METADATA FORENSIK UNTUK MENDUKUNG PROSES INVESTIGASI DIGITAL

MOH. SUBLI
14917148

Tesis diajukan sebagai syarat untuk meraih gelar Magister Komputer

Konsentrasi Forensika Digital

Program Studi Megister Teknik Informatika

Program Pascasarjana Fakultas Teknologi Industri

Universitas Islam Indonesia

2017

Lembar Pengesahan Pembimbing

Metadata Forensik Untuk Mendukung Proses Investigasi Digital

Nama: Moh. Subli

NIM: 14917148

Yogyakarta, Maret 2017

Pembimbing I	Pembimbing II
Dr. Bambang Sugiantoro, MT	Yudi Prayudi, S.Si., M.Kom

Lembar Pengesahan Penguji

Metadata Forensik Untuk Mendukung Proses Investigasi Digital

Nama: Moh. Subli

NIM: 14917148

Yogyakarta, Maret 2017

Tim Penguji,

Dr. Bambang Sugiantoro, MT
Ketua

Yudi Prayudi, S.Si., M.Kom
Anggota I

Dr. Imam Riadi, M.Kom
Anggota II

Mengetahui,
Ketua Program Pascasarjana Fakultas Teknologi Industri
Universitas Islam Indonesia

Dr. R. Teduh Dirgahayu, ST., M.Sc

Pernyataaan keaslian tulisan

Dengan ini saya menyatakan bahwa tesis ini merupakan tulisan asli dari penulis, dan tidak berisi material yang telah diterbitkan sebelumnya atau tulisan dari penulis lain terkecuali referensi atas material tersebut telah disebutkan dalam tesis. Apabila ada kontribusi dari penulis lain dalam tesis ini, maka penulis lain tersebut secara eksplisit telah disebutkan dalam tesis ini.

Dengan ini saya juga menyatakan bahwa segala kontribusi dari pihak lain terhadap tesis ini, termasuk bantuan analisis statistik, desain survei, analisis data, prosedur teknis yang bersifat signifikan, dan segala bentuk aktivitas penelitian yang dipergunakan atau dilaporkan dalam tesis ini telah secara eksplisit disebutkan dalam tesis ini.

Segala bentuk hak cipta yang terdapat dalam material dokumen tesis ini berada dalam kepemilikan pemilik hak cipta masing-masing. Untuk material yang membutuhkan izin, Saya juga telah mendapatkan izin dari pemilik hak cipta untuk menggunakan material tersebut dalam tesis ini.

Yogyakarta, Maret 2017

Moh. Subli, S.ST

Publikasi selama masa studi

Subli, M., Sugiantoro., B., & Prayudi, Y. (2017). Metadata Forensik Untuk Mendukung Proses Investigasi Digital. *Jurnal Ilmiah DASI*, 18(1), 1411-3201.

Publikasi yang menjadi bagian dari tesis

Sitasi publikasi 1

Kontributor	Jenis Kontributor
Author Moh. Subli	Mendesain eksperimen (50%) Menulis *paper* (60%)
Author Bambang Sugiantoro	Mendesain eksperimen (30%) Menulis dan mengedit *paper* (20%)
Author Yudi Prayudi	Mendesain eksperimen (20%) Menulis dan mengedit *paper* (20%)

Kontribusi yang diberikan oleh pihak lain dalam tesis ini

Ada beberapa pihak terkait yang punya kontribusi dalam penyelesaian penulisan tesis ini:

- Bapak Dr. Bambang Sugiantoro, MT selaku Pembimbing I dan Bapak Yudi Prayudi, S.Si., M.Kom selaku Pembimbing II yang telah memberikan arahan-arahannya kepada penulis, sehingga penulisan tesis ini bisa selesai dengan baik dan tepat waktu.
- Bapak Ir. H. Lalu Darmawan Bakti, M.Sc., M.Kom selaku Ketua Sekolah Tinggi Manajemen Informatika Komputer (STMIK) Mataram yang telah memberikan motivasi dan biaya-biaya selama masa studi sampai dengan tahap pembuatan tesis ini.
- Ibu Dwinita Arwidiyarti, M.Kom selaku Wakil Ketua Sekolah Tinggi Manajemen Informatika Komputer (STMIK) Mataram yang telah membimbing penulis dalam menyelesaikan tesis ini.
- Bapak Karya Gunawan, M.Kom selaku Dosen di Sekolah Tinggi Manajemen Informatika Komputer (STMIK) Mataram yang telah membantu penulis dalam menyelesaikan aplikasi sistem metadata forensik yang mana ini sangat berguna sekali dalam tesis ini.

Halaman Persembahan

Khusus kupersembahkan karya ini kepada orang-orang yang tersayang dan yang telah berjasa:

- Kepada Istri dan Anakku Zahira Ulfa yang akan berumur 1 Tahun di Bulan April besok ini. Terima kasih atas kasih sayang dan semangat yang kalian berikan.
- Kepada Kedua Orangtuaku, Mertuaku dan Guru-guruku yang tak henti-hentinya Berdoa untuk Ananda dan Kakak-kakak Adik-adikku yang selalu memberikan kata-kata penyemangat.
- Khusus kepada Bapak Direktur AMIKOM-ASM Mataram yang sekarang berubah bentuk menjadi Ketua STMIK-ASM Mataram yang telah memberikan kepercayaannya kepada Saya dalam melanjutkan Studi ini, Terima kasih banyak atas biaya dan bantuan-bantuan lainnya yang diberikan, semoga Saya berguna dan dapat memberikan sumbangsih yang banyak dan bermanfaat bagi perkembangan STMIK-ASM Mataram kedepan.

Kata Pengantar

Puji syukur penyusun panjatkan kehadirat Allah SWT, karena atas berkat rahmat, taufik serta hidayahnya sehingga penyusun dapat menyelesaikan Laporan Tesis di Universitas Islam Indonesia Yogyakarta. Tidak lupa pula shalawat serta salam Saya haturkan kepada junjungan Nabi Alam Nabi Besar Muhammad SAW, Keluarga Beliau dan Sahabat-sahabat Beliau yang telah membimbing Ummat Manusia kejalan yang lurus.

Laporan Tesis yang berjudul "METADATA FORENSIK UNTUK MENDUKUNG PROSES INVESTIGASI DIGITAL" disusun guna memenuhi salah satu syarat akademik untuk menempuh kelulusan Program Pascasarjana Magister Teknik Informatika pada Fakultas Teknologi Industri Universitas Islam Indonesia Yogyakarta.

Dan ucapan terima kasih yang sebesar - besarnya kepada semua pihak yang telah banyak membantu dalam menyelesaikan laporan ini utamanya kepada yang terhormat:

1. Bapak Dr. Ir. Harsoyo, M.Sc sebagai Rektor Universitas Islam Indonesia Yogyakarta Periode sebelumnya dan Bapak Nandang Sutrisna, S.H., LL.M., M.Hum., Ph.D yang baru saja dilantik sebagai Rektor Universitas Islam Indonesia Yogyakarta.
2. Bapak Dr. R. Teduh Dirgahayu, ST., M.Sc sebagai Ketua Program Pascasarjana Fakultas Teknologi Industri Universitas Islam Indonesia Yogyakarta.
3. Bapak Yudi Prayudi, S.Si., M.Kom selaku Ketua PUSFID Fakultas Teknologi Industri Universitas Islam Indonesia Yogyakarta sekaligus Pembimbing II yang telah meluangkan banyak waktunya dalam membimbing dan mebantu penulis selama penulisan Tesis ini.
4. Bapak Dr. Bambang Sugiantoro, MT selaku Pembimbing I yang telah memberikan arahan-arahan dalam membimbing dan mebantu penulis selama penulisan Tesis ini.
5. Bapak Dr. Imam Riadi, M.Kom selaku Penguji yang telah memberikan masukan, saran dan kritikannya kepada penulis dalam tahap perbaikan-perbaikan penyusunan laporan tesis ini.
6. Segenap Dosen dan Staf Universitas Islam Indonesia Yogyakarta, khususnya Fakutas Teknologi Industri, Terima kasih atas semuanya.
7. Teman-teman semuanya yang dari Magister Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia Yogyakarta dan juga khususnya Konsentrasi Forensika Digital Angkatan X, Terima kasih banyak sudah saling pada mengingatkan.
8. Kepada Bapak Direktur AMIKOM-ASM Mataram yang sekarang berubah bentuk menjadi Ketua STMIK-ASM Mataram yang telah memberikan kepercayaannya kepada saya dalam melanjutkan Studi ini, Terima kasih banyak atas biaya dan bantuan-bantuan lainnya yang diberikan.

9. Kepada Ibu Dwinita Arwidiyarti, M.Kom dan Bapak Karya Gunawan, M.Kom yang sangat men-support dan membantu penulis dalam penyelesaian penulisan Laporan Tesis ini, Terima kasih banyak atas segala bimbingan dan bantuannya.
10. Kepada semua Teman-teman Dosen dan Staf STMIK-ASM Mataram yang selalu memberikan supportnya kepada penulis, Terima kasih banyak.

Penyusun menyadari sepenuhnya bahwa penyusunan laporan Tesis ini masih jauh dari kata sempurna sehingga masih diperlukan penyempurnaan lebih lanjut. Untuk itu saran, kritik dan tambahan yang sifatnya membangun penyusun harapkan dari pembaca untuk selanjutnya.

Harapan penyusun, semoga Laporan Tesis ini dapat bermanfaat bagi Saya pribadi pada khususnya dan untuk kita semua pada umumnya.

Yogyakarta, Maret 2017

Moh. Subli, S.ST

Daftar Isi

Daftar Tabel

Daftar Gambar

Abstrak

Metadata adalah informasi yang terstruktur yang menggambarkan, menjelaskan, menempatkan, atau membuat lebih mudah untuk mengambil, menggunakan, atau mengelola sebuah sumber informasi. Metadata sering disebut data tentang data atau informasi tentang informasi. Selama ini fokus dari analisis forensik itu lebih banyak kepada menemukan file-file yang contennya itu sesuai dengan tujuan investigasi. Cara lain yang bisa dilakukan yaitu dengan melakukan pendekatan metadata, mengapa metadata karena metadata menyimpan informasi lain dari sebuah file. Apabila ini dilakukan, maka diharapkan proses ini bisa melihat langsung metadata file secara umum dan juga dapat menemukan file-file berdasarkan korelasi file dengan parameter dari metadata file tersebut. Cara ini umumnya belum terfasilitasi oleh alat-alat forensik yang ada, sehingga perlu dilakukan penelitian untuk melihat sejauh mana kemungkinan kemanfaatan metadata untuk mendukung proses investigasi digital.

Kata Kunci

Metadata File, Korelasi File, Sistem Aplikasi Metadata Forensik

Abstract

Metadata is structured information that describes, explains, locates, or make it easier to retrieve, use or manage an information resource. Metadata is often called data about data or information about information. During this time the focus of the forensic analysis was more to find the files that contennya accordance with the purpose of the investigation. Another way to do that is by doing approach metadata, why metadata for metadata store other information from a file. If this is done, it is expected that this process can be viewed directly metadata files in general and also be able to find files based on file correlation with the parameters of the metadata file. This method is generally not facilitated by forensic tools are there, so research must be done to see the extent to which the possibilities for the benefit of metadata to support digital investigation process.

Keywords

File Metadata, File Correlation, Application System Metadata Forensics

Bab 1 Pendahuluan

1.1 Latar Belakang

Berdasarkan Undang-undang Republik Indonesia Nomor 11 Tahun 2008 bahwa informasi elektronik dan/atau dokumen elektronik dan/atau hasil cetaknya merupakan alat bukti hukum yang sah, sesuai dengan Pasal 1 Ayat 1 bahwa Informasi Elektronik adalah satu atau sekumpulan data elektronik, termasuk tetapi tidak terbatas pada tulisan, suara, gambar, peta, rancangan, foto, electronic data interchange (EDI), surat elektronik *(electronic mail*), telegram, teleks, telecopy atau sejenisnya, huruf, tanda, angka, Kode Akses, simbol, atau perforasi yang telah diolah yang memiliki arti atau dapat dipahami oleh orang yang mampu memahaminya. Sebagaimana tertuang juga dalam Penjelasan atas Undang-undang Republik Indonesia Nomor 11 Tahun 2008 tentang Informasi dan Transaksi Elektronik yaitu pembuktian merupakan faktor yang sangat penting, mengingat informasi elektronik bukan saja belum terakomodasi dalam sistem hukum acara Indonesia secara komprehensif, melainkan juga ternyata sangat rentan untuk diubah, disadap, dipalsukan, dan dikirim ke berbagai penjuru dunia dalam waktu hitungan detik. Dengan demikian, dampak yang diakibatkannya pun bisa demikian kompleks dan rumit.

Munculnya berbagai jenis pelanggaran dan tindak kejahatan pada dunia komputer khususnya tentang masalah data atau file, seperti pencurian data, penggandaan data, penghapusan data sampai dengan masalah memanipulasikan data yang semakin sering terjadi pada saat ini. Semua setuju bahwa kejahatan pada dunia komputer terus meningkat dari tahun ke tahun sejalan dengan perkembangan dan kemajuan dalam dunia informasi dan transaksi elektronik (ITE) atau dalam ilmu pengetahuan teknologi informasi dan komunikasi.

Pada saat disadari maupun tidak disadari bahwa secara tidak langsung semua aktivitas dan kegiatan yang dilakukan termasuk identitas pribadi sesorang bisa terekam pada perangkat dunia informasi dan transaksi elektronik (ITE). Sebagai dampaknya, dewasa ini muncul berbagai jenis tindak kejahatan yang dapat mengeksploitasi informasi dan transaksi elektronik (ITE) tersebut. Salah satunya adalah yang sering terekam yang bisa di jadikan sebagai laporan barang bukti dari jenis data atau file yaitu tentang metadata file.

Metadata adalah informasi yang terstruktur yang menggambarkan, menjelaskan, menempatkan, atau membuat lebih mudah untuk mengambil, menggunakan, atau mengelola

sebuah sumber informasi. Metadata sering disebut data tentang data atau informasi tentang informasi (Niso 2004). Metadata adalah informasi yang ditanam pada sebuah file yang isinya berupa penjelasan tetang file tersebut. Metadata ini mengandung informasi mengenai isi dari suatu data yang dipakai untuk keperluan manajemen file atau data itu nantinya dalam suatu basis data (Putu Laxman Pendit 2007). Jika data tersebut dalam bentuk *document* ***docx*** metadatanya berupa keterangan mengenai *name file, content created, date last saved, content type, pages, word count, character count, line count, paragraph count, size, date created, date modified, date accessed, computer* dan masih banya lagi. Jika dalam bentuk ***pdf*** metadatanya berupa *name, type, foder path, size, date created, date modified, attributes, owner* dan *computer.* Untuk jenis data gambar ***jpg***, metadata mengandung informasi mengenai siapa pemotretnya, kapan pemotretannya, dan *setting* kamera pada saat dilakukan pemotretan. Untuk audio jenis ***mp3*** bisa tambahan metadatanya berupa *album, year, genre, length, bit rate* dan rekaman yang dipakai lainnya. Untuk jenis video ***mp4*** metadatanya bisa berupa seperti mp3 dengan tambahan *frame width, frame height, date rate, total bitrate, frame rate, channels* dan jenis perekam video lainnya. Untuk file akuisisi hasil imaging yang berupa ***dd*** dan ***E01*** metadatanya adalah *Acquired using, Case Number, Evidence Number, Unique description, Examiner, Notes* dan bisa nilai checksumnya yang berupa MD5, SHA1 dan SHA256.

Selama ini fokus dari analisis forensik itu lebih banyak kepada menemukan file-file yang contennya itu sesuai dengan tujuan investigasi. Cara lain yang bisa dilakukan yaitu dengan melakukan pendekatan metadata, mengapa metadata karena metadata menyimpan informasi lain dari sebuah file. Apabila ini dilakukan, maka diharapkan proses ini bisa melihat langsung metadata file secara umum dan juga dapat menemukan file-file berdasarkan korelasi file dengan parameter dari metadata file tersebut. Cara ini umumnya belum terfasilitasi oleh alat-alat forensik yang ada, sehingga perlu dilakukan penelitian untuk melihat sejauh mana kemungkinan kemanfaatan metadata untuk mendukung proses investigasi digital.

Untuk itu, dalam penelitian ini dibuatlah sebuah sistem aplikasi untuk metadata forensik. Sistem ini dibuat untuk memudahkan dalam memahami karakteristik metadata file secara umum dan memudahkan pencarian file-file berdasarkan korelasi metadata file tersebut.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan diatas, dapat dirumuskan masalah dalam penelitian ini adalah sebagai berikut:

a. Bagaimanakah memahami karakteristik metadata file?

b. Bagaimanakah merancang sistem atau metode algoritma untuk melakukan korelasi metadata file?
c. Bagaimanakah menguji kinerja sistem atau algoritma yang dibangun untuk melakukan analisis metadata file dalam sebuah proses investigasi?

1.3 Batasan Masalah

Batasan masalah dalam penelitian ini meliputi:

a. Membaca karakteristik metadata file secara umum.
b. Metadata file yang akan dipahami yaitu tujuh macam jenis file yang ada didalam komputer yang ber-extension Docx, Pdf, Jpg, Mp3,Mp4, DD dan E01.
c. Parameter korelasi metadata file yaitu berdasarkan Tanggal (*File Date*), Ukuran (*File Size*), Ektensi (*File Type*) dan Pemilik (*File Owner*).
d. File yang dibangun bukan berdasarkan hasil imaging/akuisisi tetapi langsung dari komputer.

1.4 Tujuan Penelitian

Adapun tujuan dari penelitian ini antara lain:

a. Melakukan pembacaan metadata file untuk memahami karakteristik metadata setiap file.
b. Melakukan perancangan sistem atau metode algoritma untuk melakukan korelasi metadata file yang bisa mencari file-file di dalam komputer.
c. Melakukan pengujian kinerja sistem atau algoritma yang dibangun untuk melakukan analisis metadata file dalam sebuah proses investigasi.

1.5 Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini adalah sebagai berikut:

a. Mempermudah seorang analisa/investigator dalam memahami karakteristik metadata setiap file.
b. Mempermudah seorang analisa/investigator dalam menemukan file-file yang ada didalam komputer dengan melakukan korelasi metadata file.
c. Selain dari itu diharapkan hasil dari penelitian ini bisa dimanfaatkan oleh penyidik untuk melakukan proses investigasi tidak hanya pada satu komputer saja tetapi bisa di terapkan pada semua komputer.

