* dapat mendukung pengunggahan dalam jumlah yang besar dengan hanya meningkatkan sedikit beban pada sumber *file* [1].

Pada perangkat lunak yang umum digunakan untuk melakukan proses sinkronisasi *file*, proses sinkronisasi *file* dilakukan dengan menyalin *file* tersebut ke server pusat terlebih dahulu. Oleh karena itu, setiap ada perubahan pada *file* pada suatu komputer, komputer tersebut akan mengirimkan salinannya ke server pusat. Apabila ada komputer lain yang ingin melakukan sinkronisasi, komputer tersebut harus mengunduh *file* yang belum tersinkronisasi dari server pusat. Pada metode sinkronisasi tersebut diperlukan server pusat yang memiliki penyimpanan data yang cukup besar. Pada tugas akhir ini akan dibangun sebuah aplikasi sinkronisasi *file* dengan metode *peer-to-peer*. Dengan adanya aplikasi tersebut diharapkan dapat mengurangi kebutuhan penyimpanan data pada *server* pusat.

II. STUDI LITERATUR

A. Sinkronisasi File

Sinkronisasi *file* adalah suatu proses untuk memastikan *file* pada dua komputer atau lebih telah sinkron melalui aturan-aturan tertentu. Pada proses sinkronisasi *file* satu arah hanya salah satu pihak yang melakukan sinkronisasi dan menyalin *file* dari komputer sumber. Proses ini disebut juga *mirroring*. Pada sinkronisasi *file* dua arah pihak pertama dan pihak kedua akan saling menyamakan *file* yang dimilikinya. Sinkronisasi

file umum digunakan sebagai cadangan pada *external hard drives* atau USB *flash drives*. Proses ini juga mencegah penyalinan *file* yang identik secara otomatis sehingga proses sinkronisasi lebih cepat dan tidak memerlukan banyak waktu serta menghindari terjadinya kesalahan [3].

Fitur-fitur umum yang dimiliki aplikasi sinkronisasi *file*:

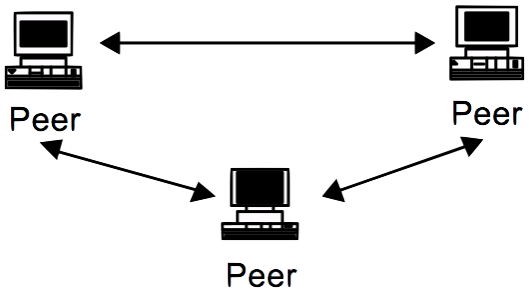
- Enkripsi
- Kompresi data
- Pendeteksi konflik

B. *Peer-to-peer File Sharing*

Peer-to-peer file sharing dapat dibagi menjadi dua menurut arsitekturnya. Arsitektur yang pertama adalah pure *peer-to-peer* yang dapat dilihat pada Gambar 1. Pada arsitektur tersebut tiap *peer* memiliki kemampuan yang sama. Arsitektur ini tidak memiliki server pusat. Arsitektur yang lainnya adalah *server-mediated peer-to-peer* yang digambarkan pada Gambar 2. Pada arsitektur ini tiap komputer memiliki peran yang berbeda. *Server* pusat bertanggung jawab untuk memelihara informasi yang saling dipertukarkan dan merespon permintaan terhadap informasi tersebut. *Peer* bertanggung jawab sebagai penyimpan data dan menentukan informasi yang ingin dibagi kepada *server* pusat. Selain itu *peer* juga dapat mengunduh data dari *peer* lain yang informasinya didapat dari *server* pusat [4].

