SISTEM PENGAMANAN DATA PADA MEDIA PENYIMPANAN *CLOUD* DENGAN METODE *SPLIT DATA DISTRIBUTION*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Sarwo Hadi Wibowo
NIM: 115060807111119

PROGRAM STUDI INFORMATIKA/ILMU KOMPUTER
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

SISTEM PENGAMANAN DATA PADA MEDIA PENYIMPANAN *CLOUD*
DENGAN METODE *SPLIT DATA DISTRIBUTION*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Sarwo Hadi Wibowo
NIM: 115060807111119

Skripsi ini telah diuji dan dinyatakan lulus pada
18 Januari 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Sabriansyah Rizqika A., S.T., M.Eng.	Eko Sakti P., S.Kom., M.Kom.
NIP. 19820809 201212 1 004	NIK. 860805 06 1 1 0252

Mengetahui
Ketua Program Studi Informatika/Ilmu Komputer

Drs. Marji, M.T.
NIP. 19670801 199203 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 26 Januari 2016

Sarwo Hadi Wibowo

NIM: 115060807111119

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah *subhanahu wa ta'ala* yang hanya karena limpahan rahmat dan karunia-Nya penulis dapat menyelesaikan skripsi dengan judul "Sistem Pengamanan Data Pada Media Penyimpanan *Cloud* Dengan Metode *Split Data Distribution*". Shalawat beserta salam semoga senantiasa tercurahkan kepada Nabi Muhammad *salallahu 'alaihi wa salam,* kepada keluarganya, para sahabatnya, dan umatnya hingga akhir zaman. *Aamiin*.

Penulis menyadari bahwa penyusunan dan penulisan laporan skripsi ini tidak akan berjalan dengan baik tanpa adanya bantuan, bimbingan serta dukungan dari berbagai pihak, khususnya dosen pembimbing dan teman-teman penulis. Oleh karen itu dalam kesempatan ini penulis dengan senang hati menyampaikan terima kasih kepada pihak tersebut diantaranya:

1. Bapak Sabriansyah Rizqika Akbar, S.T., M.Eng. selaku dosen pembimbing I yang selalu bijaksana memberikan bimbingan, nasehat dan kepercayaan dalam menyelesaikan penelitian dan penulisan skripsi ini.
2. Bapak Eko Sakti Pramukantoro, S.Kom., M.Kom., selaku dosen pembimbing II yang telah memberikan bimbingan, saran, waktu, dan ilmu yang sangat bermanfaat bagi penulis.
3. Kedua orang tua penulis, Ragil Samidi dan Agus Sulistyowati atas kasih sayang, kesabaran, doa serta dukungan moril maupun materiil.
4. Kakak dan adik penulis, Siti Hutami H., Sarwo Hadi N., dan Siti Hutami M., yang telah banyak memberikan motivasi, semangat dan doa demi lancarnya penyusanan skripsi ini.
5. Danang Januarko, Zanwar Yoga P., Ariotomo Satyo W., Ana Fachrina, M. Riza A., Taufiq Hamidhi, beserta seluruh teman-teman TIF-A 2011 atas bantuan, nasehat dan kebersamaan kalian.
6. Semua teman-teman PTIIK, khususnya Informatika/Ilmu Komputer 2011 yang telah memberikan bantuan dan dukungannya selama ini.
7. Segenap dosen dan karyawan PTIIK Universitas Brawijaya yang telah banyak membantu penulis selama mengikuti perkuliahan dan penyusunan skripsi ini.
8. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun tidak langsung demi terselesaikannya skripsi ini.

Semoga Allah *subhanahu wa ta'ala* memberikan balasan yang lebih baik atas bantuan doa yang diberikan. Dengan segala keterbatasan materi dan pengetahuan yang dimiliki penulis, skripsi ini masih jauh dari kata sempurna. Oleh karena itu kritik dan saran yang membangun sangat kami harapkan. Akhirnya, semoga skripsi ini dapat bermanfaat dan berguna bagi pembaca terutama mahasiswa PTIIK Universitas Brawijaya.

Malang, 26 Januari 2016

Penulis

sarwohadiwibowo@yahoo.co.id

ABSTRAK

Saat ini kebutuhan akan sumber daya (infrastruktur, media penyimpanan, aplikasi dan platform) yang terkomputasi bisa dikatakan menjadi sebuah kebutuhan, misalnya sumber daya media penyimpanan fisik. Walaupun menjadi sebuah kebutuhan, media penyimpanan fisik yang digunakan pada umumnya memiliki keterbatasan dalam hal aksesibilitas. *Cloud computing* adalah sebuah model komputasi yang memungkinkan penggunanya untuk menggunakan sumber daya (penyimpanan) pada jaringan internet sehingga pengguna dapat mengakses sumber daya yang dibutuhkan dari mana saja dan kapan saja. Akan tetapi, dibalik kelebihan dari *cloud computing* terdapat juga ancaman yang ditimbulkan, terutama pada sisi *security* (keamanan). Salah satu celah keamanan yang mungkin ditimbulkan adalah *loss of privacy/untrusted cloud provider* dimana kerahasian data pengguna tidak dapat dijamin dan juga dimungkinkan terjadinya *theft of information*. Pada penelitian ini, solusi yang ditawarkan adalah dengan membuat sebuah sistem yang mampu mencegah adanya *loss of privacy & theft of information* dengan cara membagi data menjadi beberapa bagian (*splitting*) kemudian didistribusikan pada beberapa penyedia media penyimpanan *cloud* yang berbeda (*multiple cloud storage*). Secara spesifik, pembagian data dimaksudkan agar informasi tidak dapat dibaca sebelum data disatukan kembali. Selanjutnya distribusi data pada *multiple cloud storage* dibutuhkan agar kerahasiaan data lebih terjaga. Konsep ini sebenarnya dapat dianalogikan dengan permainan *puzzle* pada umumnya. Pada permainan *puzzle*, informasi bisa didapat hanya jika bagian-bagian *puzzle* terkumpul dan tersusun dengan benar. Selain itu pada sistem ini terdapat fitur hash yang digunakan untuk menjamin integritas/keaslian data. Dari hasil pengujian dan analisis yang dilakukan pada penelitian ini menunjukkan bahwa secara fungsionalitas dan segi keamanan sistem dapat berjalan dengan baik.

Kata kunci: *Multiple cloud*, keamanan *cloud*, penyimpanan, data *splitting*, distribusi.

ABSTRACT

Computed resource (such as infrastructure, media storage, application and platform) can be said to be a necessity, such as physical storage media resources. Although it become a necessity, physical storage media that is used generally have limitations in terms of accessibilty. Cloud computing is a computation model that allows users to use resources (storage) on the internet network so that the user can access the needed resources anywhere and anytime. However, behind the advantages of cloud computing there is also the threat posed, especially on security. One of the vulnerabilities that may arise is the loss of privacy/untrusted cloud provider that confidentiality can't be guaranteed and there might be theft of information. In this research, the solution offered to solve this problem is to build a system that can prevent loss of privacy & theft of information by splitting the data into several parts and then distribute to several different providers of cloud storage (multiple cloud storage). Specifically, data splitting is used so that the information could not be read before uniting the data. Furthermore, distribution of data across multiple cloud storage needed to keep data confidentiality. This concept is actually analogous to a puzzle game in general. In a puzzle game, the information can be obtained only if the puzzle parts are collected and stacked correctly. In addition, there is a hash feature that can be used to ensure data integrity. Test analysis and result in this research shows that the system's functionality and security work well.

Keyword: Multiple cloud, cloud security, storage, data splitting, distribution.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR SOURCE CODE

BAB 1 PENDAHULUAN

1.1 Latar belakang

Saat ini kebutuhan akan sumber daya (infrastruktur, media penyimpanan, aplikasi dan *platform*) yang terkomputasi bagi individu atau institusi bisa dikatakan menjadi sebuah kebutuhan. Sumber daya untuk penyimpanan misalnya, hampir setiap individu yang ada pada masa perkembangan teknologi seperti saat ini, sangat jarang yang tidak mengenal *harddisk*, *flash drive*, *memory card* dan sebagainya. Akan tetapi ketersediaan data pada media penyimpanan yang ada saat ini memiliki keterbatasan dalam mobilitas atau pengaksesan datanya. Karena secara fisik pengguna harus berada dekat dengan media penyimpanan tersebut.

Seiring berkembangnya teknologi, bekembang pula sebuah arsitektur *cloud computing* atau Komputasi Awan. *Cloud computing* adalah sebuah model komputasi yang memungkinkan penggunanya untuk menggunakan sumber daya (infrastruktur, media penyimpanan, aplikasi, dan *platform*) pada jaringan internet atau yang juga sering disebut teknologi *On-Demand*. Maksud dari teknologi *On-Demand* adalah teknologi yang berbasiskan pada permintaan pengguna. Di mana sekarang ini, banyak dari individu atau institusi baik yang berskala kecil maupun besar berpikir untuk memindahkan sumber daya mereka pada arsitektur *cloud*. Sebenarnya pengguna cukup menyewa pada penyedia jasa layanan *cloud* ataupun menggunakan beberapa layanan *cloud* gratis yang ada, tidak perlu membangun arsitektur sendiri dari nol. Setiap pengguna berharap dengan memindahkan sumber daya pada lingkungan *cloud* (internet) akan memudahkan mereka dalam berbagi sumber daya antar individu, internal institusi maupun eksternal institusi. Selain itu, jika dilihat dari segi ekonomis, penerapan *cloud* mampu menghemat pengeluaran dan juga dengan *cloud computing*, fleksibilitas layanan dapat ditingkatkan (Pratiwi, 2011).

Akan tetapi, dibalik kelebihan dari *cloud computing* terdapat juga ancaman yang ditimbulkan. Ancaman utama dari *cloud computing* adalah pada sisi *security* (keamanan). Karena dengan memanfaatkan *cloud computing,* setiap resource dan informasi yang dimiliki disimpan pada layanan *cloud* atau jaringan internet (Adrianus, 2010). Salah satu celah keamanan yang mungkin ditimbulkan adalah *loss of privacy/untrusted cloud provider* dan *theft of information,* dimana kerahasian data pengguna tidak dapat dijamin. Karena dengan memanfaatkan *cloud computing* sebagai media penyimpanan (*cloud storage*) bukan tidak mungkin jika pengguna akan menyimpan berkas-berkas yang dianggap penting/rahasia. Padahal jika diinginkan, penyedia layanan dapat dengan mudah mengakses data user tersebut (Ramadhan, 2011). Bahkan mungkin juga terjadi serangan dari pihak luar seperti MITM Attack *(Man In The Middle Attack)* yang dapat mengubah data Anda sebelum sampai pada tujuan atau dengan teknik serangan-serangan lainnya yang dilakukan untuk mengubah/mencuri data.

Split adalah sebuah proses pemecahan data yang ditentukan berdasarkan konsep *byte* (ukuran pecahan). Pada dasarnya ketika sebuah data/berkas dipecah

menjadi beberapa bagian, data tidak akan dapat dibaca secara utuh. Konsep ini dapat dianalogikan dengan permainan *puzzle* pada umumnya. Pada permainan *puzzle*, informasi bisa didapat hanya jika bagian-bagian *puzzle* terkumpul dan tersusun dengan benar. Dari sini dapat dipahami dengan terbaginya data menjadi beberapa bagian mampu meningkatkan kerahasiaan informasi.

Sebelumnya telah dilakukan penelitian berkaitan dengan keamanan data pada *cloud* yang dilakukan untuk mejaga kerahasiaan data pada media penyimpanan *cloud* publik. Pada penelitian yang berjudul "*Implementing Encryption Algorithms to Enhance Data Security of Cloud in Cloud Computing*" proses pengemanan data dilakukan dengan memanfaatkan enkripsi dua langkah, di mana data akan dikirimkan pada *server Chiper Cloud* melalui koneksi yang aman (HTTPS) untuk dienkripsi dan dilanjutkan proses *upload* pada layanan *cloud*.

Penelitian ini dilakukan untuk menyelesaikan permasalahan yang sama dengan memanfaatkan metode yang berbeda. Solusi yang ditawarkan adalah dengan membuat sebuah sistem yang mampu meningkatkan keamanan (privasi) dengan metode membagi data menjadi beberapa bagian (*splitting*) (Yao & Yao, 2012) kemudian data didistribusikan pada beberapa penyedia media penyimpanan *cloud* yang berbeda (*multiple cloud storage*).

1.2 Rumusan masalah

Berdasarkan uraian latar belakang di atas, maka dapat dirumuskan permasalahan pada penelitian ini yaitu sebagai berikut:

1. Bagaimana merancang sebuah sistem yang dapat mencegah *loss of privacy & theft of information* pada media penyimpanan *cloud*?
2. Bagaimana mengimplementasikan sistem yang dirancang dalam mencegah *loss of privacy & theft of information* pada media penyimpanan *cloud*?
3. Bagaimana jaminan keamanan data yang diberikan oleh sistem yang dirancang?

1.3 Tujuan

Tujuan dari penelitian ini adalah untuk membangun sebuah aplikasi yang mampu meningkatkan kerahasiaan data pada media penyimpanan *cloud* dengan memanfaatkan metode *splitting*, serta mendistribusikannya pada beberapa penyedia layanan media penyimpanan *cloud* yang berbeda (*multiple cloud storage*) sehingga data tidak terkumpul pada satu layanan saja.

1.4 Manfaat

Menyediakan aplikasi yang dapat meningkatkan privasi kepemilikan data dari pihak penyedia jasa layanan media penyimpanan *cloud* dengan melakukan *splitting* pada data yang akan disimpan.

1.5 Batasan masalah

Adapun batasan permasalahan pada penelitian ini adalah:

1. Aplikasi pada penelitian ini merupakan *Desktop-Base Application* yang dibangun menggunakan Bahasa Pemrogramman Java
2. Aplikasi yang dibangun memanfaatkan 2 *cloud storage,* yaitu Dropbox & Google Drive sebagai media penyimpanannya
3. Otentikasi menggunakan protokol OAuth 2.0 dilakukan setiap aplikasi dijalankan (*no session token*)
4. Batas minimal berkas yang akan di-*upload* adalah 6 Mb

1.6 Sistematika pembahasan

Sistematika pembahasan dan penyusunan penelitian yang direncanakan adalah sebagai berikut:

BAB I PENDAHULUAN

Pendahuluan terdiri dari latar belakang, perumusan masalah, pembatasan masalah, tujuan dan manfaat serta sistematika pembahasan dari penelitian ini.

BAB II LANDASAN KEPUSTAKAAN

Pada bab ini dilakukan pengajian pustaka terhadap teori yang berkaitan dan menunjang dalam penyelesaian penelitian ini. Aplikasi pemecah dan pendistribusian data pada beberapa layanan media penyimpanan *cloud* dijadikan langkah untuk mengamankan data. Maka dari itu diperlukan metode yang dapat diimplementasikan pada penelitian ini. Metode *splitting* dipilih karena untuk memecah data sebelum di-*upload* pada media penyimpanan *cloud* dan Bahasa Pemrograman Java digunakan sebagai sarana untuk membangun aplikasi ini. Dasar teori yang diambil berasa dari jurnal, situs web, paper dan sumber referensi lainnya yang berhubungan dengan topik penelitian.

BAB III METODOLOGI PENELITIAN DAN PERANCANGAN

Bab ini membahas tentang metode yang digunakan dalam penulisan penelitian ini. Penjelasan langkah-langkah seperti, studi literatur, perancangan, implementasi, pengujian, dan analisis serta pengambilan kesimpulan dan saran. Selain itu, pada bab III ini juga dijelaskan mengenai rancangan dari sistem pengamanan berkas dengan metode *splitting* dan distribusi pada beberapa layanan media penyimpanan *cloud*. Sebuah skenario pengujian mencakup tujuan penelitian juga disampaikan pada bab ini.

BAB IV IMPLEMENTASI DAN PENGUJIAN

Bab ini menjelaskan mengenai penerapan dari arsitektur sistem secara terperinci dengan langkah-langkah pengerjaan serta menampilkan gambar dari implementasi yang telah dilakukan. Implementasi dilakukan berdasarkan sistematika yang telah dibuat. Kemudian pengujian terhadap sistem dilakukan berdasarkan skenario yang juga telah ditentukan pada bab sebelumnya.

BAB V HASIL PENGUJIAN DAN ANALISIS

Bab ini membahas mengenai hasil pengujian dan analisis terhadap sistem yang telah dibangun. Hasil pengujian didapatkan dari pengujian yang telah dilakukan sesuai dengan skenario yang telah ditentukan. Dari hasil pengujian tersebut dapat dilakukan analisis terhadap fungsionalitas dan keamanan yang diberikan oleh sistem, serta mengetahui bagaimana sistem mencapai tujuan.

BAB VI PENUTUP

Memuat kesimpulan dari penelitian yang telah dilakukan serta saran untuk pengembangan sistem lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini akan dibahas sedikit tentang kajian pustaka berdasarkan penelitian sebelumnya, dan apa saja dasar teori yang akan digunakan penulis sebagai penunjang dalam menyusun penelitian. Diantara dasar teori yang dibahas antara lain: *Cloud computing*, API (*Application Programming Interface*), *split & merge* & fungsi hash.