1.6 Literatur Review Penelitian

Beberapa penelitian yang berhubungan dengan metadata forensik yaitu penelitian yang dilakukan oleh Usama Salama, Vijay Varadharajan, & Michael Hitchens (2012) dalam penelitian ini bahwasanya peneliti telah menyelidiki berbagai jenis metadata yang umumnya tersedia dengan benda-benda digital seperti foto dan dokumen yang tersedia di Internet dan telah menganalisis berbagai jenis informasi metadata yang dihasilkan oleh perangkat kamera dan smartphone yang digunakan untuk menangkap dan menyimpan foto digital di web. Analisis yang digunakan adalah dengan pengembangan aturan heuristik yang dapat digunakan untuk meningkatkan kualitas pengambilan keputusan dalam penyelidikan forensik.

Penelitian yang sama di lakukan juga oleh M.P. Roberts & J. Haggerty (2013) tentang peningkatan penggunaan internet untuk menyimpan data yang dipastikan bahwa tersedianya sumber daya yang berharga untuk pemeriksa forensik selama penyelidikan. Yang menarik adalah bukti yang terkait dengan penyebaran gambar tidak senonoh anak-anak yang menyebar melalui situs jejaring sosial dan forum web.

Penelitian lain juga dilakukan Sriram Raghavan & S V Raghavan (2013) Secara tradisional, sumber bukti digital yang di analisis oleh individual meneliti berbagai artefak yang terkandung di dalamnya dan menggunakan metadata artefak untuk memvalidasi keaslian urutan mereka. Namun, ketika artefak dari gambar forensik, folder, file log, dan pembuangan paket jaringan telah di analisis, pemeriksaan artefak dan metadata dalam isolasi menghadirkan tantangan yang signifikan. Idealnya, ketika sebuah sumber diperiksa, itu adalah tugas yang berharga untuk menentukan korelasi antara artefak dan kelompok artefak yang terkait.

Selanjutnya Mark Phillips (2013) menjelaskan metodologi secara keseluruhan, memperkenalkan dua *tools opensource* sederhana yang dikembangkan untuk membantu memberikan contoh perintah dalam menunjukkan beberapa permintaan analisis metadata secara umum dan Kam Woods, Alexandra Chassanoff & Christopher A. Lee (2013) fokus pada metadata yang dihasilkan oleh *tools open-source* yang mendukung Digital Forensik XML (DFXML). Bagaimana bagian-bagian dari metadata ini dapat digunakan saat merekam peristiwa PREMIS untuk menggambarkan kegiatan yang relevan dengan pelestarian dan akses dari metadata tersebut.

Volume besar metadata tersedia dalam infrastruktur database untuk keperluan penyelidikan tetapi sebagian besar usaha terletak pada pengambilan dan analisis informasi yang dari sistem komputasi. Dengan demikian, dalam penelitian ini terutama relevansi metadata dalam desain dari alat forensik database yang umum independen dari DBMS yang difokuskan untuk digunakan. (Shraddha Suratkar & Harmeet Khanuja 2014).

Begitu juga dalam lain yang mengkatakan bahwa metadata tidak terlihat saat melihat data dalam sejumlah bentuk seperti dokumen docx atau jpg. Namun demikian, pertimbangan penting dalam penemuan informasi untuk digunakan dalam investigasi forensik digital. Berbagai jenis dokumen dan file memiliki sejumlah format dan jenis metadata, yang dapat digunakan untuk menemukan sifat-sifat dari aktivitas file, dokumen atau jaringan. Selain itu, Metadata berguna dalam banyak keadaan, di mana ia dapat memberikan bukti kolaborasi antara kelompok orang, karena beberapa dari mereka tidak menyadari jenis informasi yang disimpan di dalam dokumen mereka. Dengan demikian, penyidik digital forensik dapat mengakses informasi dokumen tersembunyi ini. Dalam kasus hukum, identifikasi bukti digital yang relevan sangat penting untuk mendukung kasus, verifikasi dan pemeriksaan yang ada pada bentuk argumen hukum. Dalam penelitian ini, ditujukan bagaimana menggunakan format dan jenis metadata yang berbeda dalam memvalidasi argumen hukum untuk dijadikan bukti yang relevan (Fahad Alanazi & Andrew Jones 2015)

Ezz El-Din Hemdan & Manjaiah D.H (2015) juga dalam penelitiannya melakukan pendekatan analisis forensik untuk benda digital seperti foto digital dan dokumen. Benda-benda ini berisi metadata penting yang dapat digunakan oleh penyidik untuk membantu menyelidiki kejahatan yang berkaitan dengan *cloud*. Metadata dapat digunakan juga oleh penyerang untuk melakukan kegiatan ilegal sehingga ada kebutuhan serius untuk melindungi metadata karena memberikan peneliti dengan informasi yang dapat dipercaya untuk melakukan penyelidikan forensik. Dalam pendekatan ini, metadata yang dihasilkan dari benda-benda dan juga algoritma hash diterapkan untuk menghasilkan nilai hash untuk menjamin integritas data yang diunggah ke layanan *cloud* seperti ADrive, Box, Microsoft onedrive, Google Drive, Copy dan Dropbox.

Andy Spore (2016) mengatakan lebih banyak orang memahami peran metadata dari segi pengembangan strategi hukum dan dengan analisis forensik yang tepat, metadata dapat membantu pola sorot, penetapan waktu, dan titik kesenjangan dalam data.

Pada penelitian ini konsep yang akan diusulkan dan membedakan dengan penelitian sebelumnya adalah pada teknik model pencarian metadata setiap file pada komputer. Berikut tabel 1.1 menampilkan literatur review penelitian sebelumnya dan penelitian yang di usulkan.

Tabel 1. 1 Literatur Review Penelitian

No.	Paper Utama	Masalah	Pemecahan Masalah
1	Usama Salama, Vijay Varadharajan, & Michael Hitchens (2012)	• Ancaman terhadap privasi pengguna yang dihasilkan dari jenis metadata file umum yang diunggah ke situs jejaring sosial. • Bagaimana mudahnya bagi penyerang dalam mengekstrak informasi privasi yang signifikan dari metadata yang tertanam dalam file upload di server web publik dan risiko privasi ini untuk pengguna pada umumnya.	• Analisis dari semua metadata tertanam dalam file yang terletak di server web publik, dengan fokus khusus pada metadata foto yang diunggah.
2	M.P. Roberts & J. Haggerty (2013)	• Bukti yang terkait dengan penyebaran gambar tidak senonoh anak-anak yang menyebar melalui situs jejaring sosial dan forum Web.	• Metafor, menggunakan *Web crawler searches* untuk *metadata signatures* untuk identifikasi otomatis dari file yang berada di server web remote. • Dengan cara ini, dapat di identifikasi repositori potensi gambar ilegal atau sumber bukti yang terkait dengan kejahatan seperti memanfaatkan metadata geo-lokasi untuk mengidentifikasi gambar digital yang diambil selama kejahatan berlangsung.
3	Sriram Raghavan & S V Raghavan (2013)	• Selama ini kebutuhan analisis bukti digital dalam memastikan sifat artefak yang terkandung dan bagaimana cara menguatkan satu sama lain untuk relevansi dengan penyelidikan. Namun, pembentukan asosiasi tersebut dan penemuan korelasi yang terus masih tetap sebagian besar manual. • Sebagai heterogenitas bukti digital terus tumbuh dengan kemajuan teknologi, • kita dihadapkan dengan perangkat digital yang lebih baru, berkas yang lebih baru dan format log dari asosiasi yang harus ditemukan.	• Menyajikan tools analisis AssocGEN yang menggunakan metadata untuk menentukan hubungan antara artefak milik *files, logs* dan *network packet dumps*, dan mengidentifikasi asosiasi metadata ke grup artefak yang terkait.

Lanjutan **Tabel 1.1** Literatur Review Penelitian

No.	Paper Utama	Masalah	Pemecahan Masalah
4	Mark Phillips (2013)	• Semakin bertambahnya koleksi item perpustakkan digital UNT yang pada akhirnya ada sejumlah besar catatan metadata dan peningkatan jumlah pembuatan metadata. • Adanya kebutuhan untuk menganalisis dan melaporkan statistik catatan metadata ini.	• Mengembangkan alat dan proses baru untuk memastikan bahwa catatan metadata dalam kualitas tinggi yang digunakan di seluruh koleksi perpustakaan digital.
5	Kam Woods, Alexandra Chassanoff & Christopher A. Lee (2013)	• Bagaimana bagian-bagian dari metadata ini dapat digunakan saat merekam peristiwa PREMIS untuk menggambarkan kegiatan yang relevan dengan pelestarian dan akses dari metadata tersebut.	• *BitCurator project* untuk mengembangkan strategi *extensible* dalam mengubah dan menggabungkan metadata digital forensik ke dalam skema metadata arsip dan fokus pada metadata yang dihasilkan oleh *tools open-source* Digital Forensik XML (DFXML).
6	Shraddha Suratkar & Harmeet Khanuja (2014)	• Sejumlah pelanggaran besar keamanan database terjadi dalam tingkat yang sangat tinggi pada setiap hari dan tidak pernah bisa tahu tentang kapan kerahasiaan pengguna dikompromikan. Karena ini masalah serius, hal ini menjadi sangat penting bagi penyidik basis data tidak hanya untuk mengkonfirmasi terjadinya akses database yang tidak sah, tetapi juga untuk menghasilkan bukti kuat terhadap penjahat untuk menyajikannya bukti-bukti dalam pengadilan hukum dalam bentuk siapa, kapan, mengapa, apa, bagaimana dan di mana transaksi penipuan terjadi.	• Analisis forensik database dengan membuat beberapa salinan dari data sensitif yang ditemukan dalam artefak server database, log audit, cache, penyimpanan dan lain-lainnya.
7	Fahad Alanazi & Andrew Jones (2015)	• Bagaimana menggunakan format dan jenis metadata yang berbeda dalam memvalidasi argumen hukum untuk dijadikan bukti yang relevan.	• Dalam kasus hukum, identifikasi bukti digital relevan sangat penting untuk mendukung kasus, verifikasi dan pemeriksaan berbagai bentuk argumen hukum yang ada.

Lanjutan **Tabel 1.1** Literatur Review Penelitian

No.	Paper Utama	Masalah	Pemecahan Masalah
8	Ezz El-Din Hemdan & Manjaiah D.H (2015)	• Metadata dapat digunakan oleh penyerang untuk melakukan kegiatan ilegal sehingga ada kebutuhan serius untuk melindungi metadata karena memberikan peneliti dengan informasi yang dapat dipercaya untuk melakukan penyelidikan forensik.	• Metadata dan nilai hash disimpan dalam penyimpanan lokal untuk tujuan penyelidikan forensik karena jika ada aktivitas ilegal yang dilakukan terhadap data yang diunggah maka kasus ini akan di selidiki dengan menggunakan nilai-nilai yang disimpan dalam penyimpanan lokal untuk memeriksa integritas dari data yang diunggah ke *cloud*.
9	Andy Spore (2016)	• Bagaimana mencari barang bukti yang berupa dokumen yang telah di hapus dan membuktikan salinan di servernya dengan tanggal yang berbeda. • Bagaimana melihat peranan metadata yang lebih besar pada proses pengadilan	• Analisis komputer forensik untuk mengumpulkan data & melestarikan metadata. • Melalui proses legal dalam strategi filtering untuk mempertahankan dan menghasilkan metadata bahkan pada file yang digandakan yaitu dengan E-Discovery.
10	Usulan Penelitian	• Selama ini fokus dari analisis forensik itu lebih banyak kepada menemukan file-file yang contennya itu sesuai dengan tujuan investigasi, cara lain yaitu dengan melakukan pendekatan metadata, mengapa metadata karena metadata menyimpan informasi lain dari sebuah file. • Apabila ini dilakukan, maka diharapkan proses pencarian sebuah file dalam komputer bisa dengan menggunakan misalnya berdasarkan membaca dan korelasi antar metadata file. • Cara ini umumnya belum terfasilitasi oleh *tool-tools* forensik yang ada. Sehingga perlu dilakukan penelitian untuk melihat sejauh mana kemungkinan kemanfaatan metadata untuk mendukung proses investigasi.	• Membuat tools analisa metadata forensik dengan aplikasi java. • Tools ini berfungsi untuk membaca atau memahami karakteristik metadata file dan juga dapat digunakan untuk mencari file-file yang ada didalam komputer dengan melakuka korelasi metadata file berdasarkan parameter - parameter yang telah di *setting*.

1.7 Metode Penelitian

Dalam menyelesaikan penelitian ini perlu disusun langkah-langkah penyelesaian penelitian secara sistematika yang disebut dengan metodologi penelitian. Metodologi yang digunakan pada penelitian ini adalah sebagai berikut:

1. Melakukan Identifikasi Masalah

   Tahap awal dalam penelitian ini adalah merumuskan masalah yang akan di jadikan sebagai objek penelitian.

2. Tinjauan Pustaka

   Tinjauan pustaka dilakukan guna mencari literatur pendukung penelitian ini.

3. Metode Pengumpulan Data

   Metode yang digunakan untuk mengumpulkan data pada penelitian ini yaitu dengan melakukan studi literatur. Studi literatur dilakukan untuk mencari semua informasi yang berkaitan tentang konsep metadata forensik dalam membaca atau memahami karakteristik metadata file dan memudahkan pencarian dalam korelasi metadata file, seperti membaca buku-buku, paper atau jurnal-jurnal dan mengunjungi situs-situs yang ada di internet yang berhubungan dengan metadata forensik.

4. Analisis Kebutuhan Sistem

   Untuk mempermudah menganalisis sebuah sistem dibutuhkan dua jenis kebutuhan. Kebutuhan fungsional dan kebutuhan nonfungsional. Kebutuhan fungsional adalah kebutuhan yang berisi proses-proses apa saja yang nantinya dilakukan oleh sistem. Sedangkan kebutuhan nonfungsional adalah kebutuhan yang menitikberatkan pada properti prilaku yang dimiliki oleh sistem.

5. Perancangan Sistem

   Metode yang digunakan untuk membangun sebuah sistem atau metode algoritma metadata forensik ini yaitu dengan menggunakan metode perancangan terstruktur serta menggunakan *Workflow* (Bagan Kerja) dan *Flowchart* (Bagan Alir). Perancangan ini dimulai dari perancangan secara umum yang disebut dengan desain konseptual (*conceptuai design*) atau desain logikal (*logical design*).

6. Implementasi Sistem

   Implementasi adalah proses untuk memastikan bahwa sistem atau metode algoritma yang dibangun bebas dari kesalahan dan mudah digunakan oleh pengguna dalam hal ini seorang investigator.

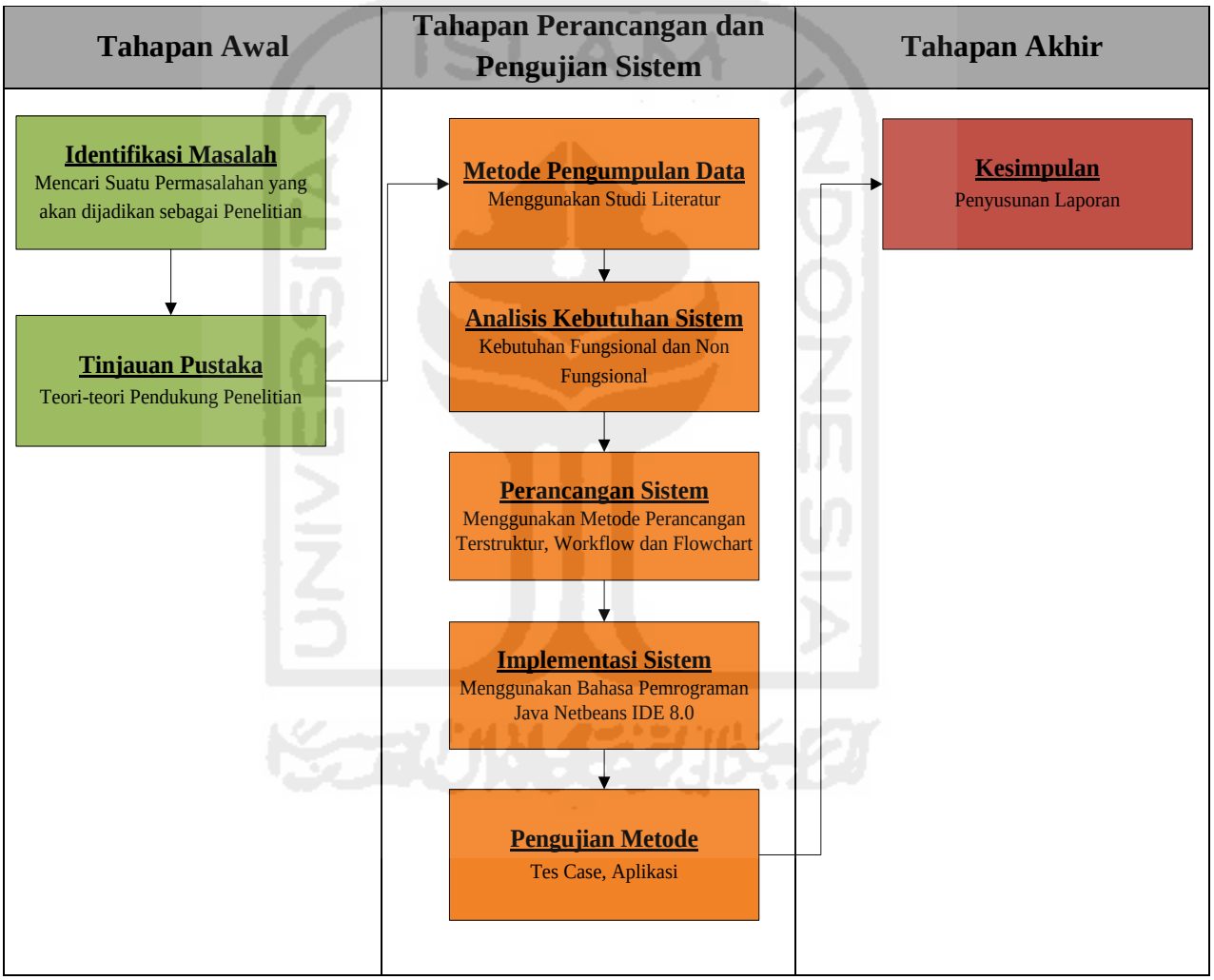
7. Pengujian Metode

Pada tahapan ini dilakukan pengujian sistem metadata forensik yang bertujuan untuk mendeteksi kegagalan perangkat lunak sehingga kesalahan sistem dapat ditemukan dan diperbaiki.

8. Kesimpulan

   Penyusunan laporan akhir penelitian ini.