1. *Pure Peer-to-peer*

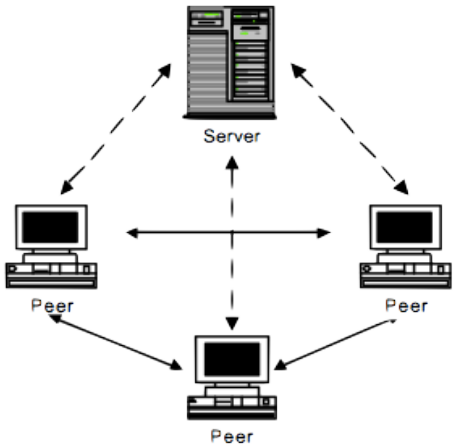
Arsitektur *pure peer-to-peer* tidak memiliki *server* pusat seperti yang diperlihatkan pada Gambar 1. Pada arsitektur ini *peer* secara dinamis saling menemukan dan berinteraksi satu dengan yang lainnya dengan cara mengirim dan menerima pesan digital. Kelebihan dari arsitektur ini adalah tidak bergantung pada tersedianya server untuk menunjukan suatu lokasi untuk berinteraksi dengan *peer* lainnya. Akan tetapi, arsitektur ini memiliki kelemahan yaitu sedikitnya jumlah *peer* yang dapat saling menemukan. Pada skenario ini *peer* dapat menggunakan informasi dari konfigurasi lokal untuk menemukan *peer* lain atau *peer* tersebut melakukan broadcasting and discovery techniques seperti multicast untuk menemukan *peer* lain. Menggunakan IP multicast dapat menimbulkan masalah karena tidak diterapkan di internet. Walaupun memiliki kelemahan skenario ini dapat berguna jika diterapkan pada jaringan intranet [4].



Gambar 1. Arsitektur *pure peer-to-peer*

2. *Server-Mediated Peer-to-peer*

Arsitektur *server-mediated peer-to-peer* bekerja seperti pada arsitektur sebelumnya kecuali pada arsitektur ini *peer* bergantung pada *server* pusat untuk saling menemukan. Pada model ini *peer* akan mengunduh daftar dari *peer* yang terhubung pada jaringan tersebut. Kemudian dari daftar tersebut *peer* dapat mengunduh *file* yang diinginkan dari *peer* yang memilikinya. Arsitektur ini memiliki kelebihan dari arsitektur sebelumnya yaitu jumlah *peer* yang dapat ditemukan jauh lebih besar. Akan tetapi, arsitektur ini sangat tergantung pada *server* pusat. Apabila server pusat mati, *peer* pun tidak dapat menemukan satu sama lain [4].



Gambar 2. Arsitektur server-mediated *peer-to-peer*

C. *Kriptografi*

Pada tugas akhir ini akan dipakai dua algoritma kriptografi dan fungsi *hash*. Algoritma yang dipakai adalah algoritma RSA dan RC4. Sedangkan fungsi *hash* yang akan dipakai adalah SHA-1.

1. *RSA*

RSA merupakan algoritma kunci publik yang berdasar pada sulitnya memfaktorkan sebuah bilangan bulat yang bernilai besar. Nama RSA diambil dari nama penemunya yaitu Ron Rivest, Adi Shamir dan Leonard Adleman. Pengguna algoritma RSA membuat dan menyebarkan hasil perkalian dari dua buah bilangan prima yang bernilai besar bersama dengan bilangan pembantu sebagai kunci publik. Faktor prima harus dirahasiakan. Setiap orang dapat menggunakan kunci publik untuk mengenkripsi pesan, tetapi dengan metode yang saat ini diterbitkan, jika nilai kunci publik cukup besar, hanya orang yang mengetahui faktor primanya yang dapat mendekripsi pesan tersebut [5].

RSA menggunakan kunci publik dan kunci privat. Kunci publik dapat diketahui oleh siapa saja untuk mengenkripsi pesan. Pesan yang terenkripsi dengan kunci publik hanya dapat didekripsi dengan kunci privat.

2. RC4

RC4 merupakan algoritma kriptografi berbasis *stream chiper*. *Stream chiper* adalah algoritma enkripsi yang penting. Algoritma ini mengenkripsi tiap karakter dari pesan tiap satu satuan waktu. Unit atau data pada umumnya sebuah *byte* atau bahkan kadang kadang *bit*. *Stream chiper* secara umum lebih cepat daripada *block chiper* [6].