2.1 Kajian pustaka

Penelitian sebelumnya berkaitan dengan keamanan data pada *cloud* dilakukan untuk mejaga kerahasiaan data pada media penyimpanan *cloud*. Pada penelitian yang berjudul "*Implementing Encryption Algorithms to Enhance Data Security of Cloud in Cloud Computing*" proses pengemanan data dilakukan dengan memanfaatkan enkripsi dua langkah, di mana data akan dikirimkan pada *server Chiper Cloud* melalui koneksi yang aman (HTTPS) untuk dienkripsi dan dilanjutkan proses *upload* pada layanan *cloud*.

Pada penelitian ini dihasilkan sebuah sistem yang dapat menjaga kerahasiaan data secara keseluruhan, mulai dari proses *upload* hingga data sampai pada penyimpanan *cloud*.

2.2 Cloud computing

Cloud computing dapat diartikan sebagai sebuah model komputasi yang memungkinkan penggunanya untuk memanfaatkan sumber daya (jaringan, *server*, media penyimpanan, aplikasi dan layanan) yang terdapat dalam sebuah jaringan *cloud*. *Cloud* di sini hanya sebagai metafora dari internet, di mana pengguna dapat mengaksesnya di mana saja kapan saja. Berbeda dengan "*local computing"*, yang mana pengguna secara fisik harus berada dekat dengan sumber daya yang dibutuhkan untuk mengaksesnya (contoh: *hard drive*). Keterbatasan mobilitas pada "*local computing*" inilah yang menjadikan dasar arsitektur *cloud computing.* Pada *cloud computing*, pengguna dapat dengan mudah berbagi informasi dengan pengguna lain tanpa saling bertemu secara fisik, bahkan pengguna juga dapat meningkatkan sumber daya (infrastruktur) dengan mudah. Oleh karea itu jika dilihat dari segi ekonomis, penerapan *cloud computing* dapat menghemat biaya pengeluaran sumber daya , karena pengguna tidak perlu mengalokasikan biaya tersendiri untuk pembelian perangkat keras ataupun perangkat lunak. Pengguna hanya perlu menyediakan biaya sewa untuk setiap sumber daya yang dibutuhkan pada layanan *cloud computing* (Pratiwi, 2011).

Menurut NIST (*National Institute of Standards and Technology*), terdapat 5 karakteristik yang menjadikan sebuah sistem dapat dikatakan sebagai *cloud computing*, yaitu (Budianto, 2012):

1. ***Resource Pooling***
   Sumber daya (infrastruktur fisik ataupun virtual) komputasi , disediakan oleh penyedia layanan bagi pelanggan.

2. ***Broad Network Access***
   Kemudahan akses pada jaringan *cloud*, sehingga dapat diakses oleh *mobile device* seperti: *notebook*, *smartphone*, dll.

3. ***Measured Service***
   Adanya transparansi antara penyedia layanan *cloud* & pengguna layanan *cloud*. Seperti layanan untuk optimasi dan monitoring layanan (pelanggan) sehingga pelanggan dapat melihat sumber daya (*bandwidth*, media penyimpanan, jumlah pemakai, dll).

4. ***Rapid Elasticity***
   Pengguna dapat memakai layanan secara dinamis, sehingga pengguna dapat meningkatkan atau menurunkan spesifikasi layanan dengan mudah.

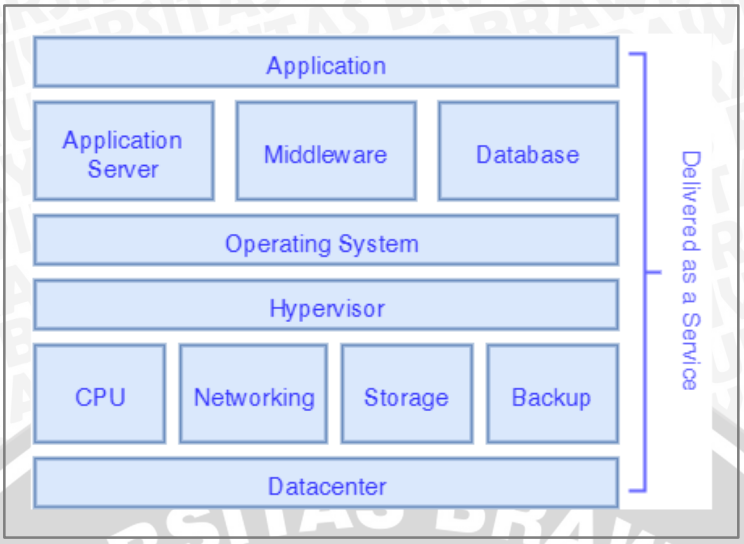
5. ***Self Service***
   Konfigurasi yang mudah, sehingga pengguna dapat melakukan instalasi secara mandiri tanpa harus berinteraksi langsung dengan penyedia layanan *cloud*

Kelima karakteristik tersebut seharusnya dimiliki oleh setiap penyedia layanan *cloud computing* jika mereka ingin diakui sebagai penyedia layanan *cloud computing*. Jika salah satu karakteristik tersebut tidak dimiliki, mungkin mereka belum layak untuk disebut sebagai penyedia layanan *cloud computing*.

Selain karakteristik dari cloud computing, NIST membagi model layanan *cloud computing* menjadi tiga kategori, yaitu:

- ***Software as a Service* (Saas)**

  SaaS merupakan salah satu model layanan *cloud computing* berupa perangkat lunak/aplikasi yang berjalan di sisi server penyedia layanan. Dengan layanan ini pengguna tidak akan direpotkan untuk menginstal perangkat lunak, mengelola ataupun mengendalikan infrastruktur jaringan, *server*, sistem operasi dll. Cukup menggunakan *web browser* atau aplikasi tertentu (dari penyedia layanan) untuk mengaksesnya. Keuntungan lain yang didapat pada layanan ini adalah pengguna akan terbebas dari permasalahan lisensi perangkat lunak, yang mana masalah inilah yang paling sering dihadapi oleh pengguna perangkat lunak saat ini. Contoh dari layanan SaaS diantaranya: GoogleDocs, Microsoft Office online, Adobe Creative Clouds, dan layanan email berbasis web pada umunya (Yahoo, Gmail, dll). Berikut gambaran mengenai pemodelan layanan SaaS.

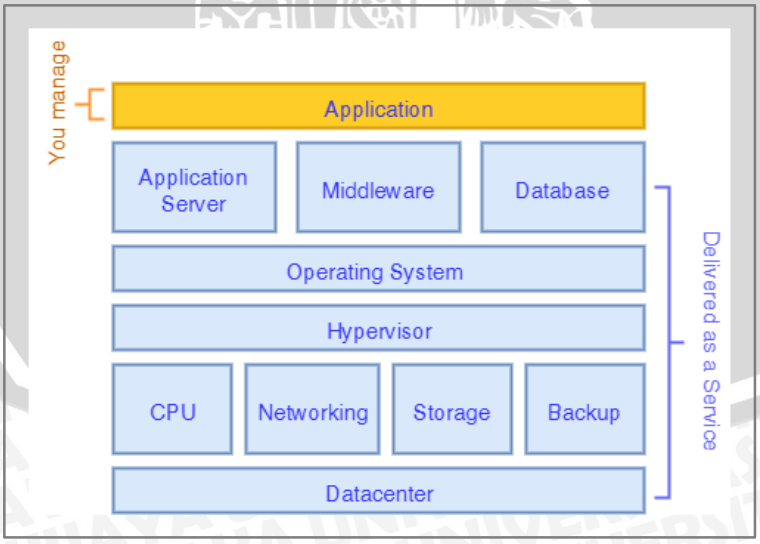


Gambar 2.1 Pemodelan SaaS

(Sumber: Adrianus, 2010)

- ***Platform as a Service*** **(PaaS)**

Paas adalah model layanan *cloud computing* yang hampir memiliki kesamaan dengan SaaS, hingga ada yang mengatakan bahwa layanan ini hanya sebuah "situs web" yang berada di lingkungan *cloud computing*. Pada dasarnya Pengguna hanya perlu mengembangkan aplikasi tersendiri yang didukung oleh penyedia layanan. Di sisi pengelolaan, konsumen tidak perlu mengelola infrastruktur jaringan, *server*, dll, namun pengguna memiliki kontrol atas aplikasi yang dikembangkannya. Contoh aplikasi yang menggunakan teknologi *cloud computing* sebagai PaaS adalah Windows Azure, Google App Engine & Amazon Web Service. Berikut gambaran mengenai pemodelan layanan PaaS.

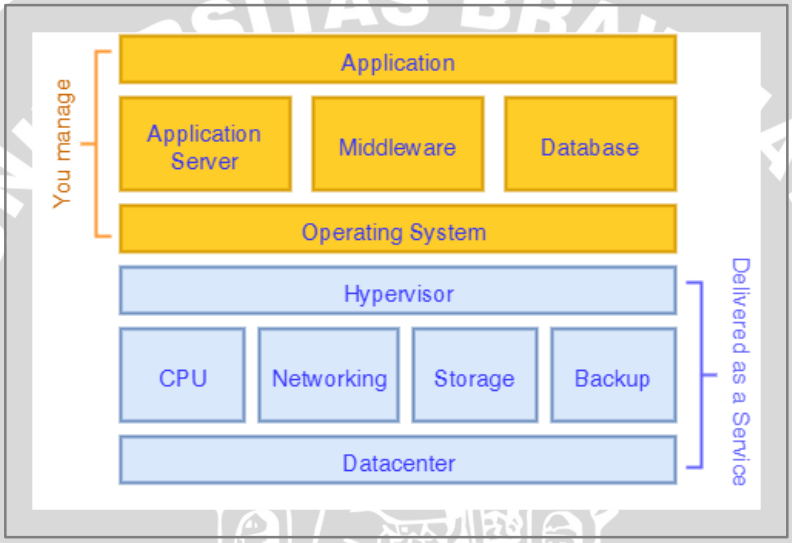


Gambar 2.2 Pemodelan PaaS

(Sumber : Adrianus, 2010)

- ***Infrastructure as a Service* (IaaS)**

IaaS adalah model layanan *cloud computing* berupa sumber daya infrastruktur (media penyimpanan, server, network, memory, dll). Layanan ini dapat dianalogikan seperti pengguna yang menyewa komputer kosong. Pengguna dapat melakukan konfigurasi, menginstal sistem operasi apa saja sesuai kebutuhan. Namun pengguna tidak bisa mengelola atau mengendalikan infrastruktur *cloud* yang menjadi dasarnya. Hanya saja pengguna mempunyai hak akses untuk mengontrol sistem operasi, media penyimpanan, aplikasi, dan beberapa komponen jaringan (contoh: *firewall*). Contoh penyedia layanan IaaS diantaranya adalah: Rackspace Cloud dan Amazon EC2. Berikut gambaran mengenai pemodelan layanan IaaS.



Gambar 2.3 Pemodelan IaaS

(Sumber: Adrianus, 2010)

2.2.1 Cloud storage

Cloud storage merupakan bagian dari teknologi *cloud computing* yang menawarkan layanan media penyimpanan pada lingkungan *cloud*. Konsep *cloud storage* sebenarnya mirip dengan konsep *file server* pada suatu institusi, hanya saja infrastruktur media penyimpanannya disediakan dan dikelola oleh peyedia layanan *cloud storage* itu sendiri, namun pemanfaatannya dijadikan sebagai layanan penyimpanan berkas yang dapat diakses di mana saja kapan saja selama terhubung dengan internet. *Cloud storage* pada dasarnya bekerja melalui virtualisasi *data center*. Secara umum layanan ini beroperasi melalui API berbasis web yang dikendalikan dari aplikasi klien. Sebenarnya *cloud storage* juga bukanlah layanan baru dalam dunia teknologi, dulunya layanan *cloud storage* lebih dikenal sebagai layanan *virtual drive*, akan tetapi ketika istilah tersebut berubah ketika layanan *cloud computing* menjadi isu hangat di dunia teknologi (Assagaf, 2012).

Terdapat beberapa jenis *cloud storage* yang umum digunakan, yaitu (Beal, n.d.):

1. ***Personal cloud storage***
   Layanan inilah yang sering dikenal sebagai *mobile cloud storage.* Media penyimpanan *cloud* pribadi ini sebenarnya merupakan bagian dari media penyimpanan *cloud* publik. Hanya saja penyimpanan ini digunakan untuk menyimpan data individu. Sehingga tidak jarang layanan ini banyak dimanfaatkan oleh perusahaan seperti: Yahoo, Google & Apple sebagai layanan penyimpanan data pribadi yang dapat disinkronisasi dengan layanan email ataupun dengan perangkat lain.
2. ***Public cloud storage***
   Media penyimpanan *cloud* publik adalah di mana perusahaan dan penyedia layanan terpisah, tidak ada sumber daya yang disimpan di pusat data perusahaan. Sehingga penyedia layanan *cloud* sepenuhnya mengelola penyimpanan *cloud* milik perusahaan.
3. ***Private cloud storage***
   Layanan ini hampir sama dengan media penyimpanan *cloud* publik, hanya saja media penyimpanan *cloud* perusahaan dan penyedia layanan terintegrasi dalam pusat data perusahaan.
4. ***Hybrid cloud storage***
   Penyimpanan *cloud* hybrid merupakan gabungan dari penyimpanan *cloud* publik dan swasta. Di mana beberapa data berapa pada penyimpanan *cloud* perusahaan, sementara data yang lain disimpan dan dapat diakses dari penyedia layanan penyimpanan publik.

Diantara keuntungan memanfaatkan layanan penyimpanan *cloud* adalah aksesibilitas tinggi dan keandalan yang lebih besar, perlindungan terhadap data (backup) yang lebih kuat dan pemulihan jika terjadi bencana dapat ditangani lebih cepat. Selain itu jika dilihat dari segi ekonomis, akan didapatkan biaya penyimpanan yang lebih rendah secara keseluruhan karena tidak perlu membeli, mengelola dan memlihara perangkat keras yang pada umumnya memiliki harga mahal. Sayangnya, kelemahan terbesar untuk penyimpanan *cloud* adalah bahwa pengguna dibatasi dengan bandwidth. Jika koneksi internet yang dimiliki tidak stabil/lambat, kemungkinan besar pengguna akan kesulitan untuk mengakses ataupun berbagi berkas.

2.2.1.1 Google Drive

Google Drive adalah layanan *cloud storage* dari Google yang diperkenalkan pada akhir April 2012, yaitu sebuah layanan gratis yang memungkinkan Anda untuk menyimpan berkas secara online dan mengaksesnya di mana saja menggunakan *cloud* (internet). Anda dapat mengakses menggunakan *personal computer* (PC), laptop dan bahkan *smartphone.* Setiap perubahan yang Anda lakukan pada berkas yang tersimpan pada Google Drive (dari satu gadget), maka secara otomatis akan muncul jika anda mengakses pada tempat lain. Bahkan

dengan layanan ini dapat dikatakan Anda tidak perlu menginstall program-program pada PC Anda, karena Google Drive sudah dapat menangani lebih dari 30 jenis berkas, termasuk Adobe Photoshop dan Adobe Illustrator serta video berkualitas *High Definition* (HD). Selain itu, jika Anda ingin berbagi berkas pada orang lain cukup mengirimkan *link file* tersebut atau Anda juga dapat berkolaborasi dengan orang lain pada sebuah folder (Horn, 2012).

Selain itu, *platform* Google Drive juga memberikan fasilitas *Application Programming Interface* (API) dan *library* bagi penggunanya untuk membantu mereka dalam mengembangkan aplikasi yang terintegrasi dengan layanan ini. Inti fungsi aplikasi Drive adalah untuk men-*download* meng-*upload* berkas pada Google Drive. Namun, *platform* Drive menyediakan lebih banyak fungsi dari sekedar penyimpanan. Dengan *platform* Drive, Anda akan menggunakan model yang didasarkan pada ID berkas -bukan hirarki folder tradisional- ketika berhubungan dengan berkas dan folder (Google, 2015).

2.2.1.2 Dropbox

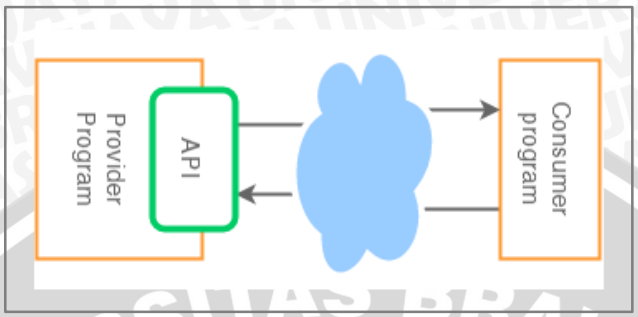
Dropbox merupakan salah satu penyedia layanan *cloud storage* yang cukup populer dan termasuk yang pertama dalam bidang ini. Dropbox bisa dikatakan sebuah layanan *cloud storage* pribadi yang sering digunakan untuk *file sharing* dan kolaborasi. Layanan ini menyediakan 2 *gigabyte* (GB) media penyimpanan gratis untuk free member hingga 100 GB untuk premium member. Dropbox tidak hanya mengakomodasi per orangan, tetapi juga terdapat layanan yang dapat digunakan untuk satu tim. Data pengguna disimpan pada Amazon *Simple Storage Service* (S3) dan dilindungi dengan *Secure Socket Layer* (SSL) dan *Advanced Encryption Standard* (AES) dengan enkripsi 256-bit. Untuk berbagi berkas, pengguna dapat mengirimkan link dari *website* Dropbox sehingga orang lain dapat melihatnya, atau mengirimkan sebuah undangan untuk bergabung pada sebuah *shared folder* (Rouse, 2011).