Berikut gambar 1.1 menampilkan tahapan metode penelitian rancangan sistem metada forensik yang akan dibangun:



Gambar 1. 1 Tahapan Metode Penelitian Rancangan Sistem Metada Forensik

1.8 Sistematika Penulisan

Dalam penyusunan penelitian ini, sistematika penulisan terbagi dalam beberapa bab yaitu :

Bab 1 Pendahuluan

Pendahuluan merupakan pengantar terhadap permasalahan yang akan dibahas. Di dalamnya menguraikan tentang gambaran suatu penelitian yang terdiri dari latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, serta sistematika penulisan.

Bab 2 Kajian Pustaka

Bab ini menjelaskan tentang teori-teori yang digunakan untuk menjelaskan konsep dasar penelitian ini. Teori yang akan dibahas pada bagian ini merupakan teori yang berhubungan dengan analisis metadata forensik untuk proses investigasi digital.

Bab 3 Metodologi Penelitian

Bab ini membahas tentang langlah-langkah penelitian, kebutuhan perangkat keras dan perangkat lunak yang akan digunakan serta perancangan antar muka aplikasi yang akan dibuat.

Bab 4 Hasil dan Pembahasan

Hasil dan Pembahasan berisi tentang implementasi dari Metadata Forensik ke dalam aplikasi yang dibangun, juga berisikan uraian tentang hasil yang dicapai, bagaimana hasil dapat dicapai dan pembahasan mengapa hasil tersebut dapat dicapai serta pengujian terhadap sistem yang telah dibuat.

Bab 5 Kesimpulan dan Saran

Kesimpulan dan Saran, memuat kesimpulan-kesimpulan dari hasil penelitian dan saran-saran yang perlu diperhatikan berdasar keterbatasan yang ditemukan dan asumsi-asumsi yang dibuat selama melakukan penelitian dan juga rekomendasi yang dibuat untuk pengembangan penelitian selanjutnya.

Bab 2 Kajian Pustaka

2.1 Sistem Yang Sudah Ada

Metadata adalah informasi yang terstruktur yang menggambarkan, menjelaskan, menempatkan, atau membuat lebih mudah untuk mengambil, menggunakan, atau mengelola sebuah sumber informasi. Metadata sering disebut data tentang data atau informasi tentang informasi. Metadata adalah informasi yang ditanam pada sebuah file yang isinya berupa penjelasan tetang file tersebut. Metadata ini mengandung informasi mengenai isi dari suatu data yang dipakai untuk keperluan manajemen file atau data itu nantinya dalam suatu basis data.

2.2 Dasar Teori

Dalam dasar teori ini berisi uraian teori-teori yang mendasari penelitian dalam metadata forensik.

2.2.1 Definisi Metadata

Definisi metadata secara sederhana dapat diartikan sebagai data tentang data (data about data). Namun definisi tersebut masih belum lengkap karena metadata tidak sesederhana itu. Salah satu ciri dari metadata adalah data tersebut harus terstruktur. Jadi definisi yang tepat untuk menggambarkan metadata adalah data terstruktur tentang data (*structured data about data*). Definisi tersebut masih sederhana dan belum sepenuhnya menjelaskan lebih detail tentang metadata. *Task Force on Metadata CC:DA* (*committee on cataloguing: description and access*) dari ALA (*American library association*) menjelaskan secara lebih detail tentang metadata yaitu data yang terstruktur, ditandai dengan kode agar dapat diproses oleh komputer, mendeskripsikan ciri-ciri satuan-satuan pembawa informasi, dan membantu identifikasi, penemuan, penilaian dan pengelolaan satuan pembawa informasi tersebut.

Metadata adalah informasi tambahan yang menyertai dan mendeskripsikan tentang sebuah data tertentu. Misalnya, sebuah gambar memiliki metadata yang menginformasikan seberapa besar ukuran file gambar, kedalaman warnanya, resolusinya, kapan dibuat, dan sebagainya. Contoh lain, metadata sebuah dokumen teks berisi informasi tentang seberapa panjang dokumen tersebut, siapa yang membuat, kapan ditulis, dan ringkasan isinya. Adapun metadata pada halaman website

adalah bagian yang dituliskan pada tag meta di bagian header halaman web, misalnya deskripsi singkat tentang website dan keywordnya.

Metadata direkam komputer secara otomatis saat sebuah file dibuat, sehingga bisa diketahui kapan file dibuat, siapa user pembuatnya, berapa ukuran filenya, demikian juga ekstensinya. Namun demikian, metadata juga dapat disusun secara manual. Untuk mengedit dan membaca metadata sebuah file, digunakan software pengolah metadata.

2.2.2 Konsep Metadata

Metadata dapat diartikan sebagai *"data tentang data (spasial)"*, berisikan informasi mengenai karakteristik data dan memegang peran penting di dalam mekanisme pertukaran data. Melalui informasi metadata diharapkan pengguna data dapat mengintepretasikan data secara sama, bilamana pengguna melihat langsung data spasialnya. Dokumen metadata berisikan informasi yang menjelaskan karakteristik data terutama isi, kwalitas, kondisi dan cara perolehannya. Metadata dipergunakan untuk melakukan dokumentasi data spasial yang berhubungan tentang siapa, apa, kapan, dimana, dan bagaimana data spasial dipersiapkan.

2.2.3 Jenis Metadata

Metadata terbagi dalam 3 jenis:

1. Metadata Deskriptif

   Data yang dapat mengidentifikasi sumber informasi sehingga dapat digunakan untuk memperlancar proses penemuan dan seleksi. Cakupan yang ada pada data ini adalah pengarang, judul, tahun terbit, tajuk subjek atau kata kunci dan informasi lain yang proses pengisian datanya sama dengan katalog tradisional.
2. Metadata Administratif

   Data yang tidak hanya dapat mengidentifikasi sumber informasi tapi juga cara pengelolaanya. Cakupan dari data ini adalah sama dengan data deskriptif hanya saja ditambah dengan pembuat data, waktu pembuatan, tipe file, data teknis lain. Selain itu data ini juga mengandung informasi tentang hak akses, hak kekayaan intelektual, penyimpanan dan pelestarian sumber informasi.
3. Metadata Struktural

   Data yang dapat membuat antara data yang berkaitan dapat saling berhubungan satu sama lain. Secara lebih jelas, Metadata ini digunakan untuk mengetahui hubungan antara berkas fisik dan halaman, halaman dan bab dan bab dengan buku sebagai produk akhir.

2.2.4 Skema Metadata

Skema metadatan terdiri dari 3 komponen yaitu:

1. *Semantic*

   Dalam kaitannya dengan metadata, semantik dapat diartikan sebagai makna kata. Lebih jelasnya adalah kesepakatan untuk membuat istilah yang digunakan untuk mewakili suatu makna. Selain itu, terkadang juga diberi keterangaan tentang status pada istilah tersebut.

2. *Content*

   Dalam hal ini, konten bisa diartikan sebagai cara mengisi semantic. content tersebut bisa berupa peraturan untuk kriteria pengisian unsur skema atau peraturan untuk nilai-nilai unsur.

3. *Sintaksis*

   Sintaksis dalam skema metadata dapat berarti sebagai machine readible (dapat dibaca mesin) atau dengan kata lain bahasa pemrogaman. Sehingga semantic dan content yang telah dibuat dapat dibaca oleh mesin.

2.2.5 Contoh Metadata

Berikut beberapa contoh metadata berdasarkan skema metadata:

1. CDWA (*Categories for Descriptions of Works of Art*), skema untuk deskripsi karya seni
2. DCMES (*Dublin Core Metadata Element Set*), skema umum untuk deskripsi berbagai macam sumber digital.
3. EAD (*Encoded Archival Description*), skema untuk menciptakan sarana temu kembali pada bahan kearsipan (*archival finding aids*) dalam bentuk elektronik.
4. GEM (*Gateway to Educational Materials*), skema untuk bahan pendidikan dan pengajaran
5. MARC (*Machine Readable Cataloguing*), skema yang digunakan di perpustakaan sejak tahun 1960-an untuk membuat standar cantuman bibliografi elektronik.
6. METS (*Metadata Encoding and Transmission Standard*), skema metadata untuk obyek digital yang kompleks dalam koleksi perpustakaan
7. MODS (*Metadata Object Description Standard*), skema untuk deskripsi rinci sumber-sumber elektronik
8. MPEG (*Moving Pictures Experts Group*) MPEG-7 dan MPEG-21, skema untuk rekaman audio dan video dalam bentuk digital
9. ONIX (*Online Information Exchange*), skema untuk data bibliografi pada penerbit dan pedagang buku

10. TEI (*Text Encoding Initiative*), skema untuk encoding teks dalam bentuk elektronik menggunakan SGML dan XML, khususnya untuk peneliti teks di bidang humaniora.
11. VRA (*Visual Resources Association*), skema untuk deskripsi karya visual dan representasinya.

2.2.6 Peranan Metadata

Suatu studi oleh United States Geological Survey (USGS) 1994, mengatakan metadata mempunyai 3 (tiga) peranan utama, yaitu :

1. Menyediakan Informasi Bagi Katalog Data dan *Clearinghauses*
   Aplikasi SIG seringkali membutuhkan banyak tema, tetapi sedikit sekali instansi/lembaga yang mampu memenuhi semua tema data yang mereka butuhkan. Seringkali data yang dibuat oleh Instansi/lembaga juga bermanfaat bagi pengguna lain, dengan tersedianya metedata melalui katalog data dan *clearinghause,* maka pengguna lain yang membutuhkan dapat menemukan data yang mereka perlukan, selain itu juga sebagai patner untuk bersama-sama mengumpulkan data, menjaga data dan pemeliharaan data.
2. Mengatur dan Menjaga Keteraturan Pemasukan Data
   Apabila terjadi pergantian personel yang mengawaki SIG seringkali informasi tentang data, yang merupakan tanggung jawab personel tersebut mungkin akan hilang sehingga data akan kehilangan nilainya. Sedangkan personel pengganti kurang memahami isi dan penggunaan basis data digital yang ada dan mungkin mereka tidak mempercayai hasil-hasil yang dimunculkan dari data yang ditinggalkan oleh personel sebelumnya, dengan adanya metadata keteraturan pemasukan data akan terjamin.
3. Menyediakan Informasi untuk Membantu Pentransferan Data
   Metadata akan membantu instansi/lembaga menerima proses data dan menginterpretasi data, menggabungkan data ke basis data-nya dan memperbaharui katalog internal pada basis data tersebut.

2.2.7 Kegunaan dan Manfaat Metadata

Adapun kegunaan dan manfaat metadata yaitu:

1. Sebagai alat/tool pengelolaan investasi (*data*) seperti melakukan monitoring kemajuan pelaksanaan pekerjaan pembangunan data spasial, mendokumentasikan data data yang ada (selesai dikerjakan), menginformasikan data data yang dimiliki untuk dapat dimanfaatkan oleh pihak lain dan melakukan estimasi rencana kerja pengumpulan data dikemudian hari.

2. Sarana untuk menyebarluaskan kepemilikan data melalui mekanisme *clearinghouse*. Metadata merupakan faktor penting dalam konsep pemanfaatan data spasial bersama (*data sharing*).
3. Memberikan penjelasan (informasi) kepada pengguna data tentang tata cara pemrosesan dan mengintepretasikannya.
4. Metadata juga mengandung (berisikan) istilah istilah baku yang dipakai dalam kasanah data spasial. Dengan pembakuan istilah, kesalahan arti dalam penuturan data spasial dapat dihindari.

Untuk mencapai tujuan tersebut di atas, maka penyusunan metadata harus dipersiapkan dengan mempertimbangkan berbagai hal sedemikian hingga produk informasi yang dihasilkan dapat dimanfaatkan oleh berbagai pihak. Informasi metadata ditetapkan berdasarkan 4 (empat) karakteristik yang menentukan peranan dari metadata, yaitu :

1. Ketersediaan - informasi yang diperlukan untuk mengetahui ketersediaan data
2. Penggunaan - informasi yang diperlukan untuk mengetahui kegunaan data
3. Akses - informasi yang diperlukan tentang tatacara mendapatkan data
4. Transfer - informasi yang diperlukan untuk mengolah dan menggunakan data.

Pada tingkat global, terdapat beberapa tingkatan metadata yang biasa digunakan, yaitu :

1. *Discovery metadata* adalah informasi minimum yang diberikan untuk menjelaskan isi dari sumber data. Jenis metadata ini tentu saja tidak dapat memenuhi kategori metadata yang bisa diaplikasikan pada tingkat internasional.
2. *Exploration metadata* adalah informasi yang lebih detil yang diberikan dalam menjelaskan isi dari sumber data. Jenis metadata ini diharapkan dapat membantu pengguna data untuk keperluan analisis
3. *Exploitation metadata* adalah metadata yang memuat informasi akses data, transfer data, load data, menginterpretasikan data dan penggunaan data untuk suatu aplikasi.

2.3 Konsep Aplikasi Metadata Forensik

Netbeans adalah sebuah aplikasi Integrated Development Environment (IDE yang berbasiskan Java dari Sun Microsystems yang berjalan di atas swing. Swing merupakan sebuah teknologi Java untuk pengembangan aplikasi dekstop yang dapat berjalan pada berbagai macam platform seperti windows, linux, Mac OS X dan Solaris. Sebuah IDE merupakan lingkup pemrograman yang di

integrasikan ke dalam suatu aplikasi perangkat lunak yang menyediakan Graphic User Interface (GUI), suatu kode editor atau text, suatu compiler dan suatu debugger.

Netbeans juga dapat digunakan progammer untuk menulis, meng-compile, mencari kesalahan dan menyebarkan program netbeans yang ditulis dalam bahasa pemrograman java namun selain itu dapat juga mendukung bahasa pemrograman lainnya dan program ini pun bebas untuk digunakan dan untuk membuat professional dekstop, enterprise, web, and mobile applications dengan Java language, C/C++, dan bahkan dynamic languages seperti PHP, JavaScript, Groovy, dan Ruby.

NetBeans merupakan sebuah proyek kode terbuka yang sukses dengan pengguna yang sangat luas, komunitas yang terus tumbuh, dan memiliki hampir 100 mitra (dan terus bertambah!). Sun Microsystems mendirikan proyek kode terbuka NetBeans pada bulan Juni 2000 dan terus menjadi sponsor utama. Dan saat ini pun netbeans memiliki 2 produk yaitu Platform Netbeans dan Netbeans IDE. Platform Netbeans merupakan framework yang dapat digunakan kembali (reusable) untuk menyederhanakan pengembangan aplikasi deskto dan Platform NetBeans juga menawarkan layanan-layanan yang umum bagi aplikasi dekstop, mengijinkan pengembang untuk fokus ke logika yang spesifik terhadap aplikasi.

Fitur fitur yang terdapat dalam netbeans antara lain:

1. *Smart Code Completion*: untuk mengusulkan nama variabel dari suatu tipe, melengkapi keyword dan mengusulkan tipe parameter dari sebuah method.
2. Bookmarking: fitur yang digunakan untuk menandai baris yang suatu saat hendak kita modifikasi.
3. *Go to commands*: fitur yang digunakan untuk jump ke deklarasi variabel, source code atau file yang ada pada project yang sama.
4. *Code generator*: jika kita menggunakan fitur ini kita dapat meng-generate constructor, setter and getter method dan yang lainnya.
5. *Error stripe*: fitur yang akan menandai baris yang eror dengan memberi highlight merah.

2.4 Konsep Analisis Metadata Forensik

2.4.1 Pengertian file

File merupakan data yang ada pada komputer. Setiap data yang ada pada komputer dapat dikategorikan sebagai file. File tidak hanya terbatas pada data-data tertentu saja. Setiap data baik itu data gambar, data angka, data kata, data video, data suara, data aplikasi, dan data-data lainnya merupakan sebuah file.

File adalah kumpulan berbagai informasi yang berhubungan dan juga tersimpan di dalam *secondary storage*, secara konsep file memiliki beberapa tipe ada yang bertipe data terdiri dari *numeric*, *character* dan *binary*, lalu ada juga file yang bertipe program atau definisi file adalah arsip ataupun data yang tersimpan di dalam komputer.

File di komputer pada umumnya disimpan di dalam suatu folder tertentu tergantung dari pemilik komputer tersebut yang ingin dimana tempat menyimpannya, setiap file memiliki ekstensi masing-masing tergantung jenis file itu sendiri. Ekstensi file adalah sebagai tanda yang membedakan jenis-jenis dari file.

Pengertian file menurut beberapa ahli, yaitu sebagai berikut:

1. Menurut Hendrayudi "File adalah data-data yang tersimpan dalam media yang mempunyai informasi besar file, tanggal & jam penyimpanan file, nama file, ciri file (ciri aplikasi yang membuat), & attribut file."
2. Lalu menurut Rachmad Hakim S. "File merupakan dokumen yang mengandung informasi tertentu & dapat dibuka dengan program."
3. Sindhunata "File adalah kumpulan catatan atau arsip."
4. Terus menurut Mcleod (PEARSON) "File adalah koleksi record yang saling berhubungan, seperti satu file dari seluruh record yang berisi field kode-kode mata kuliah & namanya."
5. Sedangkan menurut Edi S. Mulyanta "File merupakan urutan data yang digunakan untuk melakukan encode informasi digital untuk urusan penyimpanan & pertukaran data."

2.4.2 Jenis-jenis File di Komputer

Saat Anda mengklik kanan pad file dan memilih Properties, pada file di komputer pasti ada tulisan tiga huruf sesudah titik. Itulah yang dinamakan ekstensi file. Fungsinya adalah untuk mengetahui atau membedakan jenis file. Untuk mengetahui ekstensi file lainnya Anda bisa membuka *Windows Explorer,* lalu pilih menu *View – Folder Options.*Pindah ke tab *Files Types*. Di sana terdapat puluhan dan mungkin ratusan ekstensi file. Semakin banyak Anda menginstall aplikasi maka daftar ekstensi file yang ada akan semakin panjang. Di antara beberapa ekstensi file itu adalah sebagai berikut:

Tabel 2. 1 Ektensi File

Ekstensi	Jenis	Aplikasi yang digunakan
asm	Source code pemrograman Assembly	Sembarang teks editor, seperti MS Word, NotePad, Wordpad
bat	Teks	MS Word, Notepad, WordPad, Edit (pada DOS prompt)
bmp	Image	Sembarang image editor, seperti PhotoShop, Photo Paint, Paint, dan lain-lain.
cdr	Corel Draw	Corel Draw
doc	Docoument MS Word	MS Word
exe	Aplikasi	Merupakan file aplikasi
fon	File font	Font Viewer
htm, html, shtml	Internet Document	Netscape Navigator, IE Mozilla Firefox
gif	Image, animasi	Image editor. Sedangkan untuk mebuat animasinya gunakan Ulead Gif Animator, Gif Construction Set, dll.
ico	File icon	Microangelo
jpg/jpeg	Image	Image editor, seperti PhotoShop, PhotoPaint, Paint.
log	File log	Sembarang teks editor
mp3	Audio	WinAmp
pas	Source code bahasa pemrograman pascal/Delphi	Sembarang teks editor
pdf (portable document format)	Adobe Acrobat Reader	
psd	Image	Adobe PhotoShop
reg	File registry	Regedit, untuk mengedit gunakan sembarang teks editor
ttf	File font	Font Viewer
txt	Teks	Sembarang teks editor
zip	File kompresi	WinZip, WinRar

Menampilkan ekstensi file Ekstensi file dapat dikatakan sebagai pengenal jenis file. Misalnya .doc untuk file MS Word, .cdr untuk Corel Draw, dan sebagainya. Secara default ekstensi file tesebut tidak akan kelihatan. Tetapi Anda bisa menampilkan ekstensi file tersebut sehingga Anda bisa lebih memahami tentang ekstensi file pada Windows Anda. Caranya sebagai berikut:

1. Dari *Windows Explorer*, klik menu *View*, pilih *Folder Options*.
2. Hilangkan tanda *check* (centang) pada bagian *Hide file ekstention for known file types* untuk menyembunyikan ekstensi file.