RC4 digunakan pada implementasi SSL dan WEP. RC4 sangat cepat dan memiliki desain yang sederhana [8]. Pada algoritma ini enkripsi atau dekripsi dapat dilaksanakan pada panjang yang bervariasi. Algoritma ini tidak harus menunggu sejumlah input data tertentu sebelum diproses, atau menambahkan *byte* tambahan untuk mengenkripsi.

3. Fungsi Hash (SHA-1)

Fungsi *hash* adalah fungsi *mapping* yang efisien dalam melakukan *mapping binary string* dengan panjang tak tentu ke binary string dengan panjang tentu yang disebut *hash-values*. SHA-1 adalah fungsi *hash* kriptografi dirancang oleh *National Security Agency* dan diterbitkan oleh NIST sebagai *U.S. Federal Information Processing Standard*. SHA singkatan dari *Secure Hash Algorithm*. Tiga algoritma SHA terstruktur berbeda dan dibedakan sebagai SHA-0, SHA-1, dan SHA-2. SHA-1 adalah sangat mirip dengan SHA-0, tapi mengoreksi kesalahan dalam spesifikasi hash SHA asli yang menyebabkan kelemahan signifikan. Algoritma SHA-0 tidak diadopsi oleh banyak aplikasi. SHA-2 di sisi lain secara signifikan berbeda dari fungsi hash SHA-1 [6].

SHA-1 adalah fungsi hash yang didesain berdasarkan prinsip MD4. Fungsi ini mengaplikasikan paradikma Merkle-Damgard untuk fungsi kompresinya [7]. SHA-1 dapat dapat membuat bilangan unik dari suatu data. Bilangan unik tersebut dapat digunkan untuk membedakan satu data dengan yang lainnya. SHA-1 akan digunakan sebagai referensi apakah suatu *file* dengan nama *file* yang sama telah dimodifikasi atau tidak.

D. Aplikasi Terkait

Pada tugas akhir ini akan dibahas beberapa aplikasi dan penelitian terkait. Awal bab ini akan membahas tentang Dropbox dan BitTorrent.

1. Dropbox

Dropbox adalah layanan gratis yang memungkinkan penggunanya membawa *file* yang dimilikinya dimanapun pengguna berada. Hal tersebut berarti *file* yang telah disimpan di dalam dropbox akan secara otomatis tersimpan dalam semua komputer, telepon genggam, dan *website* dropbox [9].

Dropbox juga membuat pengguna dapat dengan mudah berbagi *file* kepada siapapun yang diinginkan. Apabila komputer pengguna mengalami kerusakan, pengguna tidak perlu khawatir karena *file* yang telah disimpan di dropbox dapat diunduh kembali dari dropbox.

Dropbox memastikan semua *file* yang dimiliki pengguna akan sama dimanapun pengguna berada. Oleh karena, itu pengguna dapat bekerja menggunakan *file* yang ada di dropbox tersebut dimanapun pengguna berada dan apabila belum selesai pekerjaan tersebut pengguna dapat melanjutkannya di tempat yang lain [9].

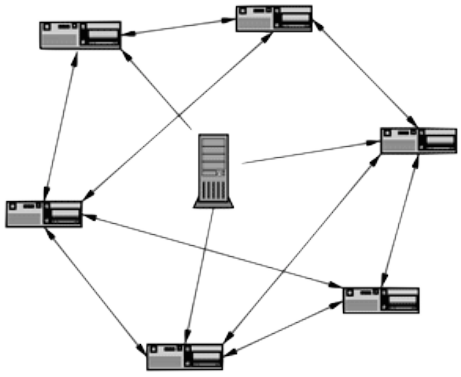
Cara kerja dropbox yaitu apabila suatu komputer memakai dropbox, akan terdapat sebuah *folder* khusus yang disinkronkan dengan *server* yang dimiliki oleh dropbox. Apabila kita membagi *file* kepada pengguna yang lain, pengguna tersebut akan mengunduh *file* yang ada di server dropbox. Apabila pengguna mengubah isi dari folder tersebut, secara otomatis dropbox akan mengunggahnya ke *server* dropbox dan seluruh pengguna yang memiliki *file* tersebut di *server* dropbox akan mensinkronkan *file* sesuai dengan versi yang terbaru.