Dropox juga menyediakan API bagi para pengembang aplikasi di Dropbox *Platform*. Dropbox Core API menyediakan cara yang fleksibel untuk membaca dan menulis berkas di Dropbox. Termasuk adanya dukungan untuk fungsi pencarian, revisi, dan *recover file.*

2.3 API (Application Programming Interface)

Application Programming Interface atau lebih dikenal dengan API adalah sebuah *software interface* yang berbentuk *library.* Pada dasarnya API dibangun dari kumpulan instruksi yang kemudian digabungkan menjadi satu paket. Paket inilah yang dimanfaatkan pengembang aplikasi untuk menghubungkan aplikasinya pada penyedia aplikasi (misal: *cloud storage*). Sehingga pengguna tetap bisa menjalankan aksi pada aplikasi *server* walaupun tidak berinteraksi langsung dengan aplikasi *server* (Binus, 2013). Selain itu adanya API merupakan salah satu jalan keluar bagi pengembang aplikasi yang ingin menggunakan sebuah layanan, karena pada umumnya sebuah layanan memiliki hak akses yang terbatas.

Sebagai contoh, facebook.com menyediakan API pada layanannya. Sehingga pengembang aplikasi web maupun mobile lebih mudah mengakses informasi yang ada pada layanan tersebut. Dengan menggunakan facebookAPI, sebuah aplikasi pihak ketiga dapat meng-*upload* foto, berbagi *link*, mengubah status dll seperti ketika menggunakan facebook secara langsung.



Gambar 2.4 Skema konektivitas API antar program

(Sumber: Binus, 2013)

Sebenarnya API adalah antarmuka antara perangkat lunak dengan perangkat lunak yang lain *(software-to-software),* bukan *user interface.* API bekerja dibalik layar aplikasi tanpa sepengetahuan pengguna. Misalkan ketika Anda membeli tiket pesawat secara online pada sebuah travel agen (pihak ketiga), setelah Anda memasukkan data yang diperlukan, situs tersebut akan mengirimkannya menggunakan API maskapai tersebut yang kemudian mengirimkan respon kembali ke situs travel agen. Sebagai pengguna, Anda hanya akan melihat antarmuka dari situs travel agen saja, akan tetapi di belakang itu semua terjadi komunikasi antara situs travel agen dengan aplikasi maskapai. Inilah yang disebut integrasi *seamless,* karena pengguna tidak pernah mengetahui ketika fungsi pada satu aplikasi dikirimkan pada aplikasi yang lain (Prasetyo, 2014).

API dapat diklasifikasikan menjadi beberapa kategori, hal ini ditentukan dari abstraksi yang dideskripsikan di dalam sistem. Berikut pada Tabel 2.1 adalah kategori API berdasarkan abstraksi (Binus, 2013):

Tabel 2.1 Kategori API

Kategori API	Deskripsi	Contoh
Operating system	API pada komputer yang berfungsi untuk melakukan perintah-perintah dasar. Seperti proses input/output dan proses eksekusi program.	WinAPI, ShellAPI
Programming languages	API ini berfungsi untuk meningkatkan kapabilitas dalam pengembangan menggunakan bahasa pemrograman	Java API, Python/C API,

Tabel 2.1 Kategori API (lanjutan)

Kategori API	Deskripsi	Contoh
Application services	API yang digunakan ketika ingin megintegrasikan aplikasi Anda dengan sebuah layanan	MySAP API
Infrastructure services	API yang dimanfaatkan untuk mengakses infrastruktur dari suatu komputer, meliputi aplikasi, media penyimpanan bahkan *peripheral*	Amazon EC2 (*Elastic Compute Cloud*), Amazon S3 (*Simple Storage Service*)
Web services	API yang bekerja pada suatu *web application.* API inilah yang seringkali dimanfaatkan.	Dropbox API & Google Drive API, Amazon API, Facebook API

2.4 Java

Java merupakan bahasa pemrograman yang berorientasi objek (OOP) dan dapat dijalankan pada berbagai *platform* sistem operasi karena bahasa pemrograman ini tidak berinteraksi secara langsung dengan *CPU* (*Central Processing Unit*), melainkan dapat berdiri sendiri. Perkembangan Java tidak hanya terfokus pada satu jenis komputer (*desktop*/laptop) dan sistem operasi, melainkan secara teori Java dapat berjalan dengan baik pada mikro & mini komputer dengan berbagai sistem operasi yang menggunakannya (Avestro, 2007).

Sebagai sebuah bahasa pemrograman yang berkembang pesat, Java tidak murni dibuat secara *from scratch,* melainkan banyak mengakomodasi dari bahasa pemrograman yang telah ada sebelumnya. Seperti bahasa SIMULA yang diadopsi oleh Java pada bentuk-bentuk dasar dari pemrograman berorientasi obyek, Objective C diadopsi pada fasilitas *interface* dan masih banyak lagi bahasa pemrograman yang diadopsi. Selain itu, SUN sebagai perusahan pengembang resmi juga menyediakan berbagai macam tools bagi penggunanya, yaitu: *compiler, interpreter,* penyusun dokumentasi, paket kelas dan sebagainya.

Kelebihan dari Java sendiri adalah hampir dapat digunakan untuk mengembangkan seluruh bentuk aplikasi, mulai dari desktop, web dan bahkan *socket programming*, sebagaimana dibuat dengan menggunakan bahasa pemrograman konvensional lainnya. Tidak hanya itu, karena sifatnya yang *open source,* banyak tersedia *plugin, library* dan *framework* yang bertujuan untuk memudahkan pengguna dalam membangun sebuah aplikasi berbasis Java.

2.4.1 Maven *project*

Maven adalah sebuah perangkat lunak yang komprehensif untuk memanejemen proyek. Berdasarkan *Project Object Model* (POM). Maven mendefinisikan bagaimana berkas .java Anda dapat dikompilasi menjadi berkas .class, dikemas menjadi berkas .jar, mengelola *classpath*, dam segala macam tugas yang dibutuhkan untuk membangun proyek berbasis Java. Perangkat lunak ini

menyederhanakan proses dalam membangun seperti halnya ANT dan GRADLE tetapi Maven lebih unggul karena Maven benar-benar tidak membutuhkan *tool* tambahan atau *script* yang memberikan perintah untuk men-*download* & menginstal *library* yang diperlukan (Anon., 2015). Tujuan utama Maven adalah untuk memungkinkan pengembang aplikasi memahami keadaan lengkap dari upaya pembangunan dalam periode waktu terpendek.

Maven juga menyediakan banyak cara bagi pengembang untuk mengelola fungsi berikut:

- *Builds*
- *Documentation*
- *Reporting*
- *Dependencies*
- *SCMs*

Beberapa kelebihan Maven:

- **Portabilitas**
  Maven dirancang untuk memudahkan akses ketika berpindah pada komputer yang lain dengan *script* & *buildscript* yang sama.
- **Konfigurasi**
  Mudah dalam berpindah proyek tanpa membangun proses baru
- **Modularisasi Proyek**
  Proyek dibagi dalam subkompenen yang lebih kecil, sehingga memudahkan ketika dilakukan debug dan mengelola struktur proyek secara keseluruhan.
- **Manajamen Ketergantungan**
  Ketergantungan berkas dapat dengan mudah diatasi dengan berkas pom.xml.
- **Siklus Hidup Proyek**
  Konfigurasi SCM yang baik dan dengan repositori internal, manajemen ketergantungan jadi lebih mudah.

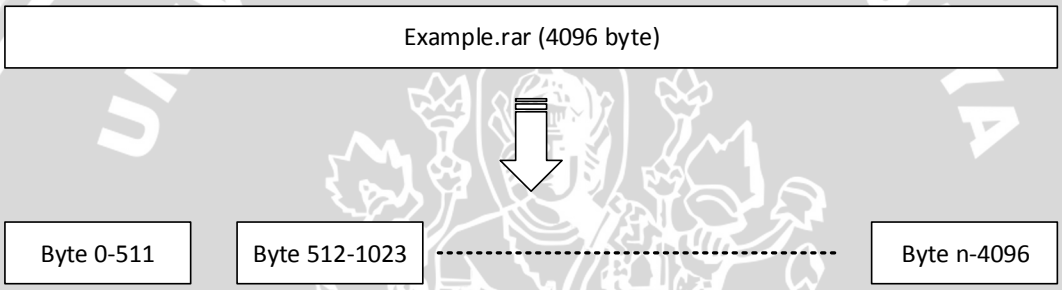
2.5 Splitting & merging

Pada setiap sistem operasi, sebutan dari karakteristik operasional atau struktural berkas disebut jenis berkas. Jenis berkas mengidentifikasi program, seperti Microsoft Word, yang digunakan untuk membuka berkas. Jenis berkas berhubungan dengan ekstensi berkas. Sebagai contoh, berkas yang memiliki ekstensi .txt atau ekstensi .log adalah dari jenis *Text Document* dan dapat dibuka menggunakan editor teks apapun.

Untuk menangani semua jenis berkas di sistem apapun, Anda harus bekerja pada konsep dasar dari sebuah komputer dimana berkas dibangun, yang

merupakan aliran byte, karena setiap berkas terlepas dari jenisnya, adalah satu set byte yang digabungkan bersama-sama membentuk format khusus dalam sistem.

Split & merge sebenarnya adalah operasi sederhana yang tidak perlu sebuah algoritma khusus untuk melakukannya. Pada *file splitter*, cukup lakukan operasi sederhana membagi berkas dengan mengacu pada konsep berkas (kumpulan byte). Sedangkan untuk *file merge*, lakukan kebalikannya. Ada banyak manfaat dari menggunakan *file splitter*, tetapi yang paling penting adalah untuk memecah berkas besar menjadi potongan-potongan yang lebih kecil sehingga potongan ini dapat dengan mudah dimasukkan pada media removable yang memiliki ukuran terbatas serta juga memudahkan ketika proses *upload* ke Internet (Rackham, 2013). Sedangkan *file merger*, berfungsi menggabungkan dari berkas yang sebelumnya telah dipecah, menjadi satu berkas utuh yang dapat dibaca. Karena setelah berkas dipecah dan menjadi potong-potongan berkas, secara otomatis berkas tersebut tidak dapat dibaca seperti berkas aslinya. Untuk itu diperlukan cara untuk menyatukan potongan-potongan berkas.



Gambar 2.5 Proses *splitting*

2.6 Fungsi hash

Secara bahasa hash berarti daging cincang yang dipanggang. Istilah hash dipakai karena pada algortima ini proses pengolahan datanya mirip dengan 'cincang dan olah'. Fungsi hash atau algoritma hash adalah sebuah metode yang menghasilkan suatu nilai hash dari sebuah data. Nilai hash (atau hanya hash) juga sering disebut *message digest* adalah gabungan angka dan huruf yang dihasilkan dari serangkaian teks. Nilai hash dari suatu fungsi hash memiliki panjang yang tetap walaupun masukan data berbeda-beda, yaitu 256-bit (32 *byte*) untuk algoritma SHA-256 (lihat Tabel 2.2) (Dwiperdana, 2007). Nilai hash secara substansial berukuran lebih kecil dari dari data itu sendiri, dan itu dihasilkan dari formula yang sedemikian rupa bahwa sangat tidak mungkin jika data satu dengan data lain menghasilkan nilai hash yang sama walaupun data tersebut memiliki kemiripan 99%.

Tabel 2.2 Perbandingan panjang nilai hash dengan algoritma SHA-256

Message	Nilai hash (SHA-256)
Fungsi hash	9C40BC06DDE34DFEBCD8CFEDFAA80E59309A0A67E71F2302B26A99DBA2225B90
Sistem Pengamanan Data Pada Media Penyimpanan *Cloud* Dengan Metode *Split Data Distribution*	12A9AF5C8C379C55DFC074EEDE65B78C57F0708C6AA6B463B0C2630FDAD5C2C2

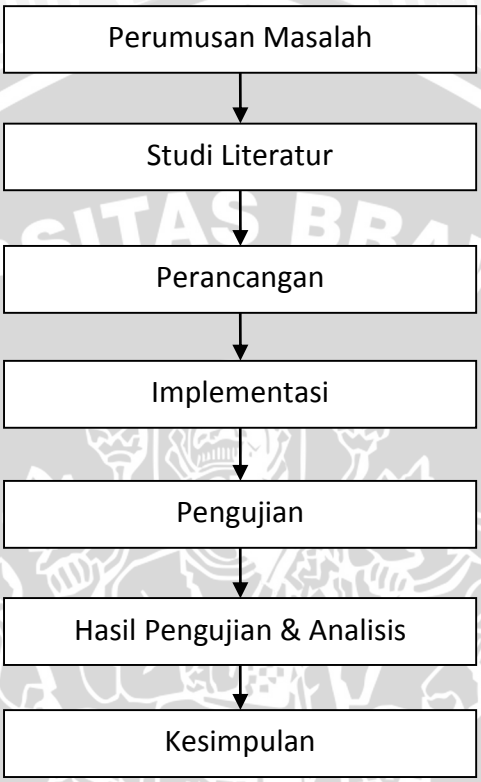
Fungsi hash sangatlah umum, bahkan sangat perlu digunakan dalam bidang keamanan dan enkripsi data untuk memastikan bahwa tidak terjadi perubahan pada data yang ditransmisikan. Proses tersebut dilakukan dengan membandingkan nilai hash dari data (pengirim) dengan nilai hash dari data (penerima). Perlu diketahui juga bahwa hash bukanlah sebuah fungsi 'satu-ke-satu', melainkan sebuah fungsi 'banyak-ke-satu'. Yang dimaksud fungsi 'banyak-ke-satu' adalah, pada algoritma ini masukan yang diterima tidak dibatasi, akan tetapi hasi nilai yang dikeluarkan sesuai dengan batasan tiap algoritma. Di mana pada fungsi 'banyak-ke-satu' hasil yang didapatkan tidak dapat dikembalikan pada nilai semula. Tetapi dengan membatasi nilai masukan, dapat dibuat sebuah fungsi hash 'satu-ke-satu' yang menjadikan data dapat dikembalikan pada keadaan semula. Pada fungsi hash yang merupakan fungsi 'banyak-ke-satu' masih dimungkinkan terjadi bentrokan pada nilai, yaitu dari data yang berbeda dihasilkan nilai yang sama. Akan tetapi peluang untuk terjadi bentrokan sangatlah kecil dan hanya terjadi pada MD5 & SHA-0.

2.6.1 SHA-256 (*Secure Hash Algorithm-256*)

SHA-256, merupakan salah satu fungsi hash yang masih umum digunakan. Fungsi merupakan salah satu varian dari SHA-2/SHA2, yang dibuat karena pada SHA-1 masih ditemukan setidaknya <80 bentrokan. SHA-1 sendiri sebenarnya peningkatan dari SHA-0. Dengan panjang nilai hash 256-bit (32 *byte*), fungsi ini memiliki 2^{256} atau sekitar 1,16x x1077 kemungkinan keluaran. Sebenarnya SHA-256 saat ini juga dimungkinkan terjadi bentrokan, akan tetapi kemungkinan yang dihasilkan lebih kecil dan akan lebih banyak usaha yang dilakukan. Sehingga SHA-2 sampai sekarang masih diimplementasikan dalam beberapa aplikasi keamanan, dan beberapa protokol termasuk SSL, TLS, IPSec, SSH dll. Sedangkan SHA-256 digunakan secara khusus sebagai bagian dari proses otentikasi perangkat lunak Debian GNU/Linux dan digunakan sebagai standar penandatanganan pesan pada DKIM.

BAB 3 METODOLOGI

Dalam bab ini akan dibahas metodologi yang digunakan dalam penelitian Aplikasi Mekanisme Pengamanan Data Pada Media Penyimpanan *Cloud* Dengan Metode *Split Data Distribution*. Langkah-langkah yang akan dilakukan dalam pada penelitian ini adalah sebagai berikut:



Gambar 3.1 Langkah-langkah penelitian

3.1 Perumusan masalah

Permasalahan yang dihadapi dalam penelitian ini adalah bagaimana menerapkan sebuah sistem yang dapat mengamankan data pengguna *cloud* terlepas dari pengamanan data yang dilakukan oleh penyedia layanan *cloud storage*. Seperti yang telah dibahas pada bab-bab sebelumnya, dibalik banyaknya kelebihan yang didapat dari penggunaan layanan *cloud storage* terdapat beberapa kekurangan. Terutama dari segi keamanan seperti hilangnya privasi data, pencurian informasi dan sebagainya. Salah satu cara yang dapat digunakan untuk mengatasi permasalahan ini adalah dengan membuat sebuah aplikasi yang mampu mengakomodasi keamanan data pengguna. Dalam mengakomodasi penggunanya, aplikasi ini menerapkan cara untuk memecah berkas menjadi beberapa bagian sehingga kerahasiaan berkas bisa lebih terjaga.

Fitur memecah berkas menjadi *chunk* sebenarnya sudah disediakan oleh beberapa penyedia layanan *cloud* sebagai layanan tambahan. Akan tetapi hal tersebut akan sama saja jika dilakukan pada *cloud storage* itu sendiri karena proses

pemecahan dilakukan pada sisi server (*cloud storage* itu sendiri). Oleh karena itu, penulis mengemukakan ide untuk melakukan *split & merge* yang dilakukan sebelum berkas di-*upload* pada layanan *cloud*, dan potongan berkas akan didistribusikan tidak hanya pada satu layanan *cloud*. Melainkan beberapa pada beberapa layanan *cloud* yang berbeda.