3. Sebaliknya beri tanda *check* untuk menyembunyikan ekstensi file dan menampilkan ekstensi file hanya untuk file yang tidak diketahui jenisnya.

2.4.3 Ekstensi File pada Windows

Tidak ada satu file pun yang tidak memiliki ekstensi, termasuk file program. File program dikenali dengan ekstensi .exe, namun ada kalanya sebuah file ekstensinya tidak dikenali Windows. Sehingga tidak bisa dijalankan menggunakan program yang sudah tersedia. File-file seperti ini biasanya dikenali dengan icon-nya berupa logo Windows saat ditampilkan oleh Windows Explorer.

Hal seperti ini sering terjadi karena tidak ada program yang terinstal dalam Windows untuk mengolah file tersebut. Misalnya, Anda meletakkan sebuah file berekstensi .pmd atau .p65 pada sebuah komputer Windows yang tidak memiliki program Adobe Page-Maker, maka file tersebut tidak akan bisa dibuka oleh Windows.

Meskipun demikian, ada juga beberapa program dasar dalam Windows yang bisa digunakan untuk membuka file-file yang tidak dikenali. Misalnya, WordPad yang bisa digunakan untuk membuka hampir semua tipe file dokumen, atau ada juga Notepad, sebuah aplikasi text editor sederhana untuk mengubah beberapa file sistem.

2.4.4 Format Penamaan Ekstensi

Sebuah ekstensi file diberikan berdasarkan jenis file yang bersangkutan. Misalnya untuk file program, ekstensinya adalah .exe. Nama ektensi file ini berupa singkatan tiga abjad jenis filenya. Misalnya ekstensi .exe berasal dari jenis file EXEcutable. Demikian pula dengan file berektensi .doc yang diambil dari jenis file DOCument. Ada pula ekstensi file yang berasal dari singkatan, namun tetap menunjukkan jenis file-nya. Misalnya .bmp dari bitmap, .rtf dari rich text format dan seterusnya.

Ada cara mudah untuk melihat ekstensi file-file Anda saat dilihat di Windows Explorer. Pertama, buka Windows Explorer, pilih menu View | Folder Options untuk Windows versi 98 ke bawah, atau pilih menu Tools | Folder Option untuk Windows XP. Akan muncul sebuah kotak dialog, buka tab View dan hilangkan tanda centang pada *Option Hide file extentions for known file types*. Setelah mengklik OK dan kembali pada Windows Explorer, maka setiap file yang tampak akan disertai dengan ekstensinya masing-masing.

Penentuan ekstensi sebuah file bisa Anda lakukan saat menyimpan file tersebut. Misalnya pada Microsoft Office, Anda bisa menentukan ekstensi file yang Anda buat dengan menekan

tombol scroll bawah pada kolom File Type dalam kotak dialog Save As, setelah memilih menu File | Save As.

Perubahan secara manual sebuah file juga bisa dilakukan, yakni setelah mengatur supaya Windows menampilkan ekstensi masing-masing file melalui Folder Options. Sorot sebuah file dan tekan tombol F2 atau klik kanan file tersebut dan pilih Rename. Beri nama baru jika perlu dan ubah semua, termasuk ekstensinya.

2.4.5 File Audio

Format MP3 memang beken dalam peredaran musik di internet, baik melalui aplikasi peer-to-peer atau pun layanan download. Walau begitu, itu bukan berarti file audio dengan format lain tidak ada.

.mp3 : Ekstensi seperti ini memang penanda untuk file MPEG-3. Aplikasi untuk memutar file jenis ini tak terhitung jumlahnya. Kebanyakan gratis pula. Namun, dengan aplikasi standar yang ada di Windows, yaitu Windows Media Player, Anda tetap bisa menikmati koleksi MP3 Anda.

.wav : Suara-suara untuk notifikasi sistem di Windows masih menggunakan format audio seperti ini. Berbeda dengan MP3, format ini cenderung gemuk. Anda bisa memutarnya dengan Windows Media Player.

.aif : CD audio komersial umumnya menggunakan format Audio Interchange File (aif), salah satu format audio digital dengan kualitas terbaik. Wajar kalau ukuran file dalam format ini rata-rata cukup besar. Untuk lagu berdurasi tiga menit saja, file .aif bisa memakan ruang hard disk sebesar 30 sampai 50 MB.

.ogg : Ogg Vorbis, begitulah nama lengkap format audio ini. Sebagian orang melihat kualitas format ini sedikit lebih baik dibanding MP3. Putarlah dengan Winamp versi terbaru, karena aplikasi tersebut sudah memiliki plug-in untuk membacanya.

.wma : Kependekan dari Windows Media Audio. Merupakan format asli file audio Windows Media Player yang dikembangkan oleh Microsoft. Rata-rata ukuran files berformat seperti ini lebih kecil dibanding format audio lainnya. Anda bisa memutarnya langsung dengan memilih [Windows Media Player] dari menu "*Open With*".

2.4.6 File video

Untuk komputer-komputer zaman sekarang, memutar files video berdurasi panjang adalah hal yang gampang. DalamWindows, ada Windows Media Player yang bisa memutar hampir semua

format video. Jika belum cukup, manfaatkan codec pembantu untuk memutar tipe file video lainnya.

.avi : Windows Media Player bisa memutar format ini, namun Anda perlu menginstal codec tambahan seperti Ace DivX Player yang bisa di-download gratis di http://www.soft32.com/download_4744.html.

.asx : Tipe seperti ini berisi link untuk video streaming yang diputar melalui jaringan internet. Gunakan Windows Media Player untuk memutarnya. File ini sendiri tidak benar-benar berisi data video di dalamnya.

.asf : Di dalamnya berisi data audio dan video. Anda bisa memutar files berekstensi ini dengan Windows Media Player.

.rm : Meski tidak bisa disimpan ke dalam PC, namun file Real Movie yang berekstensi .rm ini bisa dijalankan secara streaming via internet. Untuk memutar file ini Anda perlu aplikasi RealOne Player yang bisa diambil di http://www.realnetworks.com.

.mpeg dan **.mpg** : Format standar ini digunakan oleh kepingan DVD komersial. Karena ukurannya yang kecil, MPEG digunakan untuk mendistribusikan video lewat internet. Anda memerlukan codec agar bisa memutarnya di Windows Media Player.

.mov : Gunakan QuickTime Player, pemutar video buatan Apple, untuk menjalankan video berformat .mov. Kunjungi situs http://www.apple.com untuk men-download-nya.

File sistem beberapa tipe files akan langsung berjalan saat anda mengeklik-dobel file tersebut. Memang tidak bisa dijalankan dengan Microsoft Word dan Windows Media Player, karena file seperti di bawah ini merupakan file program atau sistem.

.exe : Ekstensi ini akan Anda temukan pada executable file. Merupakan file utama sebuah program dan berjalan secara otomatis saat Anda mengeklik-dobel.

.bat : Biasanya berisi sejumlah perintah yang dirancang untuk melakukan proses. Misalnya menyalin hard disk. File ini juga akan berjalan secara otomatis ketika diklik-dobel. Anda bisa menggunakan Notepad untuk membuat dan mengedit file ini.

.vbs : Ini adalah ekstensi dari file hasil olahan Visual Basic, aplikasi pemrograman yang dikembangkan Microsoft untuk membantu para programer membuat aplikasi berplatform Windows. Langsung saja klik-dobel untuk menjalankannya. Namun, hati-hati, virus komputer sering kali menggunakan format ini. Jika Anda menemukan e-mail dengan attachment berisi file berkestensi .vbs, hapus segera file tersebut. Sudah pasti itu virus.

2.4.7 Keterangan lain dari berbagai Extensi

PSD (Photoshop Document) Format file ini merupakan format asli dokumen Adobe Photoshop. *Format ini mampu menyimpan informasi layer dan alpha channel yang terdapat pada sebuah gambar, sehingga suatu saat dapat dibuka dan diedit kembali. Format ini juga mampu menyimpan gambar dalam beberapa mode warna yang disediakan Photoshop. Anda dapat menyimpan dengan format file ini jika ingin mengeditnya kembali.

BMP (Bitmap Image) Format file ini merupakan format grafis yang *fleksibel untuk platform Windows sehingga dapat dibaca oleh program grafis manapun. Format ini mampu menyimpan informasi dengan kualitas tingkat 1 bit samapi 24 bit. *Kelemahan format file ini adalah tidak mampu menyimpan alpha channel serta ada kendala dalam pertukaran platform. Untuk membuat sebuah objek sebagai desktop wallpaper, simpanlah dokumen Anda dengan format file ini. Anda dapat mengkompres format file ini dengan kompresi RLE. Format file ini mampu menyimpan gambar dalam mode warna RGB, Grayscale, Indexed Color, dan Bitmap.

EPS (Encapsuled Postcript) Format file ini merupakan format yang sering digunakan untuk keperluan pertukaran dokumen antar program grafis. Selain itu, format file ini sering pula digunakan ketika ingin mencetak gambar. *Keunggulan format file ini menggunakan bahasa postscript sehingga format file ini dikenali oleh hampir semua program persiapan cetak. *Kelemahan format file ini adalah tidak mampu menyimpan alpha channel, sehingga banyak pengguna Adobe Photoshop menggunakan format file ini ketika gambar yang dikerjakan sudah final. Format file ini mampu menyimpan gambar dengan mode warna RGB, CMYK, Lab, Duotone, Grayscale, Indexed Color, serta Bitmap. Selain itu format file ini juga mampu menyimpan clipping path.

JPG/JPEG (Joint Photographic Expert Group) Format file ini *mampu mengkompres objek dengan tingkat kualitas sesuai dengan pilihan yang disediakan. Format file sering dimanfaatkan untuk menyimpan gambar yang akan digunakan untuk keperluan halaman web, multimedia, dan publikasi elektronik lainnya. Format file ini mampu menyimpan gambar dengan mode warna RGB, CMYK, dan Grayscale. Format file ini juga mampu menyimpan alpha channel, namun karena orientasinya ke publikasi elektronik maka format ini berukuran relatif lebih kecil dibandingkan dengan format file lainnya.

GIF (Graphic Interchange Format) Format file ini *hanya mampu menyimpan dalam 8 bit (hanya mendukung mode warna Grayscale, Bitmap dan Indexed Color). Format file ini merupakan format standar untuk publikasi elektronik dan internet. *Format file mampu menyimpan animasi

dua dimensi yang akan dipublikasikan pada internet, desain halaman web dan publikasi elektronik. Format file ini mampu mengkompres dengan ukuran kecil menggunakan kompresi LZW.

TIF (Tagged Image Format File) Format file ini *mampu menyimpan gambar dengan kualitas hingga 32 bit. Format file ini juga dapat digunakan untuk keperluan pertukaran antar platform (PC, Machintosh, dan Silicon Graphic). Format file ini merupakan salah satu format yang dipilih dan sangat disukai oleh para pengguna komputer grafis terutama yang berorientasi pada publikasi (cetak). Hampir semua program yang mampu membaca format file bitmap juga mampu membaca format file TIF.

PCX Format file ini dikembangkan oleh perusahaan bernama Zoft Cooperation. Format file ini merupakan format yang *fleksibel karena hampir semua program dalam PC mampu membaca gambar dengan format file ini. Format file ini mampu menyimpan informasi bit depth sebesar 1 hingga 24 bit namun tidak mampu menyimpan alpha channel. Format file ini mampu menyimpan gambar dengan mode warna RGB, Grayscale, Bitmpa dan Indexed Color.

PDF (Portable Document Format) Format file ini digunakan oleh Adobe Acrobat, dan dapat digunakan oleh grafik berbasis pixel maupun vektor. *Format file ini mampu menyimpan gambar dengan mode warna RGB, CMYK, Indexed Color, Lab Color, Grayscale dan Bitmap. *Format file ini tidak mampu menyimpan alpha channel. Format file ini sering menggunakan kompresi JPG dan ZIP, kecuali untuk mode warna Bitmap yaitu menggunakan CCIT.

PNG (Portable Network Graphic) Format file ini *berfungsi sebagai alternatif lain dari format file GIF. Format file ini digunakan untuk menampilkan objek dalam halaman web. *Kelebihan dari format file ini dibandingkan dengan GIF adalah kemampuannya menyimpan file dalam bit depth hingga 24 bit serta mampu menghasilkan latar belakang (background) yang transparan dengan pinggiran yang halus. Format file ini mampu menyimpan alpha channel.

PIC (Pict) *Format file ini merupakan standar dalam aplikasi grafis dalam Macintosh dan program pengolah teks dengan kualitas menengah untuk transfer dokumen antar aplikasi. Format file ini mampu menyimpan gambar dengan mode warna RGB dengan 1 alpha channel serta Indexed Color, Grayscale dan Bitmap tanpa alpha channel. Format file ini juga menyediakan pilihan bit antara 16 dan 32 bit dalam mode warna RGB.

TGA (Targa) file ini didesain untuk platform yang menggunakan Targa True Vision Video Board. *Format file ini mampu menyimpan gambar dengan mode warna RGB dalam 32 bit serta 1 alpha channel, juga Grayscale, Indexed Color, dan RGB dalam 16 atau 24 bit tanpa alpha channel. *Format file ini berguna untuk menyimpan dokumen dari hasil render dari program animasi dengan hasil output berupa sequence seperti 3D Studio Max.

IFF (Interchange File Format) Format file ini *umumnya digunakan untuk bekerja dengan Video Toaster dan proses pertukaran dokumentasi dari dan ke Comodore Amiga System. Format file ini dikenali hampir semua program grafis yang terdapat dalam PC serta mampu menyimpan gambar dengan mode warna Bitmap. *Format file ini tidak mampu menyimpan alpha channel.

SCT (Scitex Continous Tone) Format file ini digunakan *untuk menyimpan dokumen dengan kualitas tinggi pada komputer Scitex. Format file ini mampu menyimpan gambar dengan mode warna RGB, CMYK, dan Grayscale *namun tidak mampu menyimpan alpha channel.

PXR (Pixar) Format file ini *khusus untuk pertukaran dokumen dengan Pixar Image Computer. Format file ini mampu menyimpan gambar dengan mode warna RGB dan Grayscale dengan 1 alpha channel.

RAW Format file ini merupakan format file yang fleksibel untuk pertukaran dokumen antar aplikasi dan platform. Format file ini mampu menyimpan mode warna RGB, CMYK, dan Grayscale dengan 1 alpha channel serta mode warna Multichannel, Lab Color dan Duotone tanpa alpha channel.

DCS (Dekstop Color Separation) Format file ini dikembangkan oleh Quark dan merupakan format standar untuk .eps. Format ini mampu menyimpan gambar dengan mode warna Multichannel dan CMYK dengan 1 alpha channel dan banyak spot channel. Format file ini mampu menyimpan clipping path dan sering digunakan untuk proses percetakan (publishing). Ketika menyimpan file dalam format ini maka yang akan tersimpan adalah 4 channel dari gambar tersebut dan 1 channel preview.

2.4.8 Format Kompresi

Beberapa program terutama yang berorientasi pada publikasi elektronik dan multimedia selalu memerlukan format file yang berukuran kecil agar ketika dibuka tidak akan lambat. Untuk keperluan tersebut diperlukan kompresi. Berikut ini format file yang berorientasi publikasi elektronik dan multimedia dengan kompresinya masing-masing.

RLE (Run Length Encoding) Kompresi ini mampu mengkompres file tanpa menghilangkan detail. Digunakan oleh Adobe Photoshop, TIFF dan sebagian besar program yang terdapat dalam Windows.

LZW (Lemple-Zif-Welf) Sama seperti kompresi RLE, kompresi ini juga mampu mengkompres file tanpa menghilangkan detail. Kompresi ini digunakan oleh TIFF, PDF, GIF, dan format yang mendukung bahasa postscript. Kompresi ini sangat baik untuk mengkompres gambar dengan area besar yang menggunakan 1 warna.

JPG (Joint Photographic Experts Group) Format ini mengkompres file dengan menghilangkan detail. Format file ini sering digunakan oleh JPG, PDF, dan format yang menggunakan bahasa postscript. Kompresi ini sangat baik digunakan untuk gambar dengan continous tone seperti foto. CCIT CCIT merupakan singkatan dari bahasa Perancis yang dalam bahasa Inggris disebut International Telegraph and Telekeyed Consultive Commitee. Kompresi ini digunakan untuk mengkompres gambar hitam putih, dan mampu mengkompres file tanpa menghilangkan detailnya. Kompresi ini sering digunakan oleh PDF dan format lain yang menggunakan bahasa postscript.

2.4.9 Format / Extensi Gambar

Pada umumnya enxtensi gambar terbagi menjadi beberapa bagian yaitu **.gif**, **.jpg / .jpeg**, **.png**, **.bmp**.

Graphics Interchange Format (GIF) merupakan salah satu format gambar yang banyak digunakan. Beberapa karakteristik format gambar GIF:

1. Mampu menayangkan maksimum sebanyak 256 warna karena format GIF menggunakan 8-bit untuk setiap pixel-nya
2. Mengkompresi gambar dengan sifat *lossless*
3. Mendukung warna transparan dan animasi sederhana.