Dropbox memiliki dua buah fitur keamaman yaitu *secure storage* dan *secure transfer*. *Secure storage* mengamankan data pada *server* menggunakan algoritma AES-256 Standard. *Secure transfer* mengamankan transfer data menggunakan 256-bit SSL (Secure Socket Layer) [9].

2. BitTorrent

BitTorrent adalah aplikasi internet yang paling sukses dalam pendistribusian konten [10]. BitTorrent adalah sebuah protokol untuk pendistribusian *file*. Protokol ini mengenali isi dari URL dan didesain menyatu dengan web. Kelebihannya dari HTTP sederhana adalah ketika terjadi lebih dari satu pengunduhan pada *file* yang sama pada saat bersamaan, pengunduh dapat saling mengunggah satu sama lain yang membuat sumber *file* dapat mendukung pengunggahan dalam jumlah yang besar dengan hanya meningkatkan sedikit beban pada sumber *file*. BitTorrent memiliki dua komponen, yaitu *tracker* dan *peer*. Ilustrasi pengunduhan *file* dengan BitTorrent dapat dilihat pada Gambar 3.

Tracker adalah komponen dalam BitTorrent yang berfungsi sebagai server perantara. *Tracker* memiliki peranan dalam membantu *peer* untuk dapat saling mengenali. *Tracker* menggunakan protokol sederhana yang berada diatas HTTP yang digunakan oleh pengunduh untuk mengirimkan informasi tentang *file* yang diunduh, nomor port, dan informasi sejenis. Kemudian *tracker* membalasnya dengan mengirikan daftar informasi kontak untuk *peer* yang mengunduh *file* yang sama. Pengunggah kemudian menggunakan informasi yang diterima dari *tracker* tersebut untuk dapat saling berkomunikasi antar-*peer* [11].



Gambar 3. Pengunduhan *file* dengan BitTorrent

Semua proses pengunduhan *file* ditangani pada interaksi antar-*peer*. Beberapa informasi tentang pengunggahan dan pengunduhan antar-*peer* dikirimkan ke *tracker*. *Tracker* hanya bertanggung jawab untuk membantu *peer* untuk dapat saling menemukan.

Untuk mengetahui data yang dimiliki oleh tiap *peer*, BitTorrent memotong *file* menjadi bagian-bagian yang berukuran tetap. Setiap pengunduh harus melaporkan bagian-bagian suatu *file* yang dimilikinya. Untuk memastikan integritas data digunakan SHA-1.

Peer secara terus menerus mengunduh bagian-bagian *file* dari *peer* yang lain. *Peer* tidak dapat mengunduh bagian *file* dari *peer* yang tidak terhubung dengan *peer* tersebut [11].

III. ANALISIS DAN DESKRIPSI SOLUSI

A. *Analisis Masalah*

Terdapat tiga analisis masalah yang ada pada tugas akhir ini yaitu pendeteksian perubahan *file*, pengamanan sinkronisasi *file*, dan protokol komunikasi sinkronisasi *file*. Setelah itu dilanjutkan dengan analisis dan perancangan perangkat lunak.

1. Pendeteksian Perubahan File

Perangkat lunak sinkronisasi *file* harus dapat mendeteksi perubahan *file*. Pendeteksian perubahan *file* dapat dilakukan dengan menggunakan berbagai cara. Cara yang paling mudah untuk mendeteksi perubahan *file* adalah dengan memeriksa nama *file* yang dimiliki. Jika nama *file*nya berbeda kemungkinan *file* tersebut telah berubah. Selain itu mendeteksi perubahan *file* dapat dilihat dari nilai hash dari *file* tersebut. Apabila nilai hash dari *file* tersebut berbeda pasti telah terjadi perubahan.