3.2 Studi literatur

Tahap studi literatur dilakukan untuk mengkaji dasar teori yang berkaitan dengan penelitian ini. Untuk itu dikaji dasar teori yang berkaitan dengan Bahasa Pemrograman Java, Oauth 2.0, *split & merge file,* serta Dropbox dan Drive API. Karena dasar teori yang didapat akan sangat menunjang penulisan serta pengerjaan penelitian ini. Selain itu juga dikaji tentang hal-hal yang mendukung pengujian perangkat lunak seperti *functional testing* dan *performance testing*.

3.3 Perancangan

Pada penelitian ini, proses perancangan sistem dibagi menjadi dua bagian, yaitu tahap pertama dilakukan analisis kebutuhan sistem, kemudian tahap selanjutnya dilakukan perancangan aplikasi sistem. Analisis kebutuhan sistem ini biasanya yang akan menunjang fungsi utama sistem, diantaranya adalah bagaimana proses otentikasi pengguna pada layanan *cloud* dan bagaimana cara mengetahui berkas apa saja yang telah di-*upload* dari tiap pengguna serta membuat daftar kebutuhan dari perangkat lunak dan perangkat keras yang digunakan pada penelitian ini. Sedangkan pada tahap kedua terdiri dari perancangan aplikasi sistem secara umum, dijelaskan juga proses/alur kerja sistem. Bagaimana sistem melakukan *splitting file* lalu mendistribusikan pada beberapa layanan *cloud* dan menyimpan metadata berkas pada sisi klien, serta bagaimana untuk melakukan hal sebaliknya. Melakukan *merging file* tanpa adanya perubahan pada berkas.

3.3.1 Lingkungan sistem

Setelah studi literatur dilakukan, perlu dilakukan analisis kebutuhan fungsional dari sistem. Kebutuhan tersebut diantaranya:

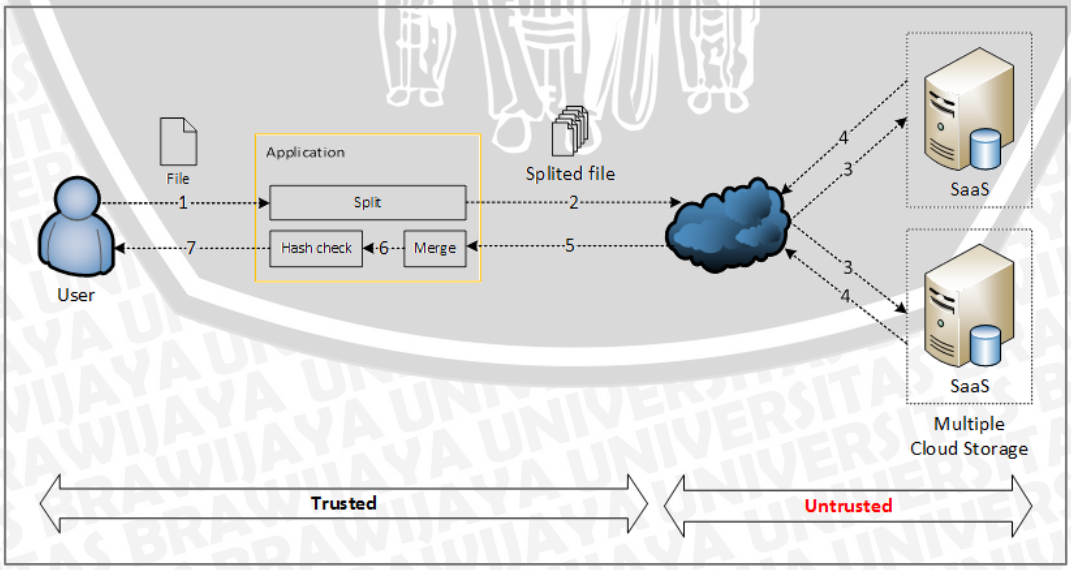
1. Diperlukan fungsi otentikasi pada sistem untuk menjembatani *user* melakukan otentikasi pada layanan *cloud*.
2. Diperlukan fungsi yang mampu secara otomatis memecah dan menggabungkan berkas yang akan di-*upload* oleh *user*.
3. Diperlukan fungsi yang mampu melakukan *upload & download file* pada layanan *cloud*.
4. Diperlukan fungsi yang mampu menyimpan metadata & kode hash dari berkas yang telah di-*upload* pada layanan *cloud*.
5. Diperlukan fungsi yang mampu secara otomatis membandingkan/mengecek keaslian berkas yang telah di-*download.*

Dari beberapa kebutuhan fungsional sistem yang telah didapat, diperlukan juga beberapa komponen non-fungsional yang dapat menunjang pengerjaan sistem ini, yaitu:

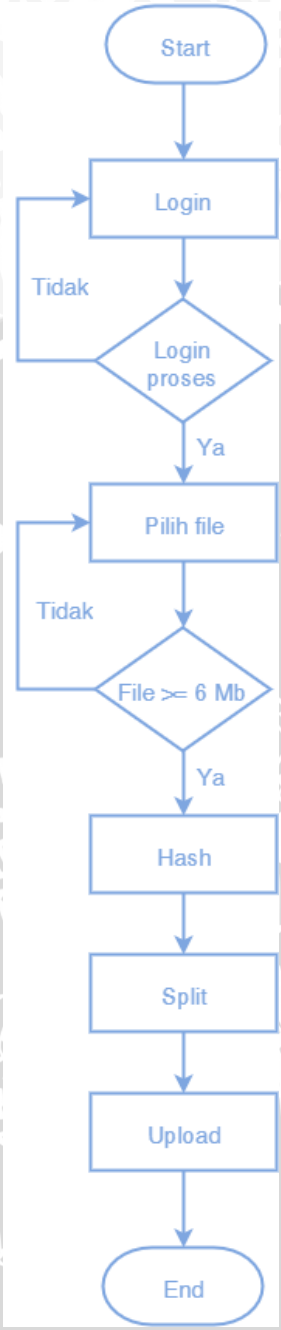
1. Kebutuhan perangkat keras (*hardware*)
    a. 1 unit PC/laptop sebagai klien
    b. Jaringan internet
2. Kebutuhan perangkat lunak (*software*)
    a. Java 8.0
    b. NetBeans IDE 8.0.2
    c. jdk1.8.0_60
    d. dropbox-java-sdk-1.7.7
    e. google-api-services-drive-v2-rev130-1.18.0-rc
3. Kebutuhan fitur
    a. *Split & merge*
    b. *Upload & download*
    c. Melihat berkas yang telah di-*upload*
    d. Hash

3.3.2 Perancangan aplikasi

Proses perancangan pada tahap ini dilakukan agar sistem yang akan dibuat berjalan secara sistematis. Untuk itu perlu dijelaskan sedikit mengenai arsitektur sistem yang akan dibuat (lihat Gambar 3.2). Sistem ini nantinya akan bekerja pada sisi pengguna atau dapat dikatakan masih berada pada jaringan yang dapat dipercaya (*trusted*). Di mana sistem di sini hanya berperan sebagai pihak kedua/jembatan antara penyedia layanan dengan pengguna itu sendiri. Karena itu pada sistem ini hanya terdapat satu *privilege*, yaitu pengguna. Maka perancangan aplikasi didasarkan pada dua fungsi utama, yaitu: fungsi *upload* & fungsi *download*.



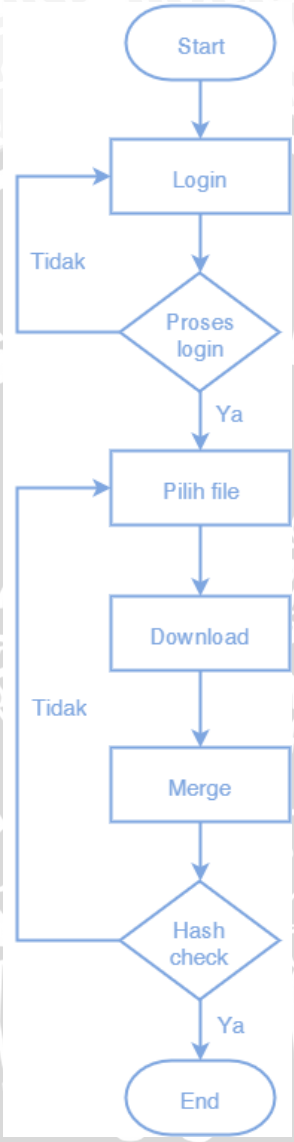
Gambar 3.2 Arsitektur sistem



Gambar 3.3 Flowchart proses *upload*

Alur kerja pada Gambar 3.3 menjelaskan bagaimana proses *upload* pada sistem. Ketika awal program dijalankan, pengguna harus melakukan *login*/otentikasi pada masing-masing layanan *cloud*. Proses otentikasi sebenarnya tidak dilakukan pada aplikasi, melainkan pada website penyedia layanan *cloud* itu sendiri. Aplikasi hanya menjembatani bagaimana pengguna dapat tetap terhubung dengan layanan. Jika otentikasi berhasil, fungsi-fungsi utama aplikasi akan dapat dijalankan. Untuk melakukan *upload*, pengguna harus terlebih dahulu memilih berkas yang akan di-*upload.* Di sini jenis berkas yang di-*upload* tidak dibatasi, akan tetapi besar ukuran berkas dibatasi minimal 6 Mb untuk memudakan dalam pembagian ukuran berkas. Setelah berkas dipilih, sistem akan

secara otomatis menjalankan fungsi hash. Kemudian berkas akan dipecah menjadi beberapa bagian, jumlah bagian ini ditentukan oleh sistem. Bagian-bagian berkas yang terbentuk lalu di-*upload* pada dua *cloud storage* yang berbeda. Jika proses *upload* berhasil, metada berkas akan disimpan pada sebuah berkas di sisi klien.



Gambar 3.4 Flowchart proses *download*

Gambar 3.5 merupakan alur kerja sistem saat proses *download.* Proses ini secara umum hampir sama dengan proses *upload* yang dijelaskan sebelumnya. Pada awalnya pengguna diharuskan untuk melakukan otentikasi pada setiap layanan *cloud* untuk dapat memanajemen berkas (*download*). Proses otentikasi dilakukan pada *web-server* masing-masing layanan. Jika otentikasi berhasil, aplikasi akan menampilkan *list* dari berkas apa saja yang sebelumnya telah di-*upload* oleh pengguna. Sedangkan jika proses otentikasi gagal, pengguna diarahkan untuk mengulangi proses otentikasi. Sebelum melakukan *download*, pengguna harus memilih salah satu dari berkas yang ditampilkan dan lanjutkan pada proses *download*. Pada proses *download* ataupun *upload,* berkas tidak

diproses secara bersamaan, melainkan bergantian dari satu layanan *cloud* ke layanan *cloud* yang lain. Setelah berkas dari semua layanan berhasil di-*download,* aplikasi akan secara otomatis melakukan *merge* dan pengecekan hash. Pengecekan hash dilakukan dengan membandingkan nilai hash dari berkas yang baru di-*download* dengan berkas hash dari berkas yang sama sebelum di-*upload,* agar terjamin keaslian data ketika sampai pada klien.

3.4 Implementasi

Implementasi akan dilakukan sesuai dengan perancangan sistem yang telah dibuat sebelumnya. Pada bagian ini dibagi menjadi beberapa tahap implementasi, diantaranya:

a. Implementasi perangkat keras (*hardware*). Implementasi ini hanya memerlukan konfigurasi untuk koneksi internet.

b. Implementasi perangkat lunak (*software*). Diperlukan beberapa perangkat lunak yang dapat menunjang jalannya sistem. Perangkat lunak tersebut meliputi: Java, sebagai basis untuk mengembangkan sistem, NetBeans IDE sebagai *compiler progam*, JDK (Java Development Kit) sebagai alat untuk pengembangan sistem, serta *google-api-services-drive-v2-rev187-java-1.20.0 & dropbox-java-sdk-1.7.7* sebagai penghubung aplikasi dengan layanan *cloud*.

Pada saat melakukan implementasi diharapkan sistem dapat berjalan sesuai rancangan. Sehingga dari sistem tersebut bisa didapatkan sebuah aplikasi yang mampu memberikan layanan *split & merge file* serta *upload & download file* pada *multiple cloud storage.*

3.5 Pengujian sistem

Pegujian sistem ini dilakukan untuk memastikan bahwa kinerja sistem bersesuaian dengan spesifikasi kebutuhan yang telah dirancang sebelumnya. Pengujian juga dapat dilakukan untuk menguji otomasi sistem apakah mampu memenuhi kebutuhan-kebutuhan yang telah ditentukan. Pengujian dilakukan dengan menjalankan fungsi sistem dan membandingkannya dengan kebutuhan fungsional yang ada.

3.6 Analisis dan hasil pengujian

Untuk mengetahui kinerja dari sistem secara keseluruhan, dilakukan analisa dari hasil pengujian yang didapat. Dari analisa hasil akan diketahui juga kelayakan dari sistem yang dibuat. Analisa hasil juga diperlukan untuk menarik kesimpulan dari penelitian yang telah dilakukan.

3.7 Kesimpulan

Tahap ini merupakan tahapan terakhir dalam penelitian dimana ditarik kesimpulan dari hasil analisa yang didapat dari sistem. Kesimpulan dituliskan sebagai rangkuman dari proses pengerjaan penelitian dan saran dituliskan untuk

pengembangan sistem kedepannya ataupun untuk memperbaiki kesalahan-kesalahan yang ada pada sistem saat ini.

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Pada bab ini dijelaskan mengenai tahap-tahap dalam pembangunan sistem distribusi berkas yang memanfaatkan beberapa penyedia media penyimpanan *cloud*, serta dilakukan pengujian terhadap sistem yang telah dibangun. Pada tahap implementasi, langkah-langkah yang dilakukan adalah pembangunan aplikasi berdasarkan kebutuhan dan rancangan yang telah ditentukan.

4.1 Implementasi

Dalam penerapan sistem yang akan dibuat, secara umum terdapat empat proses yang akan dilakukan, yaitu:

1. Konfigurasi API
2. Pengembangan antarmuka
3. Pengembangan aplikasi (*coding*)

4.1.1 Konfigurasi API

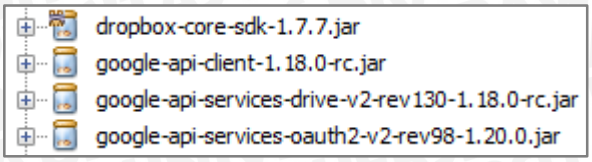
4.1.1.1 Penambahan API library

Sebelum membuat aplikasi, perlu dilakukan penambahan API *library* sesuai bahasa pemrograman yang digunakan. Pada aplikasi ini, layanan *cloud storage* yang digunakan menyediakan API versi 1.7.7 untuk Dropbox dan versi 2.0 rev. 130 pada Google Drive. Untuk menambahkan *library* pada aplikasi ini dapat dilakukan dengan dua cara, pertama menggunakan berkas *pom.xml* yang berisi seluruh *dependencies library* yang diperlukan. Sedangkan cara kedua adalah dengan mengunduh berkas *library* dan menambahkan secara manual menggunakan *command line.*

```
1  . . . . .
2  <dependency>
3       <groupId>com.dropbox.core</groupId>
4       <artifactId>dropbox-core-sdk</artifactId>
5       <version>[1.7,1.8)</version>
6  </dependency>
7  <dependency>
8       <groupId>com.google.apis</groupId>
9       <artifactId>google-api-services-drive</artifactId>
10      <version>v2-rev130-1.18.0-rc</version>
11      <type>jar</type>
12 </dependency>
13 . . . . .
```

***Source code* 4.1 Kode untuk menambahkan API**

Source code 4.1 merupakan potongan kode yang ditambahkan pada berkas *pom.xml.* Perintah diatas berfungsi untuk menambahkan API atau *library.* Dengan perintah tersebut, library-library yang diperlukan akan secara otomatis ditambahkan pada proyek pengerjaan ketika program dijalankan.



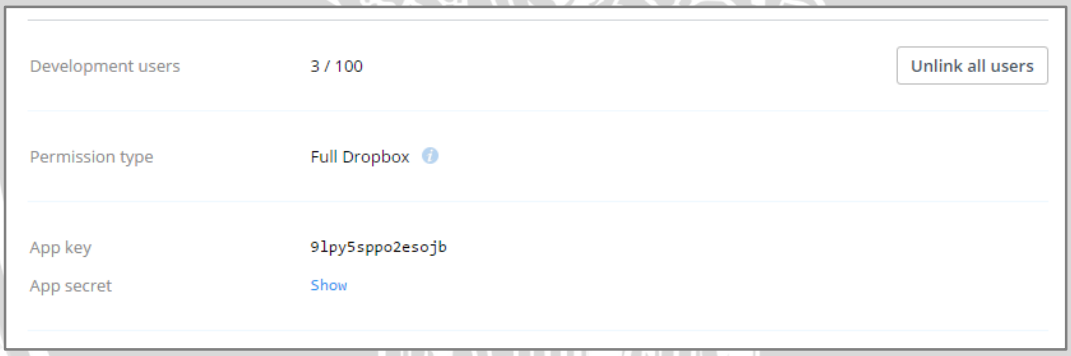
Gambar 4.1 Library utama (Dropbox & Google Drive)

Pada Gambar 4.1 ditampilkan *library* utama dan beberapa *library* pendukung. Semua library yang telah ditambahkan pada *project* pengerjaan akan ditampilkan pada folder *dependencies* pada *project*.