Joint Photographic Experts Group (JPEG) adalah format gambar yang banyak digunakan untuk menyimpan gambar-gambar dengan ukuran lebih kecil. Beberapa karakteristik gambar JPEG:

1. Memiliki ekstensi .jpg atau .jpeg.
2. Mampu menayangkan warna dengan kedalaman 24-bit true color.
3. Mengkompresi gambar dengan sifat *lossy*.
4. Umumnya digunakan untuk menyimpan gambar-gambar hasil foto.
5. Penayangan citra secara progresif (*progressive display*)

Selain itu, citra dengan format PNG mempunyai faktor kompresi yang lebih baik dibandingkan dengan GIF (5%-25% lebih baik dibanding format GIF). Satu fasilitas dari GIF yang tidak terdapat pada PNG format adalah dukungan terhadap penyimpanan multi-citra untuk keperluan animasi. Untuk keperluan pengolahan citra, meskipun format PNG bisa dijadikan alternatif selama proses pengolahan citra – karena format ini selain tidak menghilangkan bagian

dari citra yang sedang diolah (sehingga penyimpanan berulang ulang dari citra tidak akan menurunkan kualitas citra) namun format JPEG masih menjadi pilihan yang lebih baik.

Bitmap (BMP) adalah representasi dari citra grafis yang terdiri dari susunan titik yang tersimpan di memori komputer. Dikembangkan oleh Microsoft dan nilai setiap titik diawali oleh satu bit data untuk gambar hitam putih, atau lebih bagi gambar berwarna. Ukuran sebenarnya untuk n-bit (2n warna) bitmap dalam byte dapat dihitung:

Berikut adalah tabel beberapa ekstensi suatu file yang sering kita temui beserta penjelasannya :

Tabel 2. 2 Macam-macam Ektensi File

Huruf	Keterangan
A	ASM = Source Code Pemrograman Assembly dibuka dengan Notepad, Wordpad, dsb AVI = File Video (format DVD) dibuka dengan CyberLink Power DVD,dsb
B	BAT = File Batch Sebuah file text yang berisi beberapa perintah yang secara segaja untuk di eksekusi oleh command prompt. dapat dibuka dengan Notepad.
C	CDR = File Corel Draw Dapat dibuka dengan Corel Draw. CMD = File command, dibuka dengan applikasi command promt
D	DAT = File video, dibuka dengan Windows mwdia player, dll DOC = File Dokumen Dapat dibuka dengan MS Word 2003 atau versi sebelumnya DOCX = File Dokumen Dapat dibuka dengan MS Word 2007 atau versi lebih baru
E	EXE = File aplikasi (executable) Hanya bisa dibuka di sistem operasi windows
F	FLV = File Flash Video Dapat dibuka dengan aplikasi video flash seperti Total Video Player, FLV Player, Windows Media Player Classic. dsb
G	GIF = File Gambar/Animasi Dapat dibuka dengan semua aplikasi untuk edit gambar seperti PhotoShop, PhotoPaint, Paint, ACDSee, Ulead Gif Animator, dll
H	HTM/HTML/SHTML = File Internet Document dibuka dengan Netscape Navigator, MS Internet Explorer, Mozilla Firefox, dll
I	ICO = File untuk Icon Gambar dibuka dengan semua aplikasi untuk edit gambar icon.seperti ACD SEE, dll ISO = File image / virtual ISO, dibuka dengan Magic iso
J	JPG/JPEG/BMP = File Gambar dibuka dengan semua aplikasi untuk edit gambar seperti PhotoShop, PhotoPaint, Paint, ACDSee dan lain2
L	LOG = File Log Dapat dengan aplikasi seperti Notepad, Wordpad, dll Lrc = File lyric, dibuka dengan mini lyric, atau note pad untuk editing
M	MP3 = File Audio dibuka dengan Winamp, Windows Media Player, dll MPG/MPEG = File Video Dapat dibuka dengan Windows Media , dll
N	NRG – File image / virtual dibuka dengan NERO
P	PDF = File Dokumen dari Adobe dibuka Adobe Acrobat Reader.dll PPT = File MS Power Point 2003 / versi sebelumnya dibuka dengan MS Power Point PPTX = File MS Power Point 2007 / versi di atasnya dibuka dengan

Lanjutan **Tabel 2.2** Macam-macam Ektensi File

Huruf	Keterangan
	MS Power Point 2007 atau versi lebih baru. PSD = File Image, File Photoshop Dapat dibuka dengan aplikasi Adobe PhotoShop.
R	RAR = File Kompresi Dapat dibuka WinZip atau WinRar. REG = File Registry Dapat dibuka dengan aplikasi Regedit atau Notepad juga bisa.
S	SRT = File subtittle / penerjemah di video, di jalankan bersama file video, file SRT harus di simpan di folder dimana video berada
T	TTF = File Font Dapat dibuka dengan aplikasi Font Viewer. TXT = File Teks Dapat dibuka dengan Notepad, Wordpad atau yang lainnya.
V	VOB = File Video DVD, dibuka dengan Cyberlink Power DVD, dll
W	WAV = File audio dibuka dengan Winamp, Windows Media Player, dll WPS = File MS Word 2007 atau versi di atasnya dibuka dengan MS 2007 atau versi di atasnya
X	XLR = File MS EXCEL Versi 2007 atau versi lebih baru, dibuka dengan MS EXCEL 2007 atau veri terbaru XLS = File MS EXCEL Versi 2003 atau sebelumnya dibuka dengan MS EXCEL XLSX = File MS EXCEL Versi 2007 atau versi lebih baru, dibuka dengan MS EXCEL 2007 atau versi terbaru
Z	ZIP = File kompresi Dapat dibuka WinZip atau WinRar.

2.4.10 Atribut File

Atribut file adalah dimana sebuah file atau directori bisa eksis atau sebuah file mempunyai atribut yang berbeda antara sistem operasi satu dengan yang lainnya. Atribut file digunakan oleh sistem operasi untuk memisahkan tipe-tipe file. Setiap file di windows mempunyai atribut tersendiri sesuai dengan kepentingan file tersebut terhadap sistem. Misalnya, tipe file yang tidak boleh dihapus oleh user diberi attribut system, jadi ketika file manager *(windows explorer*) akan menampilkannya, file beratribut sistem tersebut tidak akan ditampilkan.

Beberapa atribut file :

1. Tipe File : Menentukan Tipe File
2. Jumlah link : Jumlah link untuk file tersebut.
3. Pemilik (Owner) : Menentukan siapa pemilik file tersebut.
4. Group : Menentukan group yang memiliki file tersebut.
5. Jumlah Karakter : Menentukan ukuran file dalam byte.
6. Waktu Pembuatan : Menentukan kapan file terakhir dimodifikasi.
7. Nama File : Menentukan nama file yang dimaksud.

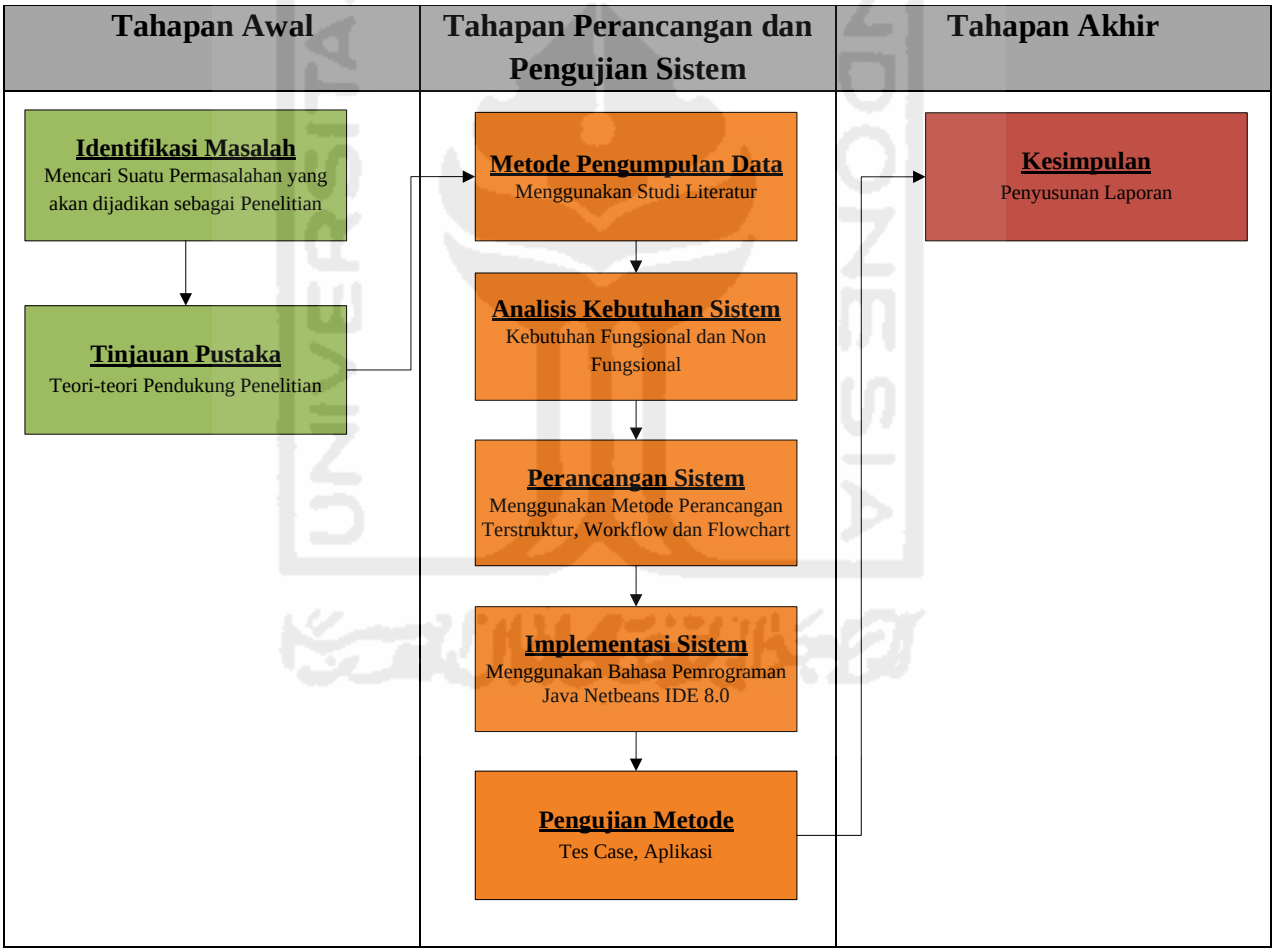
2.4.11 Atribut File Pada Windows

Beberapa atribut file pada windows yaitu:

1. *Archive*: File dengan atribut *archive* akan ditampilkan seperti file dengan atribut lain. Atribut ini berguna sebagai penanda bahwa file ini merupakan *back up copy* dari file yang asli.
2. *Hidden*: File dengan atribut *hidden* akah disembunyikan atau tidak ditampakkan secara langsung pada *windows explorer*. Kita harus merubah atributnya atau menyetel *folder options* untuk menampilkan file tersembunyi pada *windows explorer*. File dengan atribut *hidden* tetap dapat dibaca dan dirubah isinya (*Read-Write*).
3. *Read Only:* File dengan atribut *read only* hanya dapat dibaca namun tidak dapat dirubah isi filenya sebelum kita menghilangkan atribut *read only* pada file tersebut. Atribut ini berguna untuk melindungi keaslian file agar tidak dapat dimodifikasi.
4. *System*: File dengan atribut *system* akan disembunyikan dengan prioritas yang lebih diutamakan daripada file dengan atribut *hidden*. File dengan atribut system dianggap sebagai file yang dibutuhkan untuk kelangsungan berjalannya sistem pada windows sehingga keberadaannya harus dilindungi atau disembunyikan dengan aman agar tidak terhapus oleh user. File dengan atribut sistem sering juga disebut dengan super *hidden*.

Bab 3 Metodologi Penelitian

Metodologi penelitian berisi langkah-langkah yang digunakan dalam penelitian agar terstruktur dengan baik. Dengan sistematika ini proses penelitian dapat dipahami dan digunakan dengan mudah oleh pengguna. Penelitian yang dilakukan untuk merancang sebuah sistem atau metode algoritma yang diperoleh dari pengamatan data-data yang ada. Berikut adalah gambaran tahapan metode penelitian rancangan sistem metada forensik yang akan dibangun:



Gambar 3. 1 Metodologi Penelitian Rancangan Sistem Metada Forensik

Kerangka pemikiran metode penelitian yang di bangun berdasarkan gambar 3.1 diatas secara garis besar dibagi menjadi tiga tahapan yaitu tahapan pertama atau tahapan awal dimulai dari identifikasi masalah dan tinjauan pustaka, tahapan kedua atau tahapan perancangan dan pengujian sistem dimulai dari metode pengumpulan data, analisis kebutuhan sistem, perancangan sistem,

implementasi sistem dan pengujian metode, dan tahapan ketiga atau tahapan penyelesaian berupa kesimpulan atau penyusunan laporan dari penelitian ini. Berikut penjelasan masing-masing tahapan kegiatan yang dilakukan.

3.1 Identifikasi Masalah

Tahap awal dalam penelitian ini adalah merumuskan masalah yang akan di jadikan sebagai objek penelitian. Perumusan masalah dilakukan dengan terlebih dahulu melihat kondisi aktual di lapangan. Setelah masalah dirumuskan langkah selanjutnya adalah menentukan tujuan dari penelitian. Tujuan penelitian ini merupakan sasaran yang nantinya ingin diwujudkan dari penyelesaian permasalahan yang diteliti.

3.2 Tinjauan Pustaka

Tinjauan pustaka dilakukan guna mencari literatur pendukung penelitian ini. Pada tahap ini dijelaskan dengan mengunjungi dan mempelajari website atau situs-situs yang berhubungan dengan sistem metadata forensik, teori-teori, untuk pengumpulan data dan *tools* yang digunakan oleh penulis. Serta di jelaskan mengenai metode yang digunakan.

3.3 Metode Pengumpulan Data

Metode yang digunakan untuk mengumpulkan data pada penelitian ini yaitu dengan melakukan studi literatur. Studi literatur digunakan untuk mengumpulkan data dari penelitian terdahulu, pembelajaran dari berbagai macam literatur dan dokumen seperti buku, jurnal dan teori-teori yang mendukung penelitian, *tools* yang akan digunakan dan data penunjang lainnya yang berkaitan dengan sistem metadata forensik. Literatur juga berisi uraian berisi tentang teori, temuan dan bahan penelitian lain yang di peroleh dari bahan acuan untuk dijadikan landasan kegiatan.

3.4 Analisis Kebutuhan Sistem

Untuk mempermudah menganalisis sebuah sistem dibutuhkan dua jenis kebutuhan. Kebutuhan fungsional dan kebutuhan nonfungsional. Kebutuhan fungsional adalah kebutuhan yang berisi proses-proses apa saja yang nantinya dilakukan oleh sistem. Sedangkan kebutuhan nonfungsional adalah kebutuhan yang menitikberatkan pada properti prilaku yang dimiliki oleh sistem.

3.1.1 Kebutuhan Fungsional

Kebutuhan fungsional dalam analisis metadata forensik ini adalah sebagai berikut:

1. Sistem ini mampu memahami karakteristik metadata setiap file.
2. Sistem ini juga mampu merancang sebuah metode algoritma untuk melakukan korelasi metadata setiap file.

3.1.2 Kebutuhan Nonfungsional

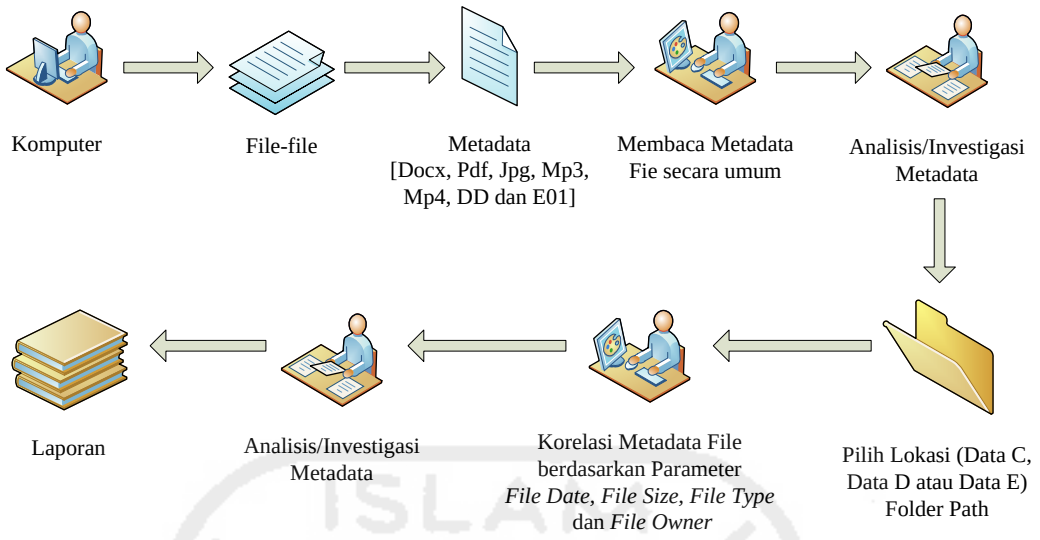
Kebutuhan nonfungsional yaitu terdiri dari perangkat keras dan perangkat lunak yang digunakan dalam membangun sistem metadata forensik ini adalah sebagai berikut:

1. Sistem Operasi Windows 10
2. Kompiler Netbeans IDE 8.0 untuk Pemrograman Java
3. Semua jenis file yang ada didalam komputer, seperti TXT, DOCX, PDF, JPG, RAR, HTML, MP3, MP4, DD, E01 dan lain-lain
4. CPU intel Core i3
5. Memory 6 GB
6. Hard Disk 500 GB
7. Monitor dengan Resolusi 1024 x 600 Pixel
8. Keyboard dan Mouse

3.5 Perancangan Sistem

Metode yang digunakan untuk membangun sebuah sistem atau metode algoritma metadata forensik ini yaitu dengan menggunakan metode perancangan terstruktur serta menggunakan *Workflow* (Bagan Kerja) dan *Flowchart* (Bagan Alir). Perancangan ini dimulai dari perancangan secara umum yang disebut dengan desain konseptual (*conceptuai design*) atau desain logikal (*logical design*). Hasil dari tahap ini adalah bentuk esensial model, yaitu apa yang harus dilakukan oleh sistem akan di implementasikan. Kemudian perancangan sistem dilanjutkan ke perancangan terperinci.