2. Pengamanan Sinkronisasi File

Pada sinkronisasi *file* terjadi transfer data dari satu komputer ke komputer yang lain. Apabila data ini dikirimkan dalam plaintext, akan membuat keamanan dari data tidak terjamin. Oleh karena itu, diperlukan pengamanan data. Pengamanan data dapat dicapai dengan mengenkripsi data yang ditranfer.

3. Protokol Komunikasi Sinkronisasi File

Protokol komunikasi yang akan digunakan pada perangkat lunak sinkronisasi *file* pada tugas akhir ini menggunakan metode *peer-to-peer*. Seperti halnya *peer-to-peer* pada BitTorrent, protokol komunikasi pada tugas akhir ini menggunakan arsitektur *server-mediated peer-to-peer*. Perbedaannya adalah pada BitTorrent transfer data dilakukan tiap potongan *file*, sedangkan pada tugas akhir ini transfer data dilakukan tiap *file*. Terdapat dua buah protokol komunikasi yaitu protokol komunikasi antara *peer* dan server pusat atau dalam hal ini disebut *tracker* dan protokol komunikasi antar-*peer*. Pada protokol komunikasi antara *peer* dan *tracker*, data yang ditransfer adalah berupa alamat semua *peer* yang melakukan proses sinkronisasi dengan *peer* tersebut, data dan metadata untuk proses authentication.

B. *Deskripsi Solusi*

Solusi yang akan diterapkan yaitu dengan membuat aplikasi sinkronisasi *file* dengan metode *server-mediated peer-to-peer*. Pada metode tersebut harus ada 2 aplikasi yaitu *peer* dan *tracker*.

Beberapa kemampuan yang harus ada dalam aplikasi *peer* yaitu:

1. Sinkronisasi *file* dilakukan secara otomatis tanpa ada perintah dari pengguna aplikasi.
2. Aplikasi dapat mendeteksi secara otomatis adanya perubahan pada *file* kemudian mengirimkan informasi perubahan tersebut ke server pusat.
3. Antar-*peer* dapat saling melakukan sinkronisasi *file*.
4. Proses sinkronisasi memakai enkripsi.

Selain aplikasi *peer* perlu dibuat aplikasi *tracker* yang harus memiliki kemampuan yaitu:

1. Mengirimkan data yang diperlukan untuk proses sinkronisasi kepada *peer* apabila ada permintaan dari *peer* tersebut.
2. Dapat memberitahu *peer* apabila terjadi perubahan *file* pada *peer* yang lain.
3. Menyimpan daftar data terbaru yang dimiliki oleh semua *peer*.

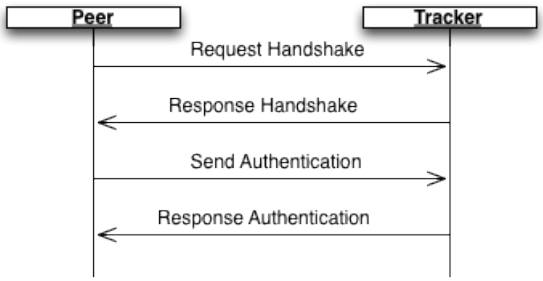
C. *Deskripsi Protokol Komunikasi*

Deskripsi protokol komunikasi ini terdiri dari perancangan protokol dan diagram komunikasi dari sinkronisasi *file*.