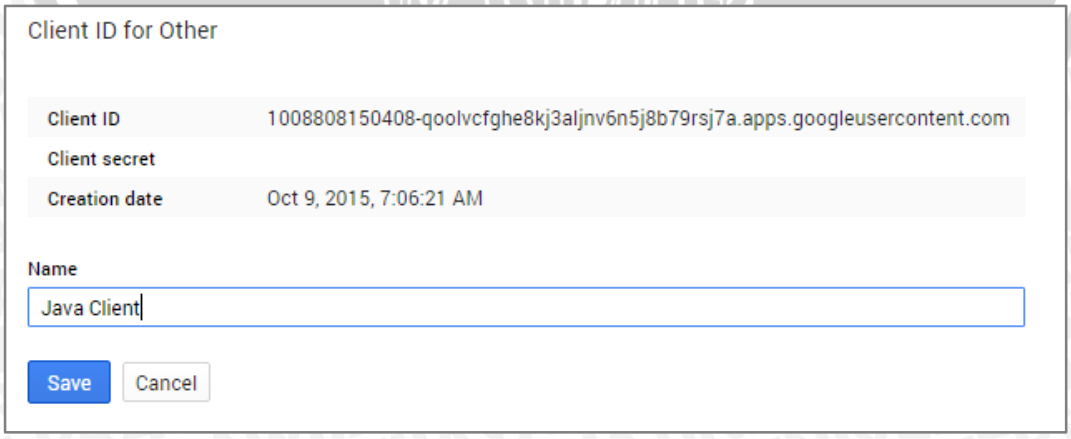
4.1.1.2 Registrasi API

Perlu diketahui, sebelum Anda dapat dengan leluasa menggunakan API suatu layanan, Anda perlu mendaftarkan aplikasi yang akan dibangun. Langkah ini perlu dilakukan untuk memudahkan penyedia layanan dalam memberikan akses pada pengembang aplikasi. Karena setiap aplikasi yang akan mengakses suatu layanan juga perlu melakukan otentikasi. Selain itu juga untuk menghindari hal-hal yang tidak diinginkan pada layanan yang diberikan.

Jika telah melakukan proses registrasi pada layanan *cloud,* maka setiap aplikasi yang terdaftar akan mendapatkan sebuah *Client ID & Client Secret.* Di mana *Client ID & Client Secret* inilah yang digunakan dalam proses otentikasi aplikasi.



Gambar 4.2 Client ID & client secret Dropbox



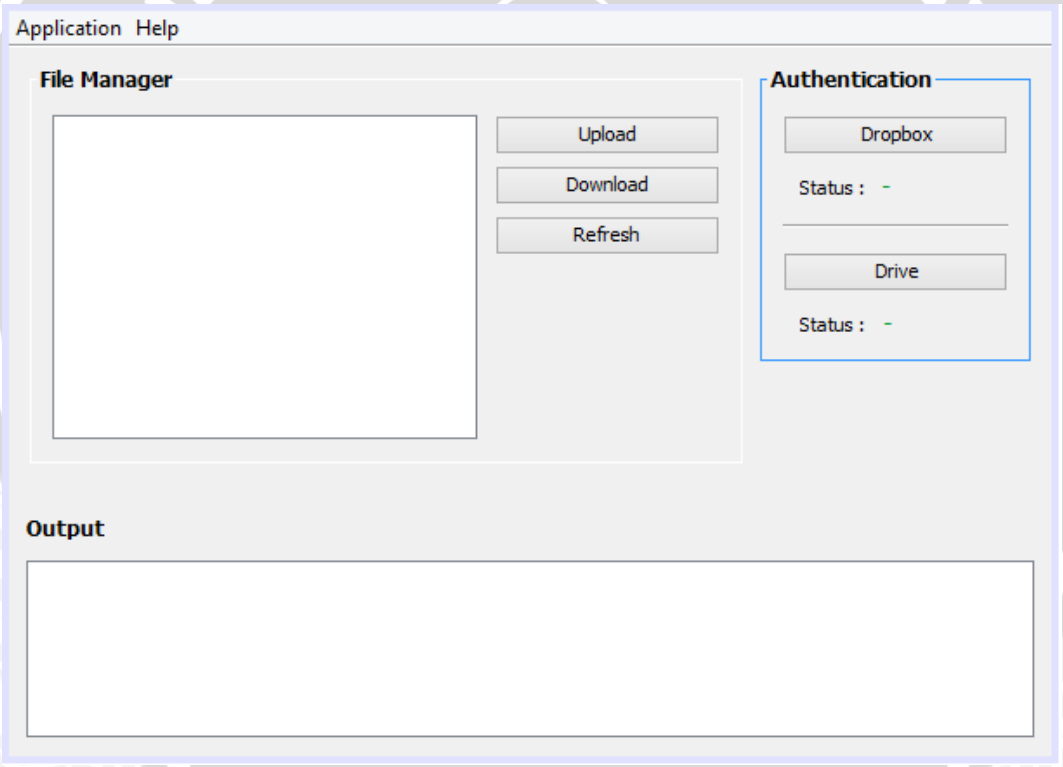
Gambar 4.3 Client ID & client secret Google Drive

4.1.2 Pengembangan antarmuka

Berdasarkan rancangan sistem sebelumnya, sistem yang dibangun ini harus mampu memenuhi kebutuhan yang telah ditentukan. Untuk itu dibangun antarmuka yang diharapkan mampu memudahkan pengguna dalam menjalankan aplikasi.

4.1.2.1 Tampilan utama

Tampilan utama merupakan tampilan dimana seluruh fungsi aplikasi berada. Pada antarmuka ini, terdapat beberapa komponen diantaranya: *menu bar, list, label, button dan text area*. *Menu bar* berfungsi menampilkan menu-menu yang disediakan oleh aplikasi, misalkan: *setting, exit, help & about.* List berfungsi menampilkan seluruh berkas yang telah di-*upload* pada layanan *cloud. Button* digunakan untuk menjalankan fungsi-fungsi tertentu. Sedangkan *text area* digunakan untuk menampilkan informasi proses-proses yang dilakukan, seperti saat melakukan *upload* & *download*. Text area ini tidak hanya ditampilkan pada Berikut tampilan utama aplikasi:

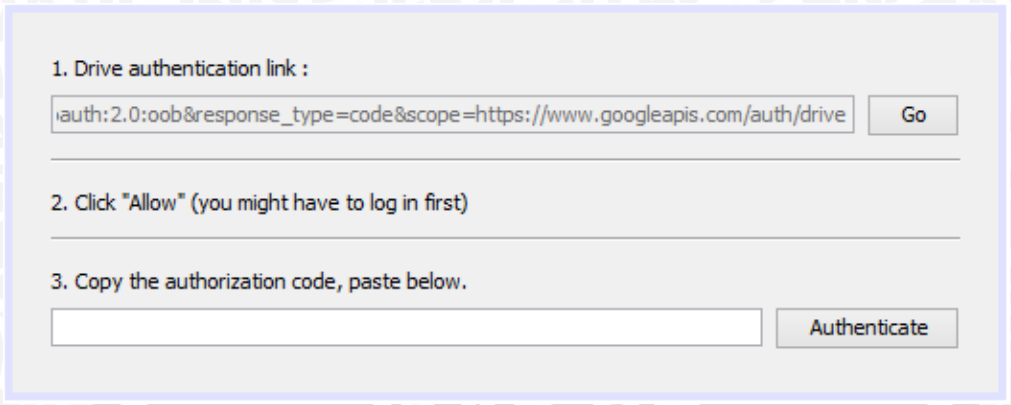


Gambar 4.4 Tampilan utama aplikasi

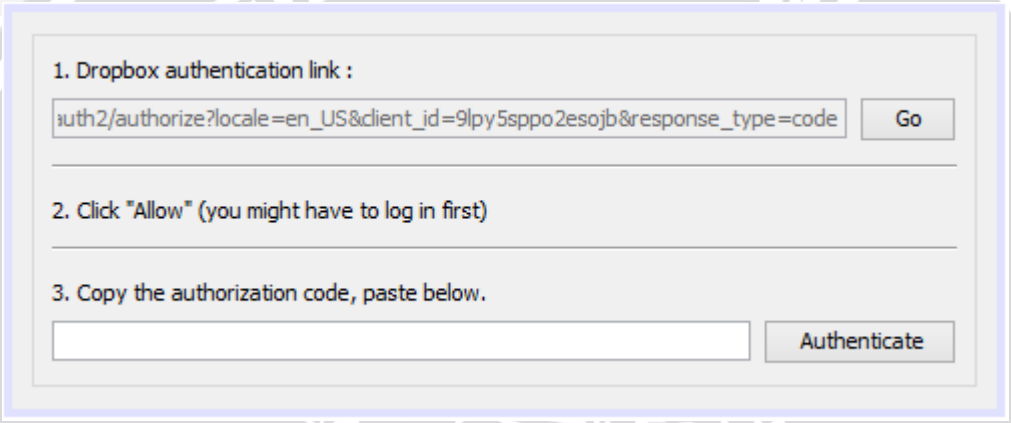
4.1.2.2 Tampilan otentikasi

Tampilan diberikan untuk memudahkan pengguna ketika akan melakukan otentikasi pada layanan *cloud*. Pada tampilan ini terdapat beberapa komponen diantaranya: *label, button & text field. Label* disini berfungsi sebagai teks untuk petunjuk penggunaan. *Button* berfungsi sebagai komponen yang akan mengarahkan pengguna pada *link* atau proses selanjutnya. Sedangkan *text field*

digunakan sebagai masukan kode otorisasi yang didapatkan. Berikut tampilan otentikasi pada aplikasi:



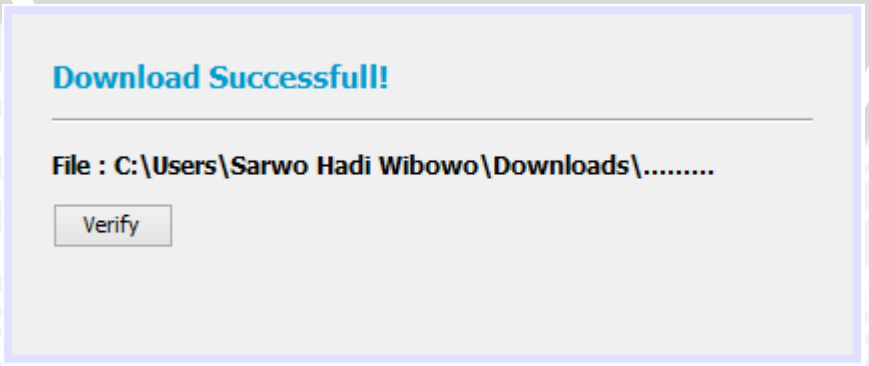
Gambar 4.5 Tampilan otentikasi Drive



Gambar 4.6 Tampilan otentikasi Dropbox

4.1.2.3 Tampilan hash

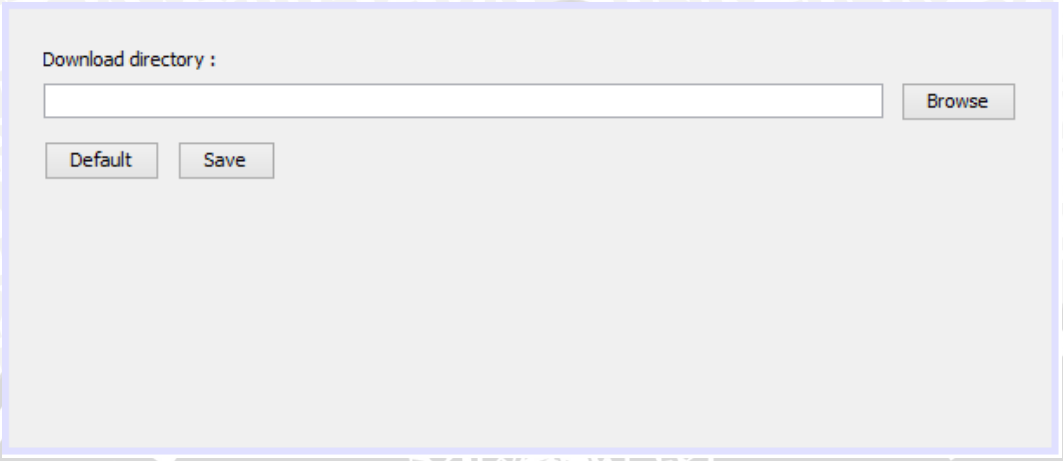
Tampilan ini dijadikan sekaligus sebagai notifikasi jika proses *download* telah selasai. Sehingga diharapkan pengguna dapat secara langsung melakukan verifikasi berkas yang telah di-*download*. Pada tampilan ini hanya diperlukan dua komponen yaitu: *label* sebagai teks informasi *& button* sebagai tombol untuk melakukan verifikasi berkas. Berikut tampilan hash pada aplikasi:



Gambar 4.7 Tampilan hash

4.1.2.4 Tampilan *setting*

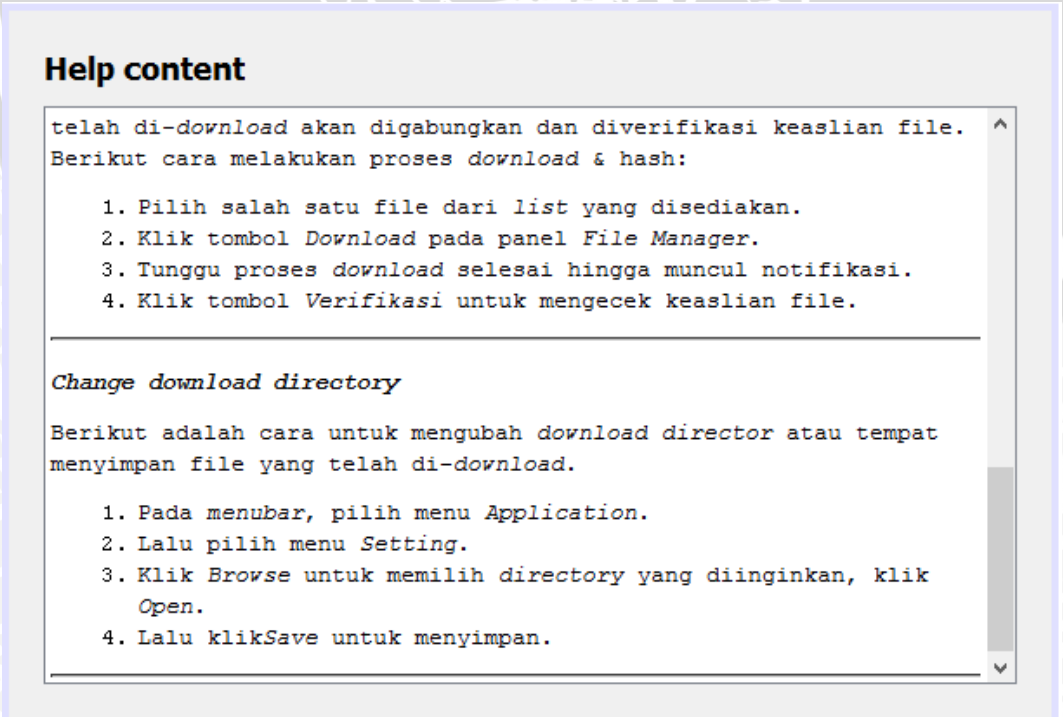
Pada tampilan *setting* hanya terdapat beberapa komponen, yaitu: *label, button & text field*. Pada tampilan ini pengguna hanya dapat mengubah *directory download.*



Gambar 4.8 Tampilan *setting*

4.1.2.5 Tampilan *help*

Tampilan ini hanya digunakan sebagai bantuan bagi pengguna dalam menjalankan fitur-fitur utama aplikasi. Di sini dijelas cara menjalankan setiap fitur secara bertahap.



Gambar 4.9 Tampilan *help*

4.1.2.6 Tampilan *about*

Pada tampilan *about* disertakan penjelasan sedikit mengenai aplikasi dan informasi tentang versi serta developer aplikasi.

Gambar 4.10 Tampilan *about*

4.1.3 Pengembangan aplikasi (*coding*)

Pada tahap ini dilakukan implementasi aplikasi berdasarkan kebutuhan dan antarmuka yang telah dibuat sebelumnya.