Adapun alur rancangan secara umum dari sistem metada forensik bisa di lihat pada gambar 3.2 berikut:

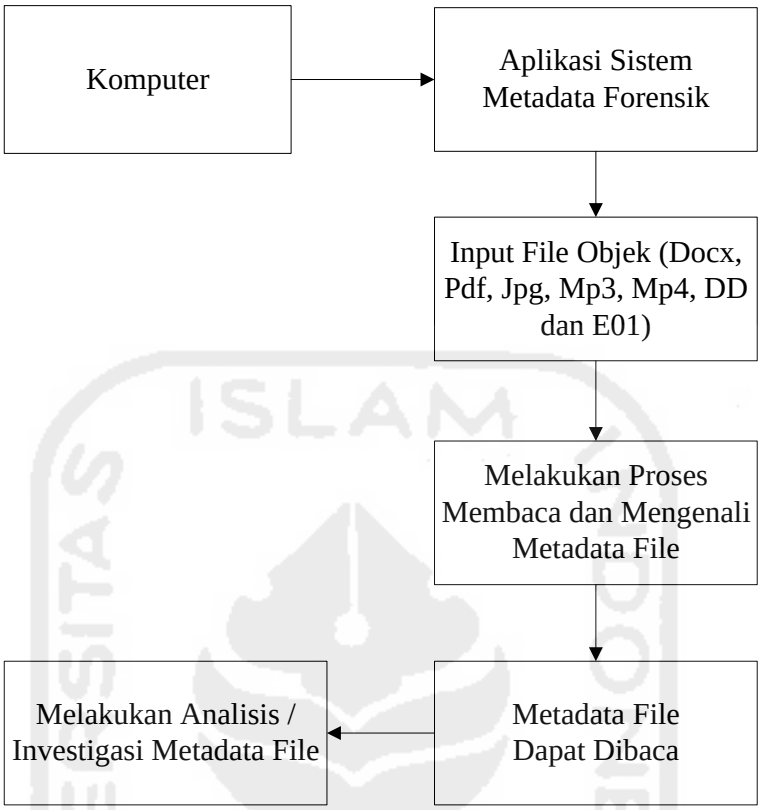


Gambar 3. 2 Alur Rancangan Umum Sistem Metada Forensik

Penjelasan gambar 3.2 alur rancangan umum sistem metada forensik:

1. Seorang penyidik atau ahli forensik digital akan memeriksa komputer/laptop yang diduga digunakan sebagai tindak kejahatan yang berkaitan dengan jenis file atau data yang ada di dalam komputer/laptop.
2. Pertama-tama Penyidik akan menyalakan sebuah komputer/laptop.
3. Ada banyak jenis file document di dalam komputer/laptop tersebut.
4. Metadata file yang akan di pahami adalah file yang ber-extension DOCX, PDF, JPG, MP3, MP4, DD dan E01.
5. Selanjutnya penyidik akan membaca metadata tersebut dengan menggunakan sebuah sistem atau algoritma metadata forensik yang telah di bangun.
6. Setelah metadata file tersebut di baca, selanjutnya dilakukan investigasi atau analisis metadata oleh seorang investigator.
7. Setelah di lakukan analisa, investigator kemudian melakukan korelasi metadata file berdasarkan parameter dari Metadata *File Date*, *File Size*, *File Type* dan *File Owner* dengan menggunakan sebuah sistem atau algoritma metadata forensik yang telah di bangun.
8. Ditemukan beberapa file yang berhubungan dengan parameter-parameter yang telah di inputkan.
9. Setelah apa yang ditemukan dari korelasi tersebut, kemudian di analisis terlebih dahulu sebelum dibuat pelaporannya.
10. Laporan-laporan dari hasil sistem metadata forensik tersebut.

Berikutnya gambar 3.3 menampilkan alur rancangan dalam memahami karakteristik metadata setiap file dari sebuah sistem metada forensik yang telah dibangun yaitu sebagai berikut:

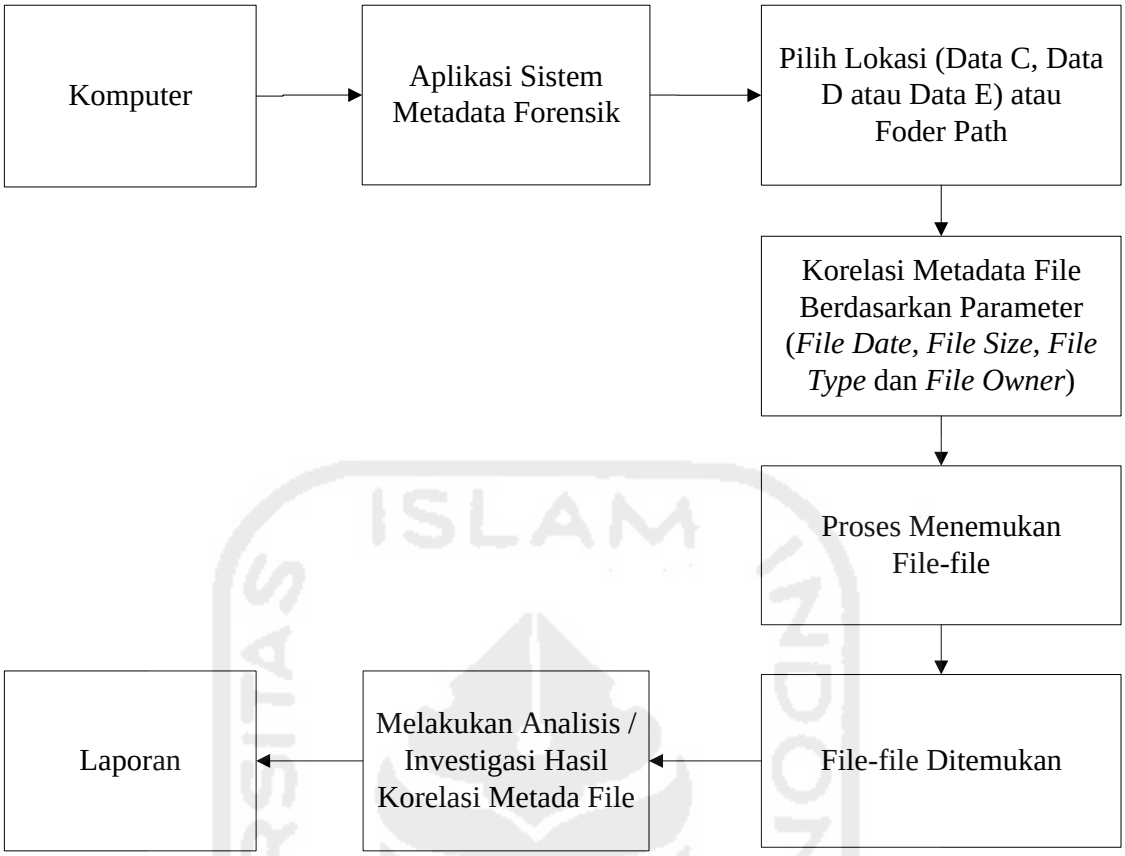


Gambar 3. 3 Alur Rancangan Memahami Karakteristik Metadata File

Penjelasan gambar 3.3 alur rancangan memahami karakteristik metadata file:

1. Pertama-tama menjalankan sebuah sistem atau algoritma metadata forensik yang telah dibangun.
2. Inputkan sebuah file kedalam sistem yang akan dibaca metadatanya.
3. Metadata file yang akan di baca yaitu file yang ber-extension DOCX, PDF, JPG, MP3, MP4, DD dan E01.
4. Pada saat menjalankan menu sistem mulai membaca, sistem akan memproses file yang telah di inputkan untuk kemudian bisa di baca metadatanya.
5. Setelah proses selesai, maka ditampilkan metadata dari file yang telah di inputkan tersebut.
6. Selanjutnya dilakukan sebuah analisa atau investigasi dari metadata file yang telah terbaca.

Selanjutnya gambar 3.4 menampilkan alur rancangan korelasi metadata file dengan parameter-parameter dari sebuah sistem metada forensik yang telah dibangun yaitu sebagai berikut:



Gambar 3. 4 Alur Rancangan Korelasi Metadata File

Penjelasan gambar 3.4 alur rancangan korelasi metadata file:

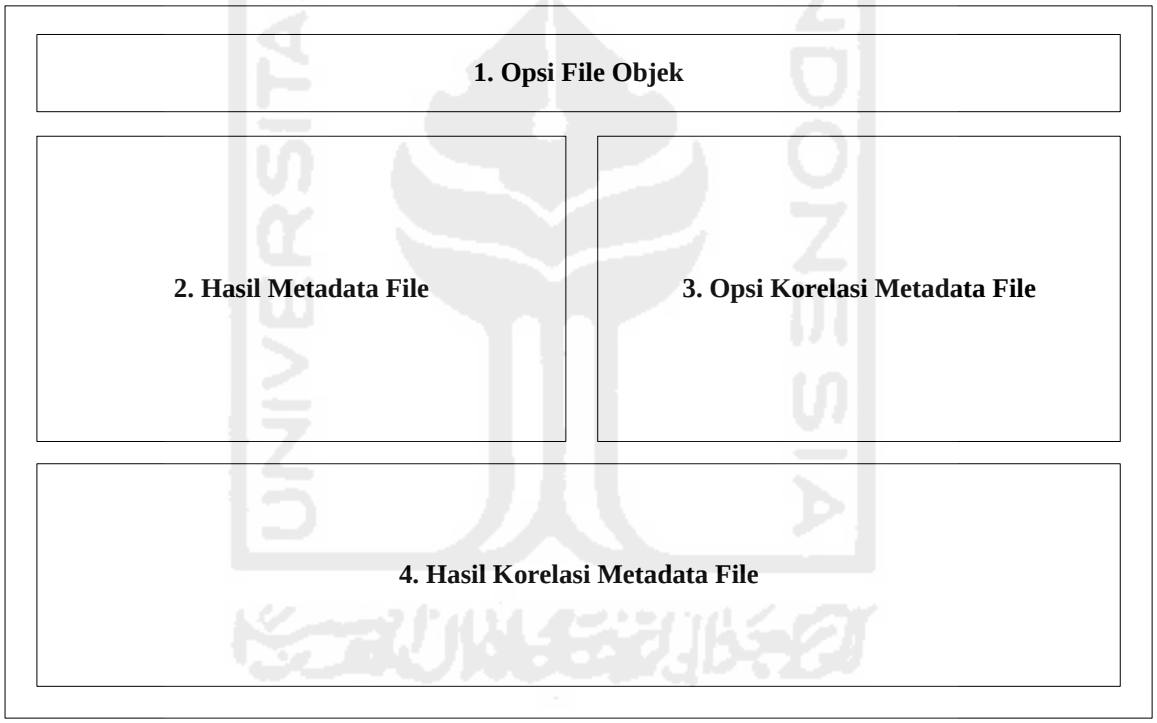
1. Sistem atau algoritma metadata forensik yang telah dijalankan sedang menunggu perintah untuk melakukan korelasi metadata file.
2. Metadata file yang sudah terbaca dan sudah di analisa/investigasi siap untuk di korelasikan metadatanya.
3. Korelasi metadata file berdasarkan parameter dari metadata yang telah di temukan yaitu berdasarkan parameter dari Metadata File Date, Size, Type File dan Owner.
4. Pilih lokasi korelasi terlebih dahulu (Data C, Data D atau Data E) atau Foder Path.
5. Pada saat Korelasi File dijalankan, sistem akan memproses untuk menemukan file-file yang berdasarkan parameter metadata file yang sudah di inputkan tersebut.
6. Ditemukan file-file dari proses korelasi yang berdasarkan parameter-parameter metadata file yang ada.
7. Dilakukan sebuah analisa atau investigasi terlebih dahulu dari hasil pencarian korelasi tersebut.
8. Dan yang terakhir, membuat laporan dari hasil sistem metadata forensik yang telah dibangun.

3.6 Implementasi Sistem

Implementasi adalah proses untuk memastikan bahwa sistem atau metode algoritma yang dibangun bebas dari kesalahan dan mudah digunakan oleh pengguna dalam hal ini seorang investigator. Dalam pembuatan sistem atau metode algoritma metadata forensik ini, implementasi sistem dilakukan dengan menggunakan bahasa pemrograman Java Netbeans IDE 8.0. Kebutuhan antarmuka dari program tersebut adalah sebagai berikut:

1. Tampilan Windows 10 yang *user friendly*.
2. Tampilan Sistem Metadata Forensik dengan Aplikasi Java.
3. Terdapat beberapa menu yang mempunyai fungsi dan pengolahan masing-masing.

Berikut adalah desain implementasi sistem metadata forensik yang akan dibangun:



Gambar 3. 5 Desain Implementasi Sistem Metadata Forensik

Penjelasan gambar 3.5 desain implementasi sistem metadata forensik:

1. Opsi File Objek, tempat memilih file yang akan dibaca metadatanya. File tersebut terdiri dari tujuh macam file yang ber-extension DOCX, PDF, JPG, MP3, MP4, DD dan E01.
2. Hasil Metadata File, tempat menampilkan hasil metadata file yang sudah diinputkan dari opsi file objek.
3. Opsi Korelasi Metadata File, tempat pemilihan korelasi metadata file yang akan dicari filenya di komputer, berdasarkan korelasi parameter dari metadata file Date, Size, Type File dan Owner.

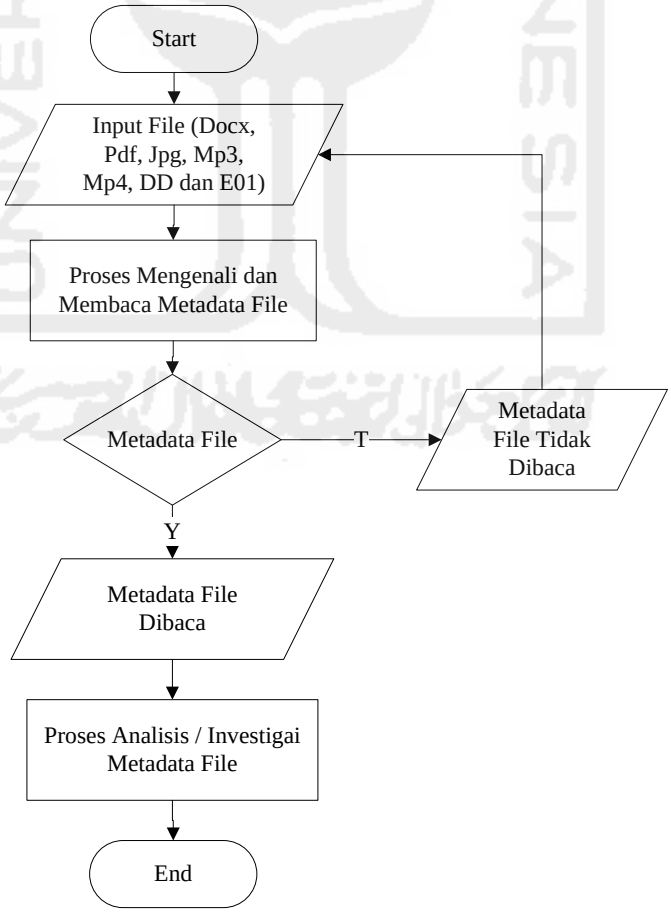
4. Hasil Korelasi Metadata File, tempat menampilkan file-file yang sudah dicari berdsarkan dari opsi korelasi metadata file.

3.7 Pengujian Sistem

Pada tahapan ini dilakukan pengujian sistem metadata forensik yang bertujuan untuk mendeteksi kegagalan perangkat lunak sehingga kesalahan sistem dapat ditemukan dan diperbaiki. Pengujian perangkat lunak (*software testing*) merupakan suatu investigasi yang dilakukan untuk mendapatkan informasi mengenai kualitas dari produk atau layanan yang sedang diuji (*under test*).

Dalam penelitian ini dibutuhkan sebuah komputer / laptop yang berisikan file-file untuk di lakukan proses uji coba. Proses pengujian sistem ini dilakukan dalam dua tahap, dimana masing-masing tahapnya yaitu sebagai berikut:

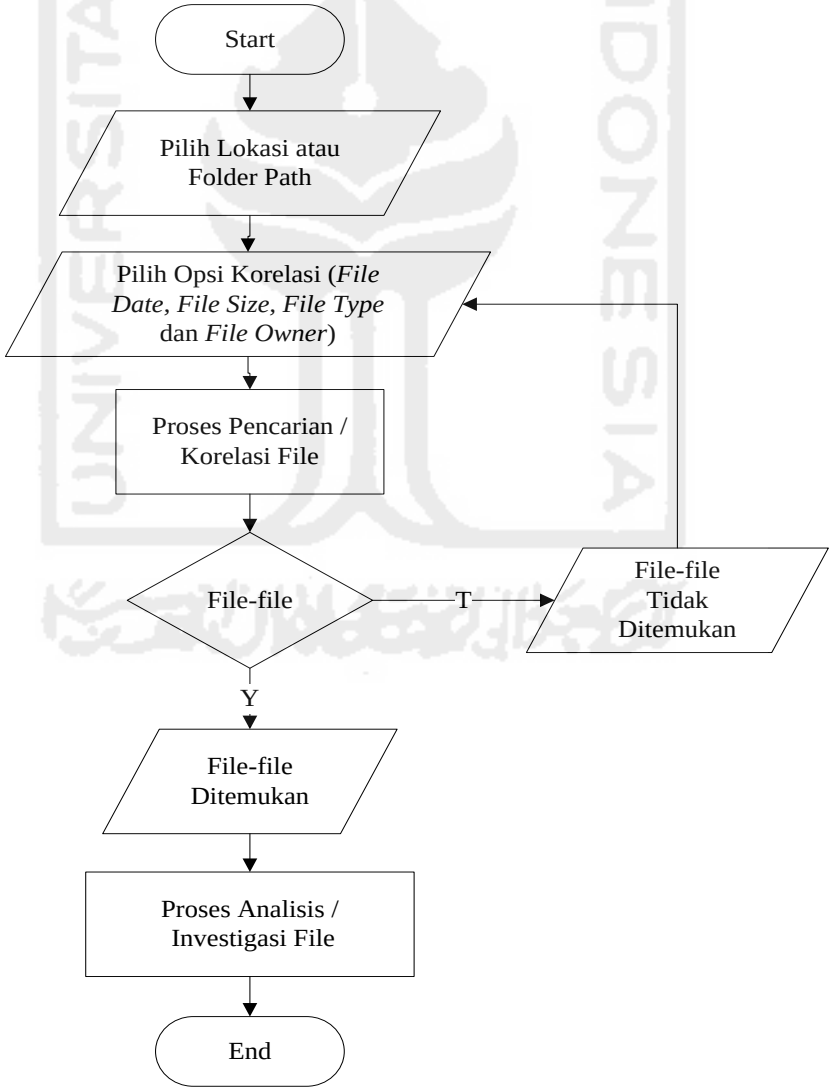
1. Tahapan pertama yaitu proses pengujian yang dilakukan untuk membaca atau memahami karakteristik metadata file dalam sebuah sistem atau algoritma metadata forensik yang di gambarkan dalam bagan alur *flowchart* berikut:



Gambar 3. 6 Alur Proses Pengujian Membaca Karakteristik Metadata File

Pertama-tama dilakukan *start* atau sistem dijalankan, setelah itu dilakukan penginputan file yang akan di baca atau di kenali metadatanya, dimana file yang akan dibaca yaitu file ber-extension docx, pdf, jpg, mp3, mp4, dd dan e01, kemudian program akan melakukan pemrosesan file yang telah di inputkan, terdapat kondisi, dimana metadata file yang tidak bisa dibaca akan kembali ke inputan file objek, tetapi metadata file yang dapat terbaca akan langsung ditampilkan metadata filenya, selanjutnya dilakukan sebuah analisis / investigasi terhadap metadata file yang sudah dibaca dan terakhir program di tutup atau selesai di jalankan.