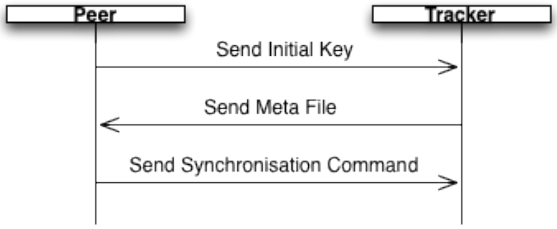
1. Protokol Tracker-to-peer

Protokol *tracker-to-peer* digunakan untuk proses komunikasi antara *tracker* dan *peer*. Proses Sinkronisasi terjadi pada *Tracker*, bukan pada *peer*. Proses sinkronisasi pada *tracker* dapat dilihat pada Gambar 4. *Tracker* memiliki Daftar *file* dan folder terbaru dilengkapi *path* dan *hash-value* pada tiap *file*. Pada daftar tersebut juga terdapat alamat *peer* yang memiliki *file* atau *folder* tersebut.

Sedangkan *peer* memiliki daftar *file* dan *folder* yang dimiliki pada *folder root* dilengkapi dengan *hash-value* pada tiap *file*. Proses authentication antara *peer* dan *tracker* dapat dilihat pada Gambar 5, sedangkan proses sinkronisasinya dapat dilihat pada Gambar 6.



Gambar 4. Proses authentication antara *peer* dan *tracker*



Gambar 5. Proses sinkronisasi antara *peer* dan *tracker*

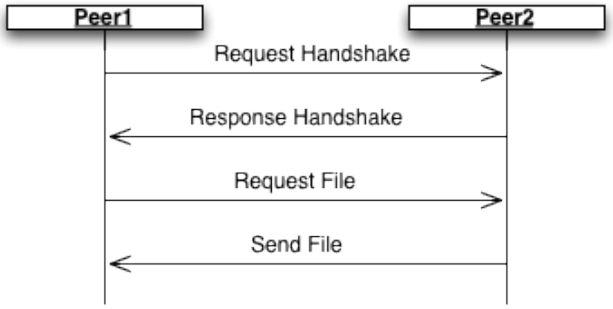
Pada protokol ini memiliki beberapa pesan yang digunakan dalam proses komunikasi yaitu:

- *Handshaking*
- *Send authentication*
- *Response authentication*
- *Send initial key*
- *Send key*
- *Send logout*
- *Send meta files*
- *Send Synchronization Command*

2. Protokol *Peer-to-peer*

Protokol ini merupakan protokol yang digunakan untuk berkomunikasi antara *peer* dan *peer* yang lain. Proses komunikasi *peer-to-peer* dapat dilihat pada Gambar III-3.Pada protokol ini memiliki beberapa pesan yang digunakan dalam proses komunikasi yaitu:

- *Handshaking*
- *Request file*
- *Send file*



Gambar 6. Proses komunikasi *peer-to-peer*

1. *Meta File*

Meta *File* adalah *file* yang menyimpan kumpulan dari metadata *file* yang dimiliki oleh *peer* atau *tracker*. Pada tugas akhir ini terdapat dua buah meta *file* yaitu meta *file tracker* dan meta *file peer*.

Meta file tracker digunakan untuk menyimpan kumpulan *metadata file* paling baru yang dimiliki oleh semua *peer* dan berada di *tracker*. Meta *file tracker* menyimpan kumpulan metadata *file* yang terdiri dari enam bagian yaitu:

- *parent*
- *name*
- *isFile*
- sha
- *timeAdded*
- *owners*

Meta file peer digunakan untuk menyimpan kumpulan metadata *file* paling baru yang dimiliki *peer*. Meta *file peer* menyimpan kumpulan metadata *file* yang terdiri dari lima bagian yaitu:

- *parent*
- *name*
- *isFile*
- sha
- *timeAdded*

4. *Daftar Perintah Sinkronisasi*

Daftar perintah sinkronisasi digunakan untuk merepresentasikan kumpulan perintah sinkronisasi yang dikirimkan oleh *tracker* kepada *peer*. Daftar perintah sinkronisasi berisi kumpulan perintah sinkronisasi yang memiliki enam bagian yaitu:

- code
- *parent1*
- *parent2*
- *name1*
- *name2*
- *addressPort*

5. *Skenario Operasi Pada File*

Terdapat beberapa skenario yang dapat terjadi pada *file* yang terdapat pada aplikasi *peer* yaitu:

- Penambahan *file*
- Perubahan *file*
- Penghapusan *file*
- Pengubahan nama *file*
- Pemindahan *file*
- Operasi konkuren

IV. PENGUJIAN DAN ANALISIS

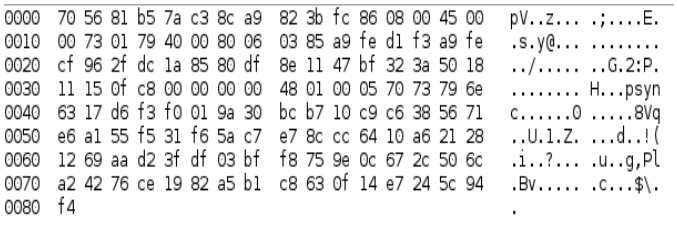
A. Pengujian Keamanan

Hasil pengujian keamanan dapat dilihat pada Gambar 7, Gambar 8, dan Gambar V-3.

```
0000  8c a9 82 3b fc 86 70 56  81 b5 7a c3 08 00 45 00   ...;..pV ..z...E.
0010  00 40 b2 68 40 00 ff 06  d3 c7 a9 fe cf 96 a9 fe   .@.h@... ........
0020  d1 f3 1a 85 2f dc 47 bf  32 22 80 df 8e 10 50 18   ..../.G. 2"....P.
0030  40 00 95 59 00 00 00 00  15 00 00 12 70 32 70 73   @..Y.... ....p2ps
0040  79 6e 63 68 72 6f 6e 69  7a 61 74 69 6f 6e         ynchroni zation
```

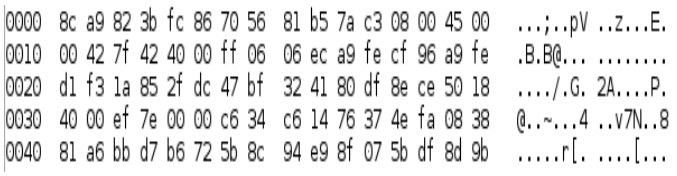
Gambar 7. Tampilan pada Wireshark saat handshaking

Pada Gambar 7 terlihat saat melakukan *handshaking* terlihat data teks berupa p2psynchronization.



Gambar 8. Tampilan pada Wireshark saat *authentication*

Pada Gambar 8 terlihat saat melakukan *authentication* terlihat data teks *username* berupa psync, tetapi *password* tidak terlihat.



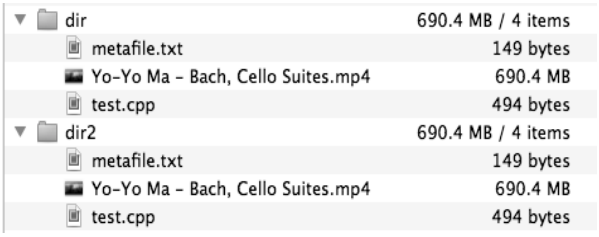
Gambar 9. Tampilan pada Wireshark saat komunikasi data

Pada Gambar 9 terlihat saat melakukan komunikasi data tidak terlihat data yang dikirimkan.

Dari hasil pengujian keamanan yang dilakukan pada bab sebelumnya terlihat bahwa penyadapan dengan menggunakan Wireshark pada proses komunikasi tersebut tidak dapat melihat data yang dipertukarkan. Oleh karena itu, sistem ini dapat dikatakan aman dari penyadapan.

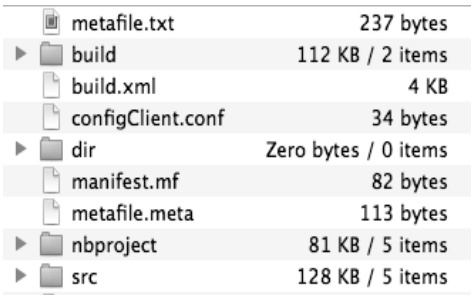
B. Pengujian Penggunaan Tempat Penyimpanan

Hasil pengujian keamanan dapat dilihat pada Gambar 10 dan Gambar 11.