4.1.3.1 Fungsi otentikasi

Dalam aplikasi ini terdapat dua *method* untuk otentikasi, yaitu *DropboxAuth()* pada Dropbox & *DriveAuth()* pada Drive. Pada intinya, *method* tersebut berfungsi untuk melakukan otentikasi pengguna pada masing-masing layanan. *Method* tersebut bekerja jika pegguna telah menginputkan *authCode* (kode otentikasi). *AuthCode* didapatkan ketika pengguna melakukan otentikasi pada link yang telah ditentukan (lihat Gambar 4.11 dan Gambar 4.12). Kemudian dilakukan *generate authCode* yang kemudian dihasilkan token *accessToken* (Dropbox) & *googleTokenResponse* (Drive). Token tersebut yang akan dijadikan sebagai *credentials* untuk menggunakan/mengakses operasi lain pada layanan. *Source code* 4.2 dan *Source code* 4.3 merupakan potongan kode program fungsi otentikasi. Untuk seluruh kode program dapat dilihat pada lampiran dari kelas Dropox (*DropboxClass.java*) & kelas Drive (*DriveClass.java*).

```
1  final String APP_KEY = "clientid";
2  final String APP_SECRET = "clientsecret";
3
4  DbxAppInfo appInfo = new DbxAppInfo(APP_KEY, APP_SECRET);
5
6  DbxRequestConfig config = new
7  DbxRequestConfig("JavaTutorial/1.0",
8     Locale.getDefault().toString());
9  DbxWebAuthNoRedirect webAuth = new
10    DbxWebAuthNoRedirect(config, appInfo);
11
12 // Have the user sign in and authorize your app.
13 String authorizeUrl = webAuth.start();
```

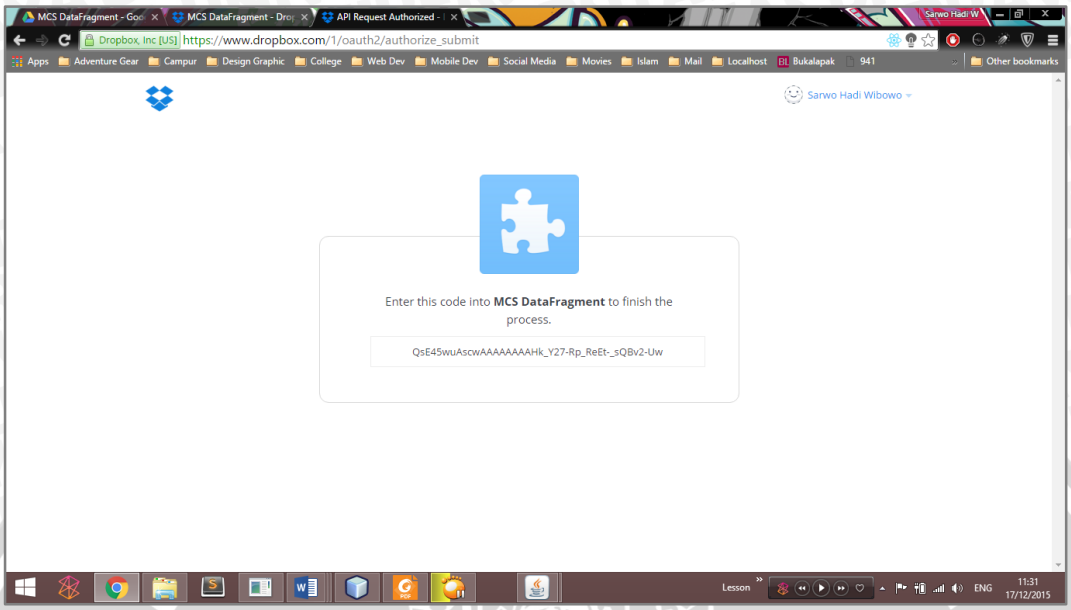
```
14  DbxClient klien;
15
16  public long DropboxAuth(String authCode) throws DbxException
17  {
18      String code = authCode;
19      DbxAuthFinish authFinish = webAuth.finish(code);
20      String accessToken = authFinish.accessToken;
21
22      DbxClient client = new DbxClient(config, accessToken);
23
24      String user = client.getAccountInfo().displayName;
25      klien = client;
26      System.out.println("Dropox linked Account : " + user);
27      System.out.println("---------------------------------");
28
29      long uid = client.getAccountInfo().userId;
30
31      return uid;
32  }
```

***Source code* 4.2 Kode fungsi otentikasi Dropbox**

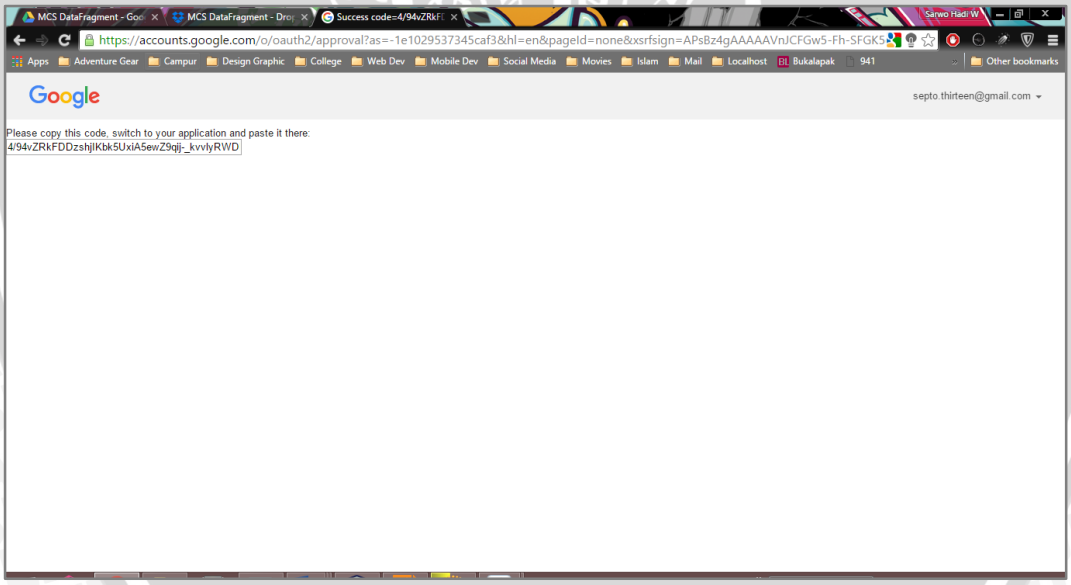
```
1   public static void DriveAuth(String code) throws
2   IOException, URISyntaxException, GeneralSecurityException {
3      String CLIENT_ID = "clientid";
4      String CLIENT_SECRET = "clientsecret";
5      String REDIRECT_URI = "urn:ietf:wg:oauth:2.0:oob";
6      HttpTransport httpTransport = new NetHttpTransport();
7      JsonFactory jsonFactory = new JacksonFactory();
8      GoogleAuthorizationCodeFlow flow = new
9         GoogleAuthorizationCodeFlow.Builder(httpTransport,
10           jsonFactory, CLIENT_ID, CLIENT_SECRET,
11           Arrays.asList(DriveScopes.DRIVE))
12           .setAccessType("online")
13           .setApprovalPrompt("auto").build();
14     String url = flow.newAuthorizationUrl()
15        .setRedirectUri(REDIRECT_URI).build();
16
17     HttpTransport ght = GoogleNetHttpTransport
18        .newTrustedTransport();
19
20     GoogleTokenResponse response = flow.newTokenRequest(code)
21        .setRedirectUri(REDIRECT_URI).execute();
22     GoogleCredential credential = new GoogleCredential()
23        .setFromTokenResponse(response);
24
25     // Create a new authorized API client
26     Drive authKey = new Drive.Builder(httpTransport,
27        jsonFactory, credential).setApplicationName(
28           "MCS DataFragment").build();
29
30     service = authKey;
31     ht = ght;
32
33     About about = authKey.about().get().execute();
34
35     System.out.println("Drive linked Account : " +
36        about.getName());
37     System.out.println("----------------------------");
```

```
38 }
```

***Source code* 4.3 Kode fungsi otentikasi Drive**



Gambar 4.11 *AuthCode* Dropbox



Gambar 4.12 *AuthCode* Google Drive

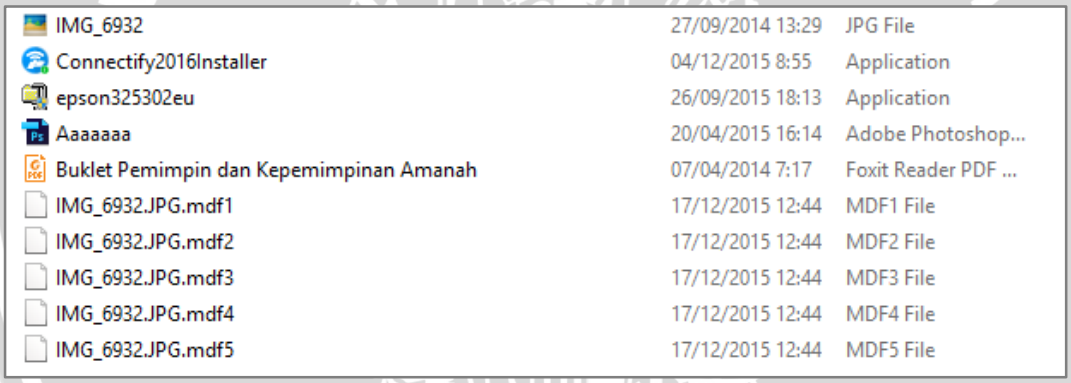
4.1.3.2 Fungsi *split*

Implementasi fungsi *split* pada aplikasi ini dapat dilihat pada *Source code* 4.4. Fungsi *split* pada aplikasi ini ditunjukkan oleh *method doSplit()* pada kelas *FileSplit.java. Method doSplit()* merupakan fungsi yang digunakan untuk memecah berkas menjadi bagian-bagian yang lebih kecil. *Method* tersebut membutuhkan masukan berupa berkas yang akan dipecah. Setelah masukan dimasukkan, *method* ini akan mengakses berkas eksternal secara bebas dari kelas *RandomAccessFile*. Kelas tersebut tidak perlu membaca seluruh isi berkas, melainkan hanya membaca

sesuai ukuran *byte* yang ditentukan dan proses ini berjalan dari *byte* awal hingga byte *akhir*. Setiap selesai pembacaan, berkas akan langsung ditulis ulang sesuai ukuran saat pembacaan. Sehingga akan didapatkan pecahan-pecahan berkas seperti pada Gambar 4.13. Berikut potongan kode program fungsi *split*.

```
1   public void doSplit() throws Exception {
2         fout = new RandomAccessFile(files.getPath() + ".mdf" +
3            j++, "rw");
4
5         while ((bacaInt = fin.read(b)) != -1) {
6            fout.write(b, 0, bacaInt);
7            e += bacaInt;
8
9            // Split size (random size)
10           if (e == 1024 * 1024 * sizeSplit) {
11              e = 0L;
12                 fout.close();
13                 doSplit();
14              }
15           }
16           cf = j;
17        }
```

Source code* 4.4 Kode fungsi *split



Gambar 4.13 Hasil *split* berkas

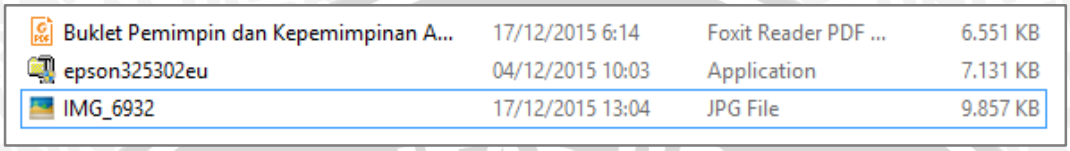
4.1.3.3 Fungsi *merge*

Implementasi fungsi *merge* pada aplikasi ini menggunakan *method doMerge()* pada kelas *FileMerge.java*. *Method doMerge()* merupakan fungsi yang digunakan untuk menggabungkan bagian-bagian berkas menjadi satu berkas yang utuh. *Method* tersebut membutuhkan masukan berupa berkas pecahan yang pertama. Proses pada fungsi ini adalah kebalikan dari proses *split* pada fungsi yang dijelaskan sebelumnya. Fungsi ini nantinya akan menghasilkan berkas asli seperti pada Gambar 4.14. *Source code* 4.5 di bawah berisi potongan kode program fungsi *merge*.

```
1   public void doMerge(File fname) throws Exception {
2         if (fname.exists()) {
3            in = new RandomAccessFile(fname, "r");
4
```

```
5           while ((nd = in.read(b)) != -1) {
6              out.write(b, 0, nd);
7           }
8           in.close();
9           doMerge(new File(fname.getPath().replace(
10             ".mdf" + j, ".mdf" + (++j))));
11       }
12       cf = j;
13    }
```

Source code* 4.5 Kode fungsi *merge



Gambar 4.14 Hasil *merge* berkas

4.1.3.4 Fungsi *upload*

Pada aplikasi ini terdapat dua *method upload* pada kelas yang berbeda, yaitu *DropboxUpload()* dan *DriveUpload(). Method* ini berfungsi meng-*upload* berkas pada masing-masing layanan *cloud.* Untuk melakukan *upload*, kedua *method* ini memerlukan masukan *filename* & *filepath.* Sedangkan pada method *DriveUpload()* dibutuhkan masukan tambahan *parentID* untuk menentukan dimana letak berkas yang akan di-*upload*. Pada dasarnya, fungsi ini mengubah berkas masukan menjadi berkas berbentuk *input stream,* selanjutnya *input stream* tersebut yang akan dikirimkan pada media penyimpanan (lihat Gambar 4.15 dan Gambar 4.16). Berikut *Source code* 4.6 dan *Source code* 4.7 berisi potongan kode program fungsi *upload.*

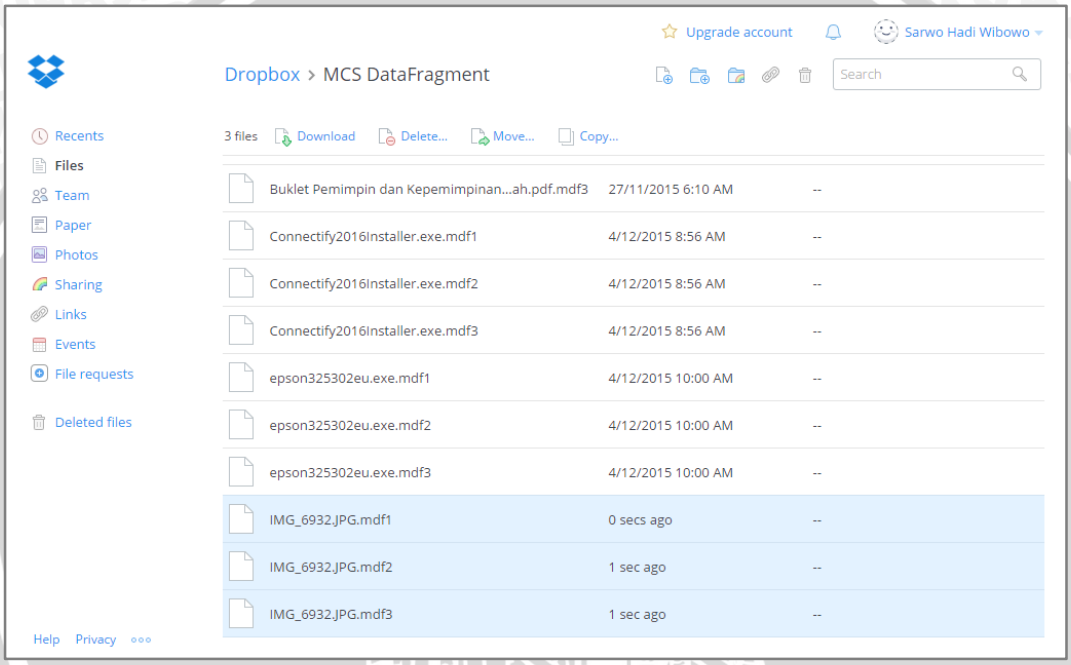
```
1   public void DropboxUpload(String filepath, String filename)
2      throws DbxException, IOException {
3        File inputFile = new File(filepath);
4        FileInputStream inputStream = new
5           FileInputStream(inputFile);
6        try {
7           DbxEntry.File uploadedFile =
8              klien.uploadFile("/MCS DataFragment/" +
9              filename, DbxWriteMode.add(),
10             inputFile.length(), inputStream);
11          System.out.println(filename +
12             " has been uploaded");
13       } finally {
14          inputStream.close();
15       }
16  }
```

***Source code* 4.6 Kode fungsi *upload* Dropbox**

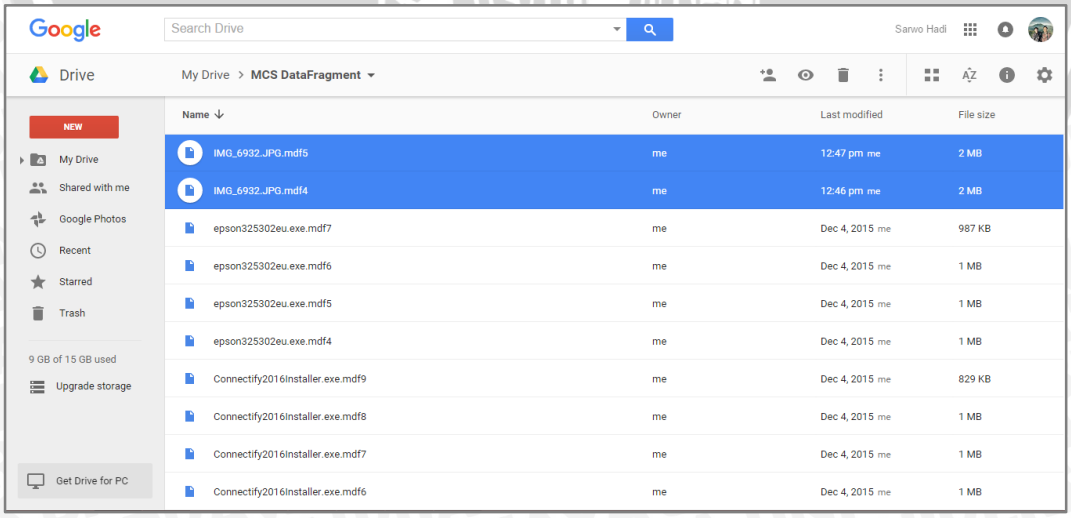
```
1   public static String DriveUpload(String path, String
2      filename, String parentId) throws IOException {
3
4        // Upload a file
5        File body = new File();
6        body.setTitle(filename);
```

```
7         body.setMimeType("*/*");
8         body.setParents(Arrays.asList(newParentReference().
9            setId(parentId)));
10        java.io.File fileContent = new java.io.File(path);
11        FileContent mediaContent = new FileContent("*/*",
12           fileContent);
13        File file = service.files().insert(body,
14           mediaContent).execute();
15        System.out.println(filename + " has been uploaded");
16
17        return file.getId();
18    }
```