2. Tahapan kedua yaitu proses pengujian yang dilakukan untuk korelasi metadata file berdasarkan parameter-parameter yang telah di *setting* dalam sebuah sistem atau algoritma metadata forensik yang di gambarkan dalam bagan alur *flowchart* berikut:



Gambar 3. 7 Alur Proses Pengujian Sistem Korelasi Metadata File

Pertama-tama dilakukan *start* atau sistem yang sudah dijalankan tinggal menunggu perintah mulai lagi, dilakukan penginputan lokasi korelasi terlebih dahulu (Data C, Data D atau Data E) atau Foder Path yang ada didalam komputer, setelah itu dilakukan pemilihan metadata file berdasarkan korelasi parameter dari *File Date, File Size, File Type* dan *File Owner*, kemudian sistem akan melakukan proses menemukan korelasi metadata file yang telah dibuat, terdapat pernyataan atau kondisi dimana terdapat banyak file-file, jika file-file masih belum ditemukan korelasi metadatanya maka sistem akan kembali memilih opsi korelasi metadata file seperti biasa, tetapi apabila file-file sudah ditemukan dari korelasi metadata filenya berdasarkan parameter yang telah dibangun maka akan dilanjutkan ke analisis / investigasi file-file yang sudah ditemukan tersebut, dan yang terakhir sistem ini selesai digunakan dan ditutup.

Untuk hasil metadata file dan hasil korelasi metadata file, jika sudah sesuai dengan opsi yang dimasukkan dan opsi yang dipilih maka terakhir dilakukan pembuatan laporan-laporan dari penelitian ini, tetapi jika hasilnya masih beda dengan yang di inputkan atau dengan yang dipilih maka aplikasi sistem yang dibuat akan perbaiki lagi sampai hasilnya benar-benar sesuai.

3.8 Kesimpulan / Penulisan Laporan

Langkah terakhir dari penelitian ini adalah membuat Laporan Tesis. Laporan ini berisi hal-hal yang dikerjakan selama penelitian dan hasil yang didapatkan pada saat melakukan penelitian. Dalam penulisannya, format yang digunakan adalah berdasarkan format yang telah diterapkan oleh Program Magister Teknik Informatika Universitas Islam Indonesia Yogyakarta.

Bab 4 Hasil dan Pembahasan

4.1 Implementasi Sistem

Dalam bab ini akan dilakukan implementasi sistem, yakni mulai pembuatan program aplikasi metadata forensik dengan membaca dan korelasi metadata file menggunakan bahasa pemrograman Java Kompiler Netbeans IDE 8.0. Program ini disebut metadata forensik, karena mampu membaca metadata setiap file dan mampu menemukan file berdasarkan korelasi setiap metadatanya. Hasil dari pembuatan program ini akan diimplementasikan pada beberapa file yang ada didalam komputer yang sudah ditentukan.

4.1.1 Jenis Komputer dan Sistem Operasi

Komputer / laptop yang digunakan dalam pengujian metode sistem metadata forensik ini adalah Laptop merk HP, berikut spesifikasinya:

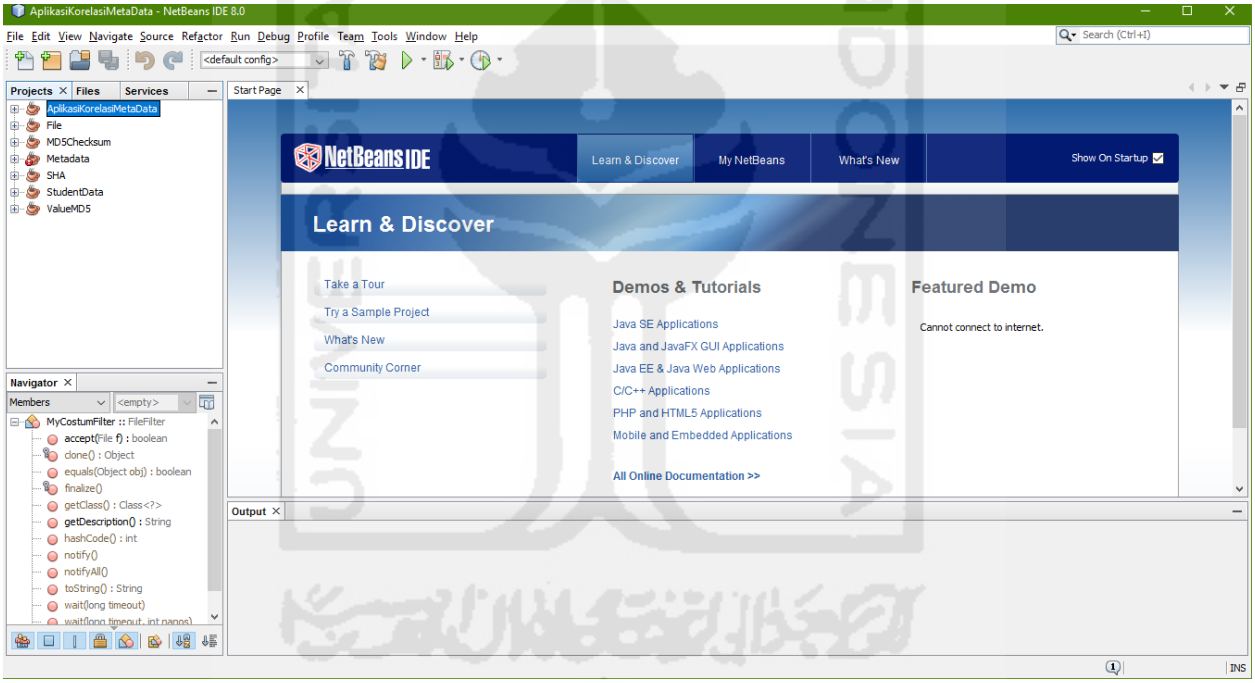
Tabel 4. 1 Spesifikasi Laptop HP

Resolusi Layar	1366 x 768
Ukuran Layar	14 FULL HD
Tipe Layar	Active Matrix TFT Color LCD
CPU	Intel® Core i3 2.40 GHz
Memori/RAM	6 GB DDR 3
Harddisk	500 GB
DVD	DVD Writer
Koneksi	Bluetoth 4.0 + HS, Wi-Fi, Gigabit Internet
Microphone	Ya
Port USB	USB 3.0 Terbaru!
Webcam	Ya Webcam terintegrasi
HDMI	Ya (untuk LCD projector)
Flash Kamera	Ya
Baterai	Lithium Ion (Li-Ion)
Berat	2.2 Kilogram

Sedangkan Sistem Operasi yang digunakan dalam komputer ini adalah jenis Sistem Operasi Windows 10. Sistem Operasi Windows 10 adalah Sistem Operasi yang dikembangkan oleh Microsoft Corporation yang menggunakan antarmuka dengan berbasikan GUI (Graphical User Interface) atau tampilan antarmuka bergrafis pada umumnya sistem oprasi ini banyak sekali di gunakan oleh masyarakat, dari kalangan menengah ke atas hingga ke bawah.

4.1.2 Jenis Tools Aplikasi

Merupakan alat yang digunakan untuk menggambarkan bentuk logika model dari suatu sistem dengan menggunakan simbol-simbol, lambang-lambang, diagram-diagram ataupun GUI yang menunjukan secara tepat arti dan fungsinya. Adapun tools aplikasi yang dijelaskan sebagai model sistem yang telah dirancang yaitu Netbeans IDE 8.0 untuk pemrograman Java.



Gambar 4. 1 Tampilan Pemrograman Java Netbeans IDE 8.0

4.1.3 Jenis File

Sistem metadata forensik yang telah dibangun ini bisa membaca semua macam tipe jenis file yang ada didalam komputer, contohnya pada laptop dengan merk HP, dimana didalamnya terdapat berbagai jenis macam file seperti file *Microsoft Office* (*word, excel, power point, acces, visio,* dan lain-lainnya), *txt, pdf, rar, py, html, php, jpg, bmp, png, apk, fbk, mp3, mp4,* dan berbagai macam extensi aplikasi yang masih banyak lagi yang ada didalam Laptop Merk HP yang semuanya bisa dibaca metadatanya, akan tetapi dalam tahap pengujian metode sistem ini, ada tujuh macam tipe jenis file yang akan diuji sebagai contoh yaitu DOCX, PDF, JPG, MP3, MP4, DD dan E01.

Ketujuh jenis file ini digunakan sebagai uji coba, karena didalam beberapa kasus yang sering terjadi didalam dunia forensika deigital yaitu seringnya melibatkan ketujuh jenis file ini, sebagai contoh seringnya pemalsuan dokumen (*docx, pdf*) yang terjadi didunia lelang e-procurement, gambar (*jpg*) yang sering menimpa artis tanah air, suara (*mp3*) sadapan yang sering digunakan oleh KPK untuk menjerat para Politikus yang Kuroptor, video-video (*mp4*) asusila yang akan dijadikan barang bukti oleh Polisi untuk menjerat para pelaku seperti video ariel yang sempat heboh beberapa tahun silam hingga kopi sianida mirna solihin yang masih hangat-hangatnya sekarang ini, dan file hasil akuisisi dd dan E01 untuk memastikan apakah dalam penangangan barang bukti sudah sesui dengan SOP yang berlaku. Berikut masing-masing penjelasannya:

1. File Dokumen Ber-Extensi DOCX

   Microsoft Word adalah aplikasi pengolah kata yang sangat populer pada saat ini, aplikasi yang dikembangkan oleh microsoft ini terdapat dalam satu paket microsoft office yang berisi microsoft word, microsoft excel, microsoft power point, microsoft office publisher microsoft office access dan lain-lain. Dalam perkembangannya microsoft word mengalami banyak perkembangan dari tahun ke tahun, dengan perkembangan tersebut microsoft telah menambahkan database dan tool yang baru untuk menyempurnakan agar microsoft word lebih mudah untuk digunakan. Dengan microsoft word dapat memudahkan kerja manusia dalam melakukan pengetikan surat maupun dokumen lain.

Gambar 4. 2 Icon File Extensi DOCX

2. File Ebook Ber-Extensi PDF

   PDF atau Portable Document Format adalah sebuah format file yang diciptakan oleh Adobe System, Inc. File jenis ini sangat populer dan banyak digunakan terutama dalam bentuk ebook. Dokumen dengan format portable ini dibuat dan digunakan untuk tujuan kemudahan sekaligus keamanan dokumen.

Gambar 4. 3 Icon File Extensi PDF

3. File Gambar Ber-Extensi JPG

   JPG adalah jenis data yang dikembangkan oleh *Joint Photographic Experts Assemble* (JPEG) yang dijadikan standar untuk para fotografer profesional. Setiap kali menyimpan ke tipe JPG dari tipe lain, ukuran gambar biasanya mengecil, dan kualitasnya turun dan tidak dapat dikembalikan lagi. Ukuran file BMP dapat turun menjadi seper sepuluh setelah dikonversi menjadi JPG. Meskipun dengan penurunan kualitas gambar, pada gambar-gambar tertentu (misalnya pemandangan), penurunan kualitas gambar hampir tidak akan terlihat oleh mata.

Gambar 4. 4 Icon File Extensi JPG

4. File Audio Ber-Extensi MP3

   Mp3 merupakan format kompresi audio yang dikembangkan oleh Moving Picture Experts Group (MPEG). Format file ini menggunakan Layer 3 kompresi audio yang secara umum digunakan untuk menyimpan file–file musik dan audiobooks dalam hard drive.

Gambar 4. 5 Icon File Extensi MP3

5. File Video Ber-Extensi MP4

   File MP4 yang dikembangkan oleh Organisasi Standarisasi Internasional (ISO) berjalan dengan baik dengan hampir semua jenis media player dan perangkat. Karena MP4 dikembangkan untuk MPEG4 dikodekan media, dapat juga ditemukan dalam MPEG-4 spesifikasi sebagai bagian 14 itu MP4 tidak format kontainer hanya dibawah MPEG-4 tetapi itu adalah turunan dari MPEG-4 bagian 12 spesifikasi yang lebih umum untuk menyimpan file MPEG-4.

Gambar 4. 6 Icon File Extensi MP4

6. File Akuisisi Ber-Extensi DD

   Disk Image atau sering disebut Imaging dalam dunia digital forensic adalah suatu proses dari file tunggal atau suatu perangkat media penyimpanan seperti hardisk int/eks, usb flash drive, cd, dvd, dan lain-lain yang mengandung isi lengkap dengan strukturnya yang kemudian di *cloning* (perbanyak / penggandaan) dengan isi dan struktur yang sama persissempurna dari yang asli tanpa selisih ukuran se-bit pun di dalamnya. Mudahnya *disk image* itu proses memetakan penggandaan barang bukti dengan metode *bit by bit copy*.

Gambar 4. 7 File Extensi DD

7. File Akuisisi Ber-Extensi E01

   Encase Forensic adalah alat forensik yang paling banyak dikenal dan digunakan, yang telah diproduksi dan diluncurkan oleh Guidance Software Inc. Encase tertanam dengan berbagai fungsi forensik yang meliputi atribut seperti pencitraan disk dan pelestarian, pemulihan data

mutlak dalam bentuk aliran bit, dalam seri ini aplikasi *humongous*, ketika encase digunakan untuk membuat *backup* dari hard drive, CD, USB drive, dll, file yang dikenal sebagai "E01" diproduksi. Ini ".E01" ekstensi file terutama diakui sebagai "Encase Image File Format". The E01 format file image juga dikenal sebagai EWF (singkatan Saksi Ahli Format). Konsep E01 image encase dikembangkan oleh perangkat lunak encase muncul sebagai hasil dari upaya efisien oleh Guidance Software untuk membantu penyelidik forensik, analis, dan ilmuwan forensik dalam menemukan data yang terorganisasi dan sistematis untuk penyelidikan.

Gambar 4. 8 File Extensi E01

4.1.4 Jenis Karakteristik Metadata File

Metadata mempunyai peranan dalam pencarian informasi yaitu memberikan informasi ketersediaan data (diperlukan untuk mengetahui ketersediaan data pada suatu lokasi geografis), kesesuaian pengguna (mengetahui suatu data telah memenuhi spesifikasi yang diinginkan), akses (memperoleh suatu data yang teridentifikasi) dan transfer (memperoleh, menggunakan dan memproses data) (SNI Metadata, 2008). Metadata terdiri dari komponen (role) dan elemen. Role merupakan header dari elemen, elemen berisi informasi mengenai data.

Metadata terdiri atas beberap jenis standar dalam menampilkan data. Secara sederhana yang dimaksud dengan standar metadata adalah satu set terminologi serta definisi umum yang digunakan dalam metadata serta dipresentasikan dalam format terstruktur. Standar metadata spasial dibuat dan dikembangkan untuk mendefinisikan informasi yang diperlukan oleh seorang pengguna prospektif untuk mengetahui ketersediaan suatu set data spasial, mengetahui kesesuaian set data spasial untuk penggunaan yang diinginkan, mengetahui cara-cara pengaksesan data spasial serta untuk mentransfer set data spasial dengan sukses. Walaupun demikian standar tidak menetapkan tatacara bagaimana informasi diorganisasikan dalam suatu sistem komputer atau dalam suatu transfer data, tidak juga menetapkan tatacara bagaimana informasi tersebut ditransmisikan, dikomunikasikan atau disampaikan kepada pengguna. Jika standar metadata

geospatial terkesan sangat komplek itu karena standar tersebut didesain untuk mendeskripsikan seluruh data geospasial yang bisa dideskripsikan.

Beberapa standar yang digunakan dalam pembuatan metadata spasial, yaitu: FGDC, ISO 19115, Dublin Core dan SNI Metadata. Standar metadata ISO 19115/19139 merupakan standar untuk pembuatan metadata data geospasial. Format ISO 19115 merupakan standar internasional untuk metadata informasi geografi dan format ISO 19139 merupakan skema implementasi untuk ISO 19115. ISO 19115 mempunyai 409 elemen dan terdapat 22 elemen inti (core element) yang dibutuhkan untuk mendeskripsikan data dan memiliki elemen compound (role) dibawahnya. Role tersebut terbagi menjadi 11 komponen utama, yaitu identifikasi, batasan, kualitas data, representasi spasial, sistem referensi, informasi data, referensi portal katalog, distribusi, informasi tambahan dan informasi skema aplikasi (ISO, 2003). Skema ISO 19139 digunakan untuk mendeskripsikan, melakukan validasi dan melakukan pertukaran metadata geospasial yang disiapkan dalam format XML (Extensible Markup Language).

Beberapa karakteristik metadata yang ditampilkan dalam sistem ini yaitu dibagi dalam 3 kategori:

1. Metadata General, yaitu lokasi file, nama file, type file, owner dan computer.
2. Metadata Detail, yaitu creationTime, lastAccessTime, lastModifiedTime, isDirectory, isOther, isRegularFile, isSymbolicLink dan Size.
3. Metadata Checksum, yaitu Nilai MD5 dan SHA-256.

4.1.5 Jenis Korelasi Metadata File

Secara sederhana, korelasi dapat diartikan sebagai hubungan. Namun ketika dikembangkan lebih jauh, korelasi tidak hanya dapat dipahami sebatas pengertian tersebut. Korelasi merupakan salah satu teknik analisis dalam statistik yang digunakan untuk mencari hubungan antara dua variabel yang bersifat kuantitatif. Hubungan dua variabel tersebut dapat terjadi karena adanya hubungan sebab akibat atau dapat pula terjadi karena kebetulan saja. Dua variabel dikatakan berkolerasi apabila perubahan pada variabel yang satu akan diikuti perubahan pada variabel yang lain secara teratur dengan arah yang sama (*korelasi positif*) atau berlawanan (*korelasi negatif*).

Komputer berisi data dan program, Program merupakan file komputer yang digunakan untuk melakukan tugas tertentu, sedangkan data merupakan file hasil kerja program komputer yang dapat diedit, dibuka, dihapus, dan sebagainya. Sementara itu, folder adalah suatu tempat untuk mengumpulkan file.