Gambar 10. Gambar Penyimpanan *File* pada *Peer*

Pada Gambar 10 terlihat besarnya termpat yang dibutuhkan oleh *peer* pertama pada *folder* dir sebesar 690,4 MB dan *peer* kedua pada *folder* dir2 690,4 MB



Gambar 11. Penyimpanan *File* pada *Tracker*

Pada Gambar 11 terlihat besarnya termpat yang dibutuhkan oleh *tracker* yaitu pada *file* metafile.txt sebesar 237 bytes.

Dari hasil pengujian kebutuhan tempat penyimpanan pada bab sebelumnya dapat dilihat kebutuhan pada *tracker* dan *peer. Tracker* menggunakan data sekitar 237 *bytes,* sedangkan *peer* sekitar 690,4 MB. Apabila *tracker* harus menyimpan juga data yang dimiliki oleh *peer*, tempat penyimpanan yang diperlukan oleh *tracker* naik menjadi 690,4 MB lebih. Oleh karena itu, sinkronisasi *file* dengan metode *peer-to-peer* memerlukan jauh lebih sedikit tempat penyimpanan pada *tracker* daripada yang dibutuhkan oleh sinkronisasi *file* konvensional yang harus menyimpan *file* yang dimiliki pengguna di *server* pusat.

V. KESIMPULAN

Dari aplikasi yang telah dibuat dihasilkan kesimpulan sebagai berikut:

1. Aplikasi sinkronisasi *file* dengan metode *peer-to-peer* berhasil dibangun dengan menggunakan arsitektur server-mediated *peer-to-peer*.
2. Aplikasi sinkronisasi *file* yang dibangun dapat dikatakan aman dari tindakan penyadapan.
3. Penggunaan tempat penyimpanan pada *tracker* sangat sedikit dibandingkan dengan server yang digunakan pada sinkronisasi *file* secara konvensional.

REFERENSI

[1] Cohen, Bram (2009). The BitTorrent Protocol Specification http://www.bittorrent.org/beps/bep_0003.html, diakses 17 November 2012.

[2] Benjamin, P., & Vouillon, J. (2004). What's in Unison? A Formal Specification and Reference Implementation of a *File* Synchronizer. Technical Report MS-CIS-03-36 Department of Computer and Information Science University of Pennsylvania.

[3] Tridgell, A. (1999). Efficient algorithms for sorting and synchronization. PhD thesis. The Australian National University.

[4] Siu Man, L., & Sai Ho, K. (2002) Interoperability of *Peer*-To-*Peer File* Sharing Protocols. ACM SIGecom Exchanges, Vol. 3, No. 3, August 2002, Halaman 25-33.

[5] Rivest, R., & Shamir, A., & Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM 21 (2): 120–126. http://people.csail.mit.edu/rivest/Rsapaper.pdf, diakses 4 November 2012.

[6] Menezes, A., & Oorschot, P., & Vanstone, S. (1997) Handbook of Applied Cryptography, CRC Press, Inc.

[7] Manuel, Stephane. (2007). Classification and Generation of Disturbance Vectors for Collision Attacks against SHA-1. CRI - Paris Rocquencourt

[8] Paul, Souradyuti and Preneel, Bart. (2002). Analysis of Non-fortuitous Predictive States of the RC4 Keystream Generator. Katholieke Universiteit Leuven, Dept. ESAT/COSIC

[9] Dropbox, Dropbox Security Overview, https://www.dropbox.com/privacy, 7 Januari 2013.

[10] Zhang, C. (2011). Unraveling the BitTorrent Ecosystem. IEEE Transactions on parallel and distributed systems, Vol. 22, No. 7.

[11] Cohen, Bram. (2003). Incentives Build Robustness in BitTorrent. http://www.ittc.ku.edu/~niehaus/classes/750-s06/documents/BT-description.pdf, diakses 4 November 2012