***Source code* 4.7 Kode fungsi *upload* Drive**



Gambar 4.15 *Upload* berkas pada layanan Dropbox



Gambar 4.16 *Upload* berkas pada layanan Google Drive

4.1.3.5 Fungsi *download*

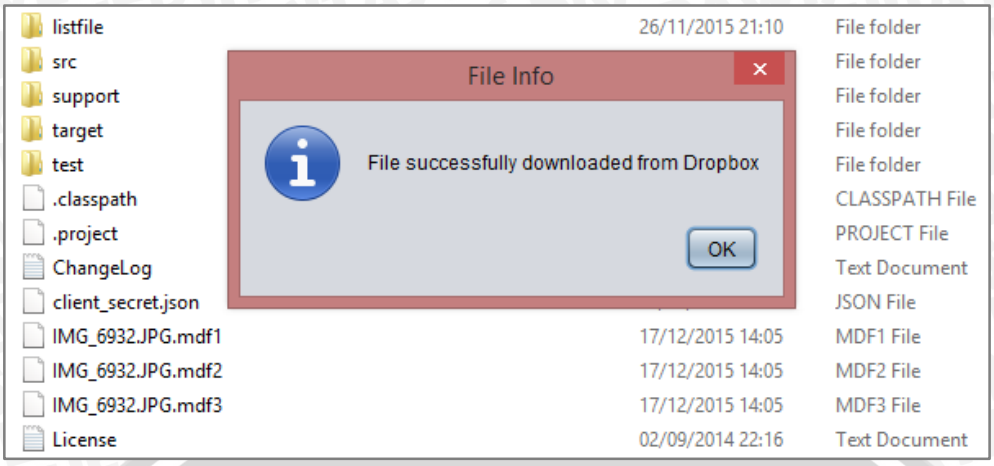
Pada fungsi *download* juga terdapat dua *method* pada kelas yang berbeda, yaitu *DropboxDownload()* dan *DriveDownload(). Method* ini berfungsi men-*download* berkas dari masing-masing layanan *cloud.* Untuk melakukan *download*, *method* ini memerlukan masukan berupa nama atau ID berkas. Berbeda dengan fungsi sebelumnya, fungsi ini mengubah masukan menjadi *output stream* agar dapat di-*download* dari media penyimpanan (lihat Gambar 4.17 dan Gambar 4.18). Berikut *Source code* 4.8 dan *Source code* 4.9 berisi potongan kode program fungsi *download.*

```
1  public void DropboxDownload(String selected) throws
2     DbxException, IOException {
3        // Write file to local storage (Downloaded file
4           location)
5        FileOutputStream outputStream = new
6           FileOutputStream(selected);
7        try {
8           DbxEntry.File downloadedFile = klien.getFile(
9              "/MCS DataFragment/" + selected, null,
10             outputStream);
11          System.out.println(selected +
12             " successfully downloaded");
13       } finally {
14          outputStream.close();
15       }
16 }
```

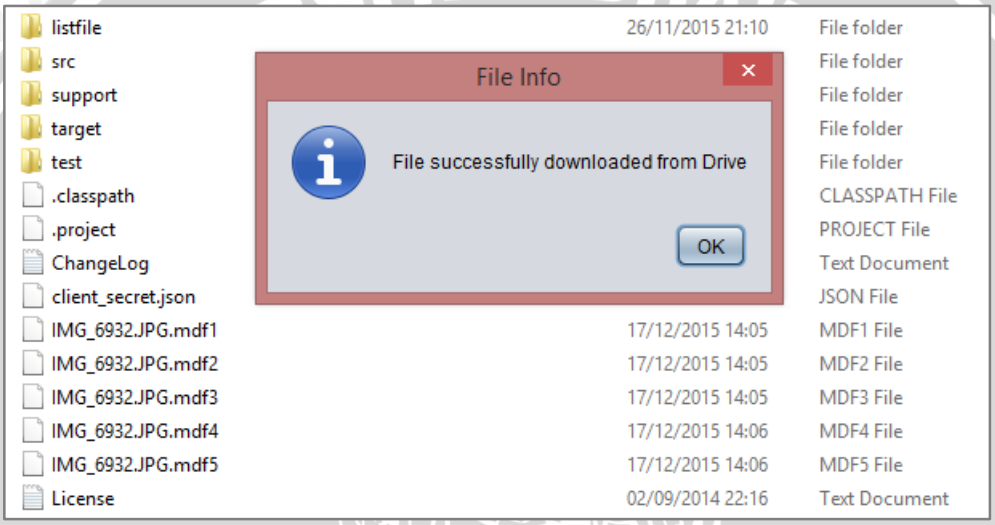
***Source code* 4.8 Kode fungsi *download* Dropbox**

```
1  public static void DriveDownload(String fileID,
2     String local) throws IOException {
3
4        // Write file to local storage (Downloaded file
5           location)
6        FileOutputStream os = new FileOutputStream(local);
7        File elif = service.files().get(fileID).execute();
8        String link = elif.getDownloadUrl();
9
10       MediaHttpDownloader dl = new MediaHttpDownloader(ht,
11          service.getRequestFactory().getInitializer());
12
13       dl.setDirectDownloadEnabled(false);
14       dl.setChunkSize(calcChunkSize(1));
15
16       dl.download(new GenericUrl(link), os);
17       System.out.println(local + " successfully downloaded");
18 }
```

***Source code* 4.9 Kode fungsi *download* Drive**



Gambar 4.17 *Download* berkas dari layanan Dropbox



Gambar 4.18 *Download* berkas dari layanan Drive

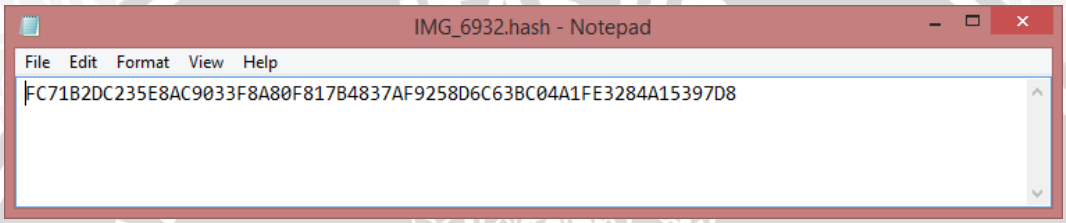
4.1.3.6 Fungsi Hash

Pada implementasi aplikasi ini terdapat *method checksum()* dan *toHex()* sebagai *method* utama fungsi hash. Method ini berfungsi untuk mengonversi suatu berkas menjadi nilai hash. Nilai hash ini yang akan disimpan setiap melakukan *upload* berkas. Pada *method* ini diperlukan masukan berupa berkas. Dari masukan tersebut, berkas diubah menjadi *input stream*. Kemudian *input stream* diubah menjadi nilai hash melalui proses yang ada pada kelas *MessageDigest*. Nilai yang dihasilkan dapat dilihat pada Gambar 4.19, yaitu berupa gabungan angka dan huruf sepanjang 64 karakter (32 byte). Berikut potongan kode fungsi hash dapat dilihat pada *Source code* 4.10.

```
1  public byte[] checksum(File input) {
2      try {
3          FileInputStream in = new FileInputStream(input);
4          MessageDigest digester =
5              MessageDigest.getInstance("SHA-256");
6          byte[] block = new byte[4096];
7          int length;
```

```
8           while ((length = in.read(block)) > 0) {
9               digester.update(block, 0, length);
10          }
11
12          return digester.digest();
13      } catch (Exception e) {
14          e.printStackTrace();
15          return null;
16      }
17  }
18
19  private static String toHex(byte[] bytes) {
20      return DatatypeConverter.printHexBinary(bytes);
21  }
```

***Source code* 4.10 Kode fungsi hash**



Gambar 4.19 Nilai hash

4.2 Pengujian

Dalam penelitian ini, pengujian dibedakan menjadi dua tahap yaitu: pengujian validasi & pengujian keamanan. Pengujian validasi dilakukan untuk mengetahui setiap fitur apakah telah sesuai dengan kebutuhan aplikasi atau belum. Sedangkan untuk pengujian keamanan, dilakukan untuk mengetahui apakah aplikasi yang dibuat dapat memenuhi beberapa aspek keamanan jaringan yang ditentukan atau tidak.

4.2.1 Pengujian validasi fitur

Pada pengujian ini dilakukan uji coba pada masing-masing fitur. Pengujian dilakukan untuk mengetahui fungsionalitas fitur apakah sudah berjalan baik atau belum. Pengujian dilakukan dengan cara menjalankan semua fitur yang ada pada aplikasi, dan memastikan semua fitur berjalan dengan baik. Fitur yang telah ditentukan sebelumnyalah yang dijadikan acuan dalam melakukan pengujian ini.

4.2.2 Pengujian keamanan data

Pada pengujian keamanan ini, hanya dilakukan dua dari 4 aspek keamanan pada umumnya, diantaranya: *integrity data & confidentiality data*. Pengujian dilakukan untuk mengetahui seberapa kuat jaminan keamanan data yang diberikan oleh aplikasi. Pengujian akan dilakukan dengan menjalankan fitur pengecekan hash dan uji *confidentiality*. Keamanan data yang diberikan akan diuji sesuai dengan skenario yang ditentukan.

BAB 5 HASIL PENGUJIAN DAN ANALISIS

5.1 Hasil pengujian

Pengujian yang dilakukan pada penelitian ini meliputi beberapa macam pengujian, meliputi pengujian validasi fitur aplikasi dan pengujian keamanan jaringan (*integrity & confidentiality*).

5.1.1 Hasil pengujian validasi fitur

Pengujian validasi dilakukan untuk memastikan semua fitur yang telah dibuat pada aplikasi berjalan dengan baik sesuai kebutuhan yang telah dirancang sebelumnya. Berikut hasil pengujian validasi fitur aplikasi dapat dilihat pada Tabel 5.1. Pengujian dikelompokkan berdasarkan kategori masing-masing.

Tabel 5.1 Hasil pengujian validasi fitur

Fitur layanan	Status
Dropbox *authentication*	Terpenuhi
Drive *authentication*	Terpenuhi
Dropbox *upload*	Terpenuhi
Drive *upload*	Terpenuhi
Dropbox *download*	Terpenuhi
Drive *download*	Terpenuhi

Fitur *split & merge*	Status
***Split* berkas**	Terpenuhi
***Merge* berkas**	Terpenuhi

Fitur *list*	Status
Tampilan *list* berkas	Terpenuhi
***Refresh list* berkas**	Terpenuhi
Membuat *upload list* berkas	Terpenuhi

Fitur hash	Status
Membuat berkas hash	Terpenuhi
Pengecekan hash	Terpenuhi

Tabel 5.1 Hasil pengujian validasi fitur (lanjutan)

Fitur *menubar*	Status
Membuka tampilan utama	Terpenuhi
Membuka tampilan setting	Terpenuhi
Ubah *directory download*	Terpenuhi
Exit aplikasi	Terpenuhi
Help	Terpenuhi
About	Terpenuhi

Dari hasil pengujian fungsionalitas aplikasi pada Tabel 5.1 dapat dilihat bahwa fungsi dari tiap-tiap fitur terpenuhi dan berjalan dengan baik.

5.1.2 Hasil pengujian keamanan jaringan

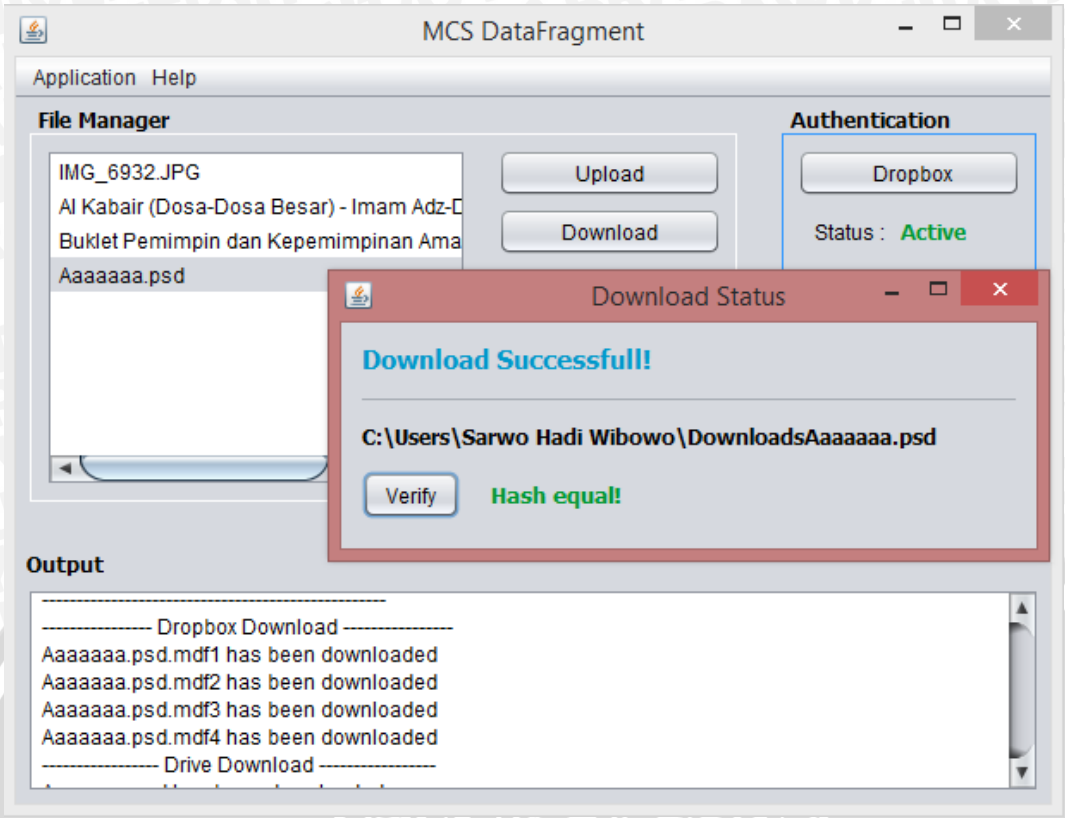
Pengujian keamanan jaringan dilakukan untuk mengetahui seberapa tingkat keamanan yang diberikan oleh sistem. Untuk itu dilakukan percobaan pada fungsi hash untuk mengetahui integritas/keutuhan data dan fungsi *split* untuk mengetahui seberapa kerahasiaan data yang diberikan.

5.1.2.1 *Integrity*

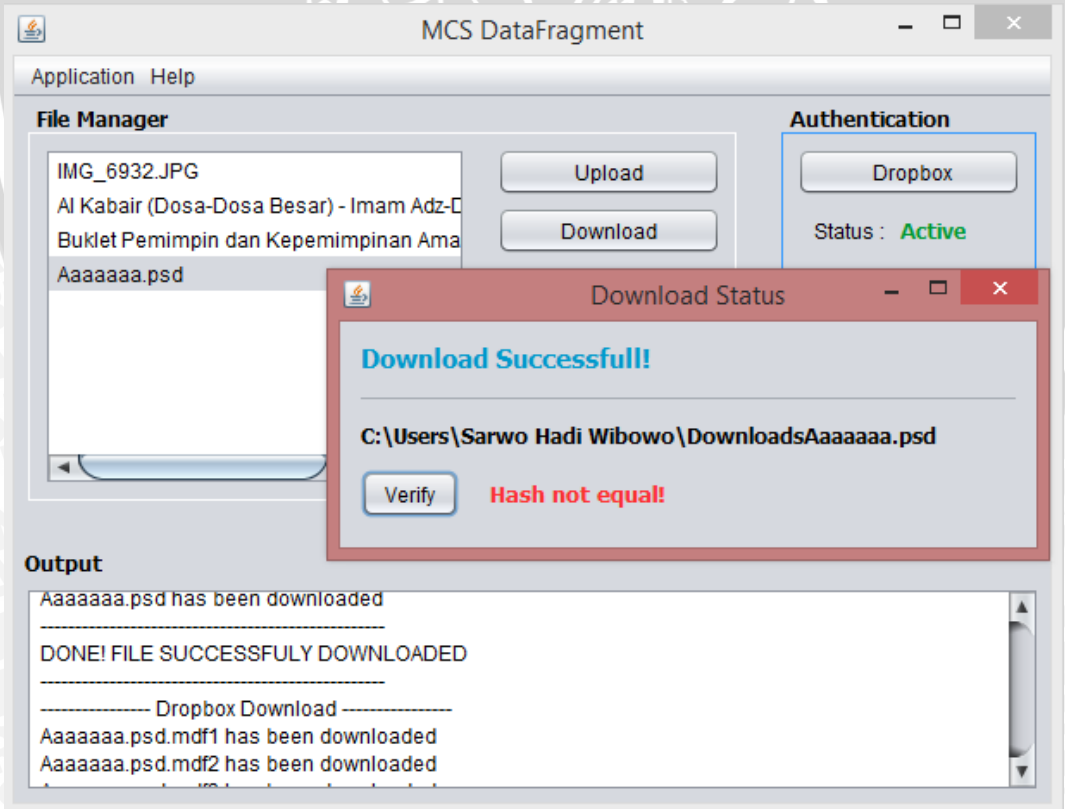
Pada pengujian integritas data dilakukan pengujian dengan 3 skenario yang berbeda. Skenario dan keseluruhan hasil pengujian ini dapat dilihat pada Tabel 5.2. Pada Gambar 5.1 ditampilkan respon sistem ketika diuji dengan skenario I1, sendangkan pada Gambar 5.2 ditampilkan respon sistem ketika diuji dengan skenario I2 dan I3.

Tabel 5.2 Hasil pengujian integritas data

ID	Skenario	Respon	Status
I1	Tidak dilakukan perubahan pada berkas	Hash equal	Terpenuhi
I2	Penguijan sensitifitas (*minor changes*)	Hash not equal	Terpenuhi
I3	Menghilangkan beberapa bagian berkas	Hash not equal	Terpenuhi



Gambar 5.1 Hasil jidak dilakukan perubahan pada berkas



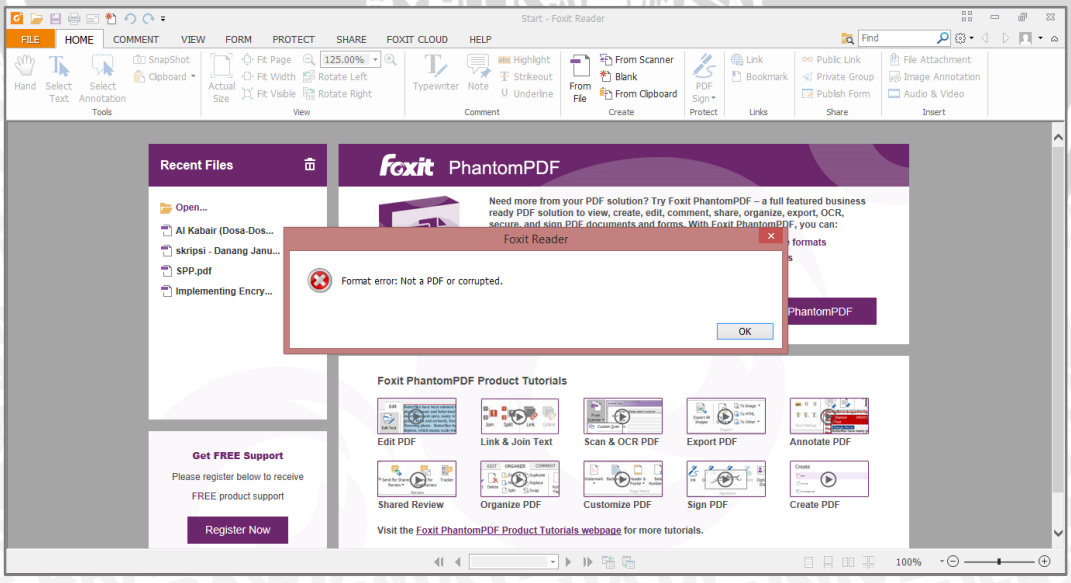
Gambar 5.2 Hasil jika terjadi perubahan pada berkas

5.1.2.2 *Confidentiality*

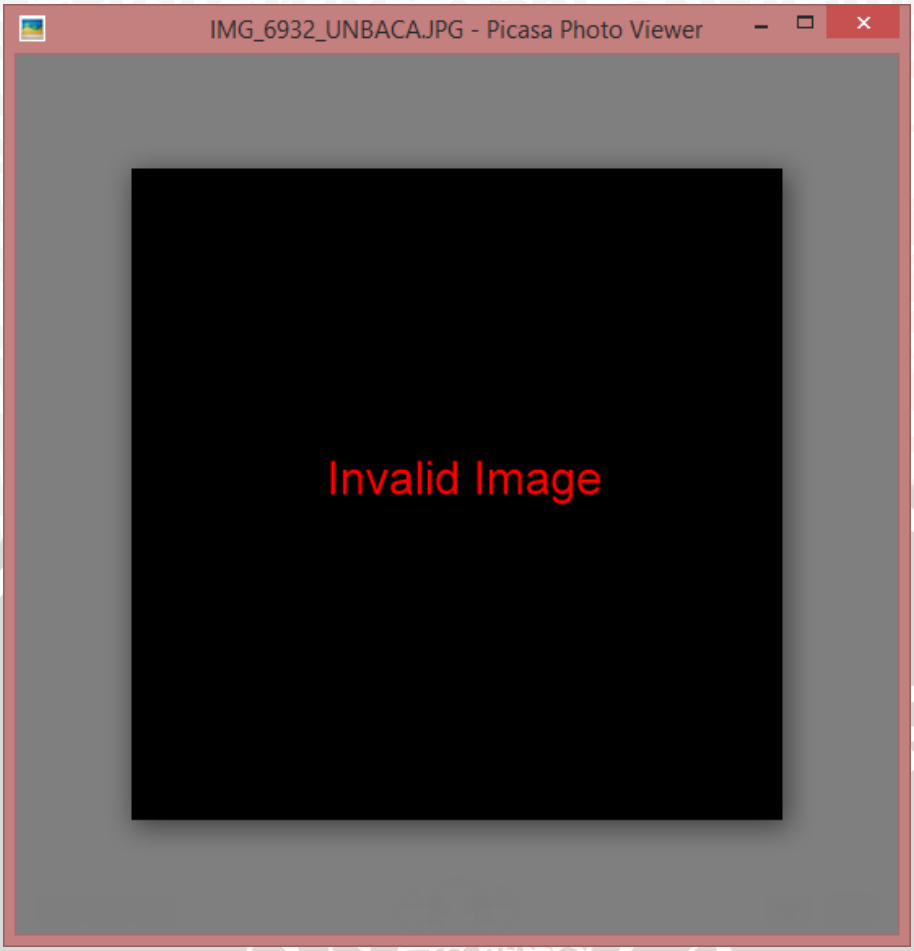
Pada pengujian kerahasiaan data, dilakukan pengujian dengan 4 skenario pada berkas yang telah dipecah oleh sistem. Skenario keseluruhan hasil dapat dilihat pada Tabel 5.3. Pada Gambar 5.3 dan Gambar 5.5 ditampilkan hasil uji skenario C2 dan C4, yaitu berkas yang telah dipecah tidak dapat dibaca jika tidak digabung terlebih dahulu. Sedangkan pada Gambar 5.5, berkas multimedia tetap dapat dibaca jika diakses secara langsung pada berkas pertama ataupun jika digabung secara berurutan dengan berkas pertama.

Tabel 5.3 Hasil pengujian kerahasiaan data

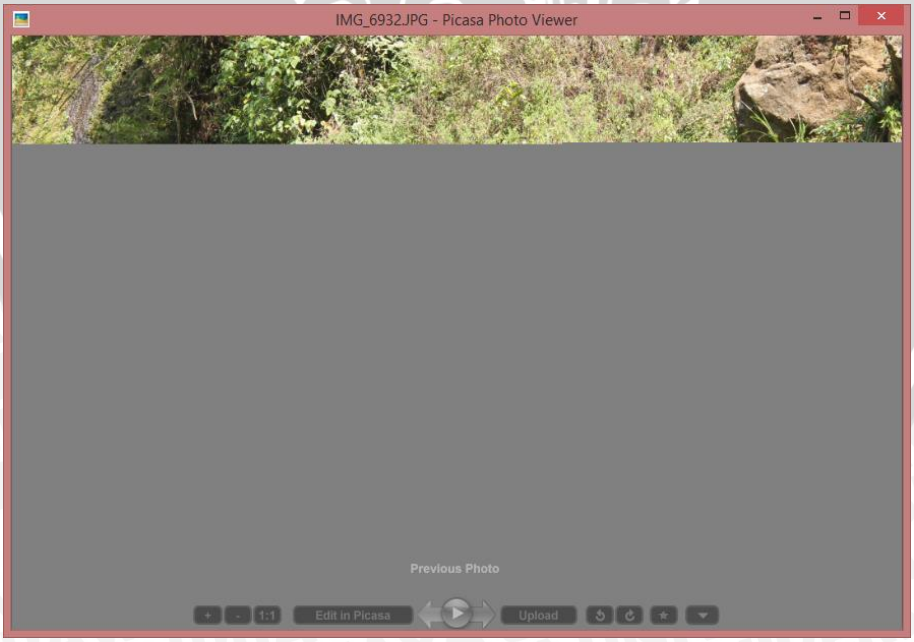
ID	Skenario	Tipe berkas dokumen	Tiper berkas multimedia	Kerahasiaan
C1	Akses salah satu bagian berkas (bagian pertama)	Tidak terbaca	Terbaca (sebagian)	Kurang
C2	Akses salah satu berkas (bagian selanjutnya)	Tidak terbaca	Tidak terbaca	Terjaga
C3	Akses berkas setelah digabung dengan bagian yang lain berurutan dari bagian pertama (tidak lengkap) Contoh: Berkas pada *cloud* 1	Tidak terbaca	Terbaca (sebagian)	Kurang
C4	Akses berkas setelah digabung dengan bagian yang lain berurutan dari bagian selanjutnya (tidak lengkap) Contoh : Berkas pada *cloud* 2	Tidak terbaca	Tidak terbaca	Terjaga



Gambar 5.3 Berkas dokumen tidak terbaca



Gambar 5.4 Berkas multimedia tidak terbaca



Gambar 5.5 Berkas terbaca sebagian

5.2 Analisis

Pada penelitian ini, analisis dilakukan dengan melakukan pengujian validasi terhadap 5 kategori fitur yang ada. Pada masing-masing kategori dilakukan pengujian secara fungsionalitas pada tiap sub fitur untuk memastikan fitur tersebut telah berjalan dengan benar dan memenuhi kebutuhan yang ditentukan. Dari hasil pengujian, dapat diketahui bahwa semua fitur berjalan dengan baik sesuai kebutuhan.

Dari hasil pengujian keamanan yang dilakukan, pada aspek *integrity* dapat dipastikan bahwa berkas yang diterima dari media penyimpanan *cloud* dapat diketahui keutuhan datanya walaupun terjadi perubahan yang sangat kecil. Sedangkan pada aspek *confidentiality*, data yang telah disimpan dapat dijaga kerahasiaannya. Akan tetapi masih dimungkinkan sebagian isi data terbaca, kelemahan ini hanya akan terjadi jika data tersebut bertipe multimedia seperti video dan gambar. Berdasarkan hasil pengujian yang dilakukan, diketahui bahwa kelemahan ini diakibatkan dari potongan data multimedia yang pertama.

BAB 6 KESIMPULAN

Bab ini berisi kesimpulan dari hasil implementasi dan analisis sistem serta saran untuk pengembangan sistem selanjutnya.

6.1 Kesimpulan

Dari hasil implementasi, pengujian, dan analisis pada implementasi aplikasi ini dapat disimpulkan bahwa:

1. Untuk mencegah adanya *loss of privacy & theft of information* pada media penyimpanan dirancang sebuah metode yang mengombinasikan *data splitting* dan *multiple cloud storage*. Data splitting berfungsi membagi data menjadi beberapa bagian. Sedangkan *multiple cloud storage* diperlukan sebagai penunjang agar pecahan data tidak terkumpul satu dengan yang lain.
2. Dalam meningkatkan keamanan data, sistem yang dirancang diimplementasikan pada sebuah aplikasi berbasis *desktop* yang memanfaatkan Dropbox & Google Drive sebagai media penyimpanan *cloud*. Selain itu diperlukan library (API) *dropbox-java-sdk-1.7.7* & *google-api-services-drive-v2-rev130-1.18.0-rc* sebagai penghubung aplikasi dengan layanan *cloud*.
3. Berdasarkan rancangan yang telah dibuat, sistem mampu menjamin 2 aspek keamanan jaringan, yaitu: *confidentiality* dan *integrity* data. Di mana data yang telah di-*split* tidak dapat dibaca secara utuh/sempurna sebelum digabungkan dengan bagian yang tepat dan keaslian data dapat dijamin dengan adanya fungsi hash.

6.2 Saran

Untuk keperluan penelitian atau pengembangan lebih lanjut, penulis menyampaikan beberapa saran yang mungkin dapat diterapkan:

1. Untuk meningkatkan keamanan pada aspek kerahasiaan data, perlu dilakukan penelitian lebih lanjut untuk pengembangan metode *split* yang digunakan. Selain itu dapat ditambahkan proses enkripsi pada berkas sebelum dilakukan *split* ataupun sesudah dilakukan *split*. Sedangkan untuk meningkatkan keamanan pada aspek ketersediaan data, perlu dilakukan replikasi data.
2. Perlu ditambahkan fitur-fitur pendukung, seperti fitur yang dapat menangani jika terjadi kegagalan saat proses *upload/download* berkas, yaitu fitur *resume*. Karena akan memudahkan pengguna ketika terjadi kegagalan di tengah proses *upload/download*, sehingga dapat mengefektifkan waktu dan *bandwidth*. Selain fitur *resume*, fitur *share* juga dapat ditambahkan pada pengguna jika ingin berbagi berkas.

3. Membangun sebuah server basis data sebagai tempat mengelola metadata berkas (data yang tersimpan pada media penyimpanan *cloud)* tiap pengguna.

DAFTAR PUSTAKA

Adrianus, E., 2010. *Theft of Information di dalam Teknologi Cloud Computing dan Teknik Mengamankan Data sebagai Pencegahan Pencurian Data,* Bandung: s.n.

Anon., 2015. *What is Maven? | The Javabook.* [online] Tersedia di: <http://www.thejavabook.com/2015/09/what-is-maven> [Diakses 13 Desember 2015]

Assagaf, N., 2012. *Layanan Cloud Storage | Cloud Indonesia.* [online] Tersedia di: <http://www.cloudindonesia.or.id/layanan-cloud-storage.html> [Diakses 11 Desember 2015]

Assagaf, N., 2012. *Mengenal & Menggunakan Google Drive | Cloud Indonesia.* [online] Tersedia di: <http://www.cloudindonesia.or.id/mengenal-menggunakan-google-drive.html> [Diakses 11 Desember 2015]

Avestro, J., 2007. *JENI Pengenalan Pemrograman 1.* 1.2 ed. s.l.:s.n.

Beal, V., n.d. *What is Cloud Storage | Webopedia.* [online] Tersedia di: <http://www.webopedia.com/TERM/C/cloud_storage.html> [Diakses 23 Desember 2015]

Binus, 2013. *Library Binus University.* [e-book] Tersedia di: <http://library.binus.ac.id/eColls/eThesisdoc/Bab2/2013-1-00621-IF%20Bab2001.pdf> [Diakses 15 Desember 2015].

Budianto, A., 2012. *Pengantar Cloud Computing | Cloud Indonesia.* [online] Tersedia di: <http://cloudindonesia.or.id> [Diakses 10 Desember 2015]

Dable, N. D. & Mishra, N., 2014. Enhanced File Security using Encryption and Splitting technique over Multi-Cloud Environment. *International Journal on Advanced Computer Theory and Engineering (IJACTE),* III(4), pp. 6-10.

Dwiperdana, A., 2007. Cryptographic Hash Function Dan Penggunaannya Dalam Digital Signature. pp. 1-10.

Ferretti, L., Colajanni, M. & Marchetti, M., 2013. Transparent Access on Encrypted Data Distributed over Multiple Cloud Infrastructures. *The Fourth International Conference on Cloud Computing, GRIDs, and Virtualization,* pp. 201-206.

Google, 2015. *Google Drive API | Google Developers.* [online] Tersedia di: <https://developers.google.com/drive/web/about-sdk> [Diakses 14 Desember 2015]

Hayder, M., 2009. Design and Implementation of a File Splitter and Merger Software. *Journal of Kerbala University,* VII(4), pp. 6-20.

Horn, L., 2012. *What is Google Drive? | Gizmodo.* [online] Tersedia di: <http://gizmodo.com/5904653/what-is-google-drive> [Diakses 14 Desember 2015]

Prasetyo, A. B., 2014. *Memahami API | Jejaring.* [online] Tersedia di: <http://www.jejaring.web.id/mudahnya-memahami-application-programming-interface-api> [Diakses 28 Desember 2015]

Pratiwi, O. N., 2011. Analisis Keamanan Aplikasi Penyimpanan Data Pada Sistem Cloud Computing. *e-Indonesia Initiative 2011,* pp. 127-139.

Rackham, S., 2013. *SplitFile.* [online] Tersedia di: <http://www.methods.co.nz/splitfile> [Diakses 14 Desember 2015]

Ramadhan, Z., 2011. Aspek Keamanan Pada Cloud Computing. *Jurnal Ilmiah Abdi Ilmu,* IV(2), pp. 645-651.

Rouse, M., 2011. *What is Dropbox? | Techtarget.* [online] Tersedia di: <http://searchmobilecomputing.techtarget.com/definition/Dropbox> [Diakses 14 Desember 2015]

Uehara, M., 2013. Split File Model for Big Data in Low Throughput Storage. *International Conference on Complex, Intelligent, and Software Intensive Systems,* Volume VII, pp. 250-256.

Yao, X. & Yao, W., 2012. A Trustworthy Storage Framework on the Cloud. *Trustworthy Computing and Services,* pp. 69-77.