Dalam pengujian metode ini ada empat jenis korelasi metadata yang dijadikan sebagai contoh, yaitu metadata *file date, size, type file* dan *owner*. Sesorang investigasi bisa mencari semua jenis file (tidak hanya 7 macam jenis file yang telah dibahas diatas; Docx, Pdf, Jpg, Mp3, Mp4, DD dan E01) yang ada didalam komputer berdasarkan dari empat pilihan korelasi tersebut.

4.1.6 Live Data dalam Proses Investigasi Metadata

Live data adalah format data didalam sebuah sistem yang dapat diakses secara langsung oleh pengguna. Dalam pengumpulan barang bukti live data cenderung lebih mudah karena data yang didapatkan secara langsung dapat dilihat dan dianalisa lebih lanjut lagi.

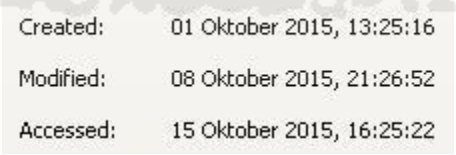
Secara umum live data mempunyai nilai yang kuat untuk dijadikan barang bukti karena data yang diperoleh terbukti secara langsung dan dilihat secara langsung serta berhubungan dengan sesuatu maupun tindakan apa yang telah dilakukan terhadap file tersebut.

Selanjutnya karena live data dibuat dan dikelola oleh sistem opaerasi dan aplikasi perangkat lunak. Live data memiliki catatan waktu yang akurat, tergantung dengan kesesuaian jam yang ada dalam perangkat tersesebut.

Proses yang terjadi dalam sebuah sistem mempunyai catatan waktu. Catatan waktu yang terjadi biasanya dikenal dengan istilah MAC (*Modified, Accessed, Created*).

1. ***Modified*** adalah catatan yang menampilkan waktu kapan terakhir kali data dimodifikasi, yaitu ketika terakhir kali disimpan. Modified menampilkan waktu terakhir perubahan file.
2. ***Accessed*** adalah catatan untuk menampilkan kapan waktu terakhir kali file tersebut diakses.
3. ***Created*** adalah catatan yang menampilkan kapan data tersebut dibuat pertama kalinya.

Untuk lebih jelasnya mari kita lihat gambar berikut ini:



Gambar 4. 9 MAC (*Modified, Accessed, Created*)

Gambar tersebut menunjukan kapan file tersebut dibuat pertama kali, kemudian menujukkan juga kapan file dimofikasi dan di simpan kemabli dan yang terakhir menunjukkan kapan file tersebut di access oleh pengguna.

Catatan waktu ini merupakan sebuah file metadata yang dapat menunjukan urutan kejadian maupun kegiatan yang terjadi didalam sebuah sistem.

4.2 Source Code Metadata Forensik

Pada kali ini akan dibahas *source code* algortima metadata forensik. *Source code* yang diambil hanya sebagian dari *coding* yang mempunyai peranan khusus untuk Melihat Karaktersitik Metadata File dan Korelasi File berdasarkan metadata file dari Parameter *Date File, Size File, Type File* dan *Owner File*.

4.2.1 Source Code Membaca Karakteristik Metadata File

1. *Source Code* Melihat Metadata File Secara Umum

Tabel 4. 2 *Source Code Metadata General*

No.	Source Code
1	public void getMetaData(File f, String namafile) {
2	tModel = new DefaultTableModel(header, 0);
3	Path file = Paths.get(f.getAbsolutePath());
4	try {
5	txtLokasi.setText(f.getPath());
6	txtNamaFile2.setText(namafile);
7	txtTypeFile.setText(getFileExtension(f));
8	txtComputer.setText(getComputerName());
9	FileOwnerAttributeView ownerAttributeView = Files.getFileAttributeView(f.toPath(), FileOwnerAttributeView.class);
10	UserPrincipal owner = ownerAttributeView.getOwner();
11	txtAuthors.setText(owner.getName().substring(owner.getName().lastIndexOf("\\") + 1));
12	} catch (IOException ex) {
13	}
14	tabelMeta.setModel(tModel);
15	}

Pada baris 1 untuk memvalidasi bahwa method ini diakses dan mengembalikan nilai yang dikirim dari class Main, pada baris 2 untuk menaruh nilai yang dihasilkan pada sebuah tabel, pada baris 3 akan selalu mengembalikan pemisah direktori sesuai dengan platform pengguna, pada baris 5 mendifinisikan metadata Folder Path atau Lokasi File, pada baris 6 mendifinisikan metadata Name File, pada baris 7 mendifinisikan metadata Type File, pada baris 8 mendifinisikan metadata Computer yang dipakai, pada baris 9 dan 10 atribut yang di definisikan oleh interface untuk Owner File, pada baris 11 mendifinisikan metadata Owner atau pemilik file yang ada didalam komputer, pada baris 12 untuk *exception* yang berhubungan dengan *input* dan *output*, pada baris 14 di *set* langsung pada interface tabel metadata.

2. *Source Code* Melihat Metadata File Secara Jelas

Tabel 4. 3 *Source Code Metadata Detail*

No.	Source Code
1	public void getMetaData(File f) {
2	tModel = new DefaultTableModel(header, 0);
3	Path file = Paths.get(f.getAbsolutePath());
4	try {
5	BasicFileAttributes attr = Files.readAttributes(file, BasicFileAttributes.class);
6	String data1[] = {"1", "creationTime: ", attr.creationTime().toString()};
7	tModel.addRow(data1);
8	String data8[] = {"2", "lastAccessTime: ", attr.lastAccessTime().toString()};
9	tModel.addRow(data8);
10	String data2[] = {"3", "lastModifiedTime: ", attr.lastModifiedTime().toString()};
11	tModel.addRow(data2);
12	lastmodifide=attr.lastModifiedTime().toMillis();
13	System.out.println("LONG File: "+lastmodifide);
14	String data3[] = {"4", "isDirectory: ", String.valueOf(attr.isDirectory())};
15	tModel.addRow(data3);
16	String data4[] = {"5", "isOther: ", String.valueOf(attr.isOther())};
17	tModel.addRow(data4);
18	String data5[] = {"6", "isRegularFile: ", String.valueOf(attr.isRegularFile())};
19	tModel.addRow(data5);
20	String data6[] = {"7", "isSymbolicLink: ", String.valueOf(attr.isSymbolicLink())};
21	tModel.addRow(data6);
22	String data7[] = {"8", "size: ", String.valueOf(attr.size())};
23	size = attr.size();
24	tModel.addRow(data7);
25	} catch (IOException ex) {
26	}
27	tabelMeta.setModel(tModel);
28	}

Pada baris 1 untuk memvalidasi bahwa method ini diakses dan mengembalikan nilai yang dikirim dari class Main, pada baris 2 untuk menaruh nilai yang dihasilkan pada sebuah tabel, pada baris 3 akan selalu mengembalikan pemisah direktori sesuai dengan platform pengguna, pada baris 5 atribut yang di definisikan oleh interface, pada baris 6 dan 7 mendifinisikan metadata *creationTime* yang akan dibaca dan langsung memiliki sebuah baris tabel pada *interface*, begitu juga pada baris selanjutnya sampai dengan baris 24 mendifinisakan metadata *lastAccessTime*, *lastModifiedTime*, *isDirectory*, *isOther*, *isRegularFile*, *isSymbolicLink* dan *Size* serta memiliki sebuah baris tabel pada *interface*, pada baris 25 untuk *exception* yang berhubungan dengan *input* dan *output*, pada baris 27 di *set* langsung pada *interface* tabel metadata.

3. *Source Code* Melihat Metadata File Nilai *Hashing*

Tabel 4. 4 *Source Code Metadata Checksum*

No.	Source Code
1	public byte[] createChecksum(String filename, String type) throws Exception {
2	InputStream fis = new FileInputStream(filename);
3	byte[] buffer = new byte[1024];
4	MessageDigest complete = MessageDigest.getInstance(type);
6	int numRead;
7	do {
8	numRead = fis.read(buffer);
9	if (numRead > 0) {
10	complete.update(buffer, 0, numRead);
12	}
13	} while (numRead != -1);
14	fis.close();
15	return complete.digest();
16	}
17	public String getChecksum(String filename, String type) {
18	String result = "";
19	byte[] b;
20	try {
21	b = createChecksum(filename, type);
22	for (int i = 0; i < b.length; i++) {
23	result += Integer.toString((b[i] & 0xff) + 0x100, 16).substring(1);
24	}
25	} catch (Exception ex) {
26	JOptionPane.showMessageDialog(null, "GAGAL cek CheckSum");
27	}
28	return result;
29	}
30	public void getMetaData(File f, String namafile) {
31	tModel = new DefaultTableModel(header, 0);
32	Path file = Paths.get(f.getAbsolutePath());
33	try {
34	String checksum = "MD5 :\n" + getChecksum(f.getPath(), "MD5") + "\nSHA-256 :\n" +
35	getChecksum(f.getPath(), "SHA-256");
36	txtSUM.setText(checksum);
37	} catch (IOException ex) {
38	}
39	tabelMeta.setModel(tModel);
40	}

Pada baris 1 sampai dengan 16 *source coding untuk* mencari nilai MD5 dan pada baris 17 sampai dengan 29 *source coding untuk* mencari nilai SHA-256. Pada baris 30 untuk memvalidasi bahwa method ini diakses dan mengembalikan nilai yang dikirim dari class Main, pada baris 31 untuk menaruh nilai yang dihasilkan pada sebuah tabel, pada baris 32 akan selalu mengembalikan pemisah direktori sesuai dengan platform pengguna, pada baris 34 dan 35 untuk memanggil nilai Checksum MD5 dan SHA-256 yang ada pada baris 1 sampai dengan 16 dan baris 17 sampai dengan 29, pada baris 36 mendifinisikan metadata Checksum MD5 dan SHA-256, pada baris 37 untuk *exception* yang berhubungan dengan *input* dan *output*, pada baris 39 di *set* langsung pada *interface* tabel metadata.

4.2.2 Source Code Korelasi Metadata File

1. *Source Code* Korelasi Berdasarkan Tanggal File (*Date*)

Tabel 4. 5 *Source Code* Korelasi Berdasarkan Tanggal File (*Date*)

No.	Source Code
1	public void getHasilLastModified(String dirPath) {
2	try{
3	File dir = new File(dirPath);
4	File[] files = dir.listFiles(tglFilter);
5	if (files.length == 0) {
6	} else {
7	for (File aFile : files) {
8	String data[] = {aFile.getName(), String.valueOf(aFile.length()), sdf.format(new Date(aFile.lastModified())), aFile.getPath()};
9	tModelHasil.addRow(data);
10	}
11	tabelHasil.setModel(tModelHasil);
12	}
13	}catch(Exception ex){
14	}
15	}
16	FileFilter tglFilter = new FileFilter() {
17	public boolean accept(File file) {
18	if (file.isFile()) {
19	if (file.lastModified()== lastmodifide && perbandingan == 1) {
20	return true;
21	} else if (file.lastModified()< lastmodifide && perbandingan == 2) {
22	return true;
23	} else if (file.lastModified()> lastmodifide && perbandingan == 3) {

Lanjutan **Tabel 4.5** *Source Code* Korelasi Berdasarkan Tanggal File (*Date*)

No.	Source Code
24	return true;
25	} else if (file.lastModified()<= lastmodifide && perbandingan == 4) {
26	return true;
27	} else if (file.lastModified()>= lastmodifide && perbandingan == 5) {
28	return true;
29	} else {
30	return false;
31	}
32	} else {
33	return false;
34	}
35	}
36	}

Pada baris 1 untuk validasi hasil metadata lastModified yang dijadikan sebagai acuan korelasi metadata file, pada baris 3 berfungsi untuk melihat daftar file/folder yang berada di direktori atau folder tertentu, pada baris 4 direktori file yang dilihat berdasarkan tanggalnya, pada baris 5 sampai 8 sebuah kondisi yang akan menemukan file tertentu berdsarkan tanggalnya, pada baris 9 dan 11 direktori file yang di temukan akan memiliki baris dan di set langsung pada tabel korelasi di *interface*, pada baris 13 untuk *exception* yang berhubungan dengan *input* dan *output*, pada baris 16 akan memfilter file-file berdasarkan tanggal yang mau ditemukan, pada baris 17 validasi nilai *boolean* dipresentasikan dengan *true* untuk pernyataan bernilai benar dan *false* untuk pernyataan bernilai salah dari file-file yang akan ditemukan, pada baris 18 file yang di inputkan, pada baris 19 file-file direktori yang ditampilkan berdasarkan sama dengan file yang sudah di inputkan, pada baris 21 file-file direktori yang ditampilkan berdasarkan lebih kecil dengan file yang sudah di inputkan, pada baris 23 file-file direktori yang ditampilkan berdasarkan lebih besar dengan file yang sudah di inputkan, pada baris 25 file-file direktori yang ditampilkan berdasarkan lebih kecil sama dengan dengan file yang sudah di inputkan, pada baris 27 file-file direktori yang ditampilkan berdasarkan lebih besar sama dengan dengan file yang sudah di inputkan.

2. *Source Code* Korelasi Berdasarkan Ukuran File (*Size*)

Tabel 4. 6 *Source Code* Korelasi Berdasarkan Ukuran File (*Size*)

No.	Source Code
1	public void getHasilSize(String dirPath) {
2	try{

Lanjutan **Tabel 4. 6** *Source Code* Korelasi Berdasarkan Ukuran File (*Size*)

No.	Source Code
3	File dir = new File(dirPath);
4	File[] files = dir.listFiles(sizeFilter);
5	if (files.length == 0) {
6	} else {
7	for (File aFile : files) {
8	String data[] = {aFile.getName(), String.valueOf(aFile.length()), sdf.format(new Date(aFile.lastModified())), aFile.getPath()};
9	tModelHasil.addRow(data);
10	}
11	tabelHasil.setModel(tModelHasil);
12	}
13	}catch(Exception ex){
14	**}**
15	}
16	FileFilter sizeFilter = new FileFilter() {
17	public boolean accept(File file) {
18	if (file.isFile()) {
19	if (file.length() == size && perbandingan == 1) {
20	return true;
21	} else if (file.length() < size && perbandingan == 2) {
22	return true;
23	} else if (file.length() > size && perbandingan == 3) {
24	return true;
25	} else if (file.length() <= size && perbandingan == 4) {
26	return true;
27	} else if (file.length() >= size && perbandingan == 5) {
28	return true;
29	} else {
30	return false;
31	}
32	} else {
33	return false;
34	}
35	}
36	}

Pada baris 1 untuk validasi hasil metadata size yang dijadikan sebagai acuan korelasi metadata file, pada baris 3 berfungsi untuk melihat daftar file/folder yang berada di direktori atau folder tertentu, pada baris 4 direktori file yang dilihat berdsarkan ukuran filenya, pada baris 5

sampai 8 sebuah kondisi yang akan menemukan file tertentu berdasarkan ukuran filenya, pada baris 9 dan 11 direktori file yang di temukan akan memiliki baris dan di set langsung pada tabel korelasi di *interface*, pada baris 13 untuk *exception* yang berhubungan dengan *input* dan *output*, pada baris 16 akan memfilter file-file berdasarkan ukuran file yang mau ditemukan, pada baris 17 validasi nilai *boolean* dipresentasikan dengan *true* untuk pernyataan bernilai benar dan *false* untuk pernyataan bernilai salah dari file-file yang akan ditemukan, pada baris 18 file yang di inputkan, pada baris 19 file-file direktori yang ditampilkan berdasarkan sama dengan file yang sudah di inputkan, pada baris 21 file-file direktori yang ditampilkan berdasarkan lebih kecil dengan file yang sudah di inputkan, pada baris 23 file-file direktori yang ditampilkan berdasarkan lebih besar dengan file yang sudah di inputkan, pada baris 25 file-file direktori yang ditampilkan berdasarkan lebih kecil sama dengan dengan file yang sudah di inputkan, pada baris 27 file-file direktori yang ditampilkan berdasarkan lebih besar sama dengan dengan file yang sudah di inputkan.

3. *Source Code* Korelasi Berdasarkan Ektensi File (*Type File*)

Tabel 4. 7 *Source Code* Korelasi Berdasarkan Ektensi File *(Type File*)

No.	Source Code
1	FileFilter extensi = new FileFilter() {
2	public boolean accept(File file) {
3	if (file.isFile()) {
4	if (getFileExtension(file).equalsIgnoreCase(txtTypeFile.getText().trim())) {
5	return true;
6	} else {
7	return false;
8	}
9	} else {
10	return false;
11	}
12	}
13	};
14	private String getFileExtension(File file) {
15	String name = file.getName();
16	try {
17	return name.substring(name.lastIndexOf(".") + 1);
18	} catch (Exception e) {
19	return "";
20	}

Pada baris 1 untuk filter hasil metadata yang khusus ektensi file, pada baris 2 validasi nilai *boolean* di presentasikan dengan *true* untuk pernyataan bernilai benar dan *false* untuk pernyataan bernilai salah dari file-file yang akan ditemukan, pada baris 3 sebuah kondisi yang akan menemukan file tertentu berdasarkan ektensi filenya, pada baris 4 dan 5 jika file-file yang ditemukan berdasarkan file ektensi maka akan ditampilkan, pada baris 6 dan 7 jika tidak ditemukan berdasarkan file ektensi maka tidak akan ditampilkan, begitu juga dengan baris 9 dan 10. Pada baris 14 untuk validasi hasil metadata ektensi file yang dijadikan sebagai acuan korelasi metadata file, pada baris 15 direktori file yang dilihat berdasarkan ektensi filenya, pada baris 17 memilih file ektensi yang dicari berdasarkan dari nilai metadata yang sudah ditemukan, pada baris 18 untuk *exception* yang berhubungan dengan *input* dan *output*.

4. *Source Code* Korelasi Berdasarkan Pemilik File (*Owner*)

Tabel 4. 8 *Source Code* Korelasi Berdasarkan Pemilik File (*Owner*)

No.	Source Code
1	FileFilter author = new FileFilter() {
2	public boolean accept(File file) {
3	FileOwnerAttributeView ownerAttributeView =
4	Files.getFileAttributeView(file.toPath(), FileOwnerAttributeView.class);
5	UserPrincipal owner = null;
6	try {
7	owner = ownerAttributeView.getOwner();
8	String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
9	if (file.isFile()) {
10	if (dd.equalsIgnoreCase(txtAuthors.getText().trim())) {
11	return true;
12	} else {
13	return false;
14	}
15	} else {
16	return false;
17	}
18	} catch (IOException ex) {
19	return false;
20	}
21	}
22	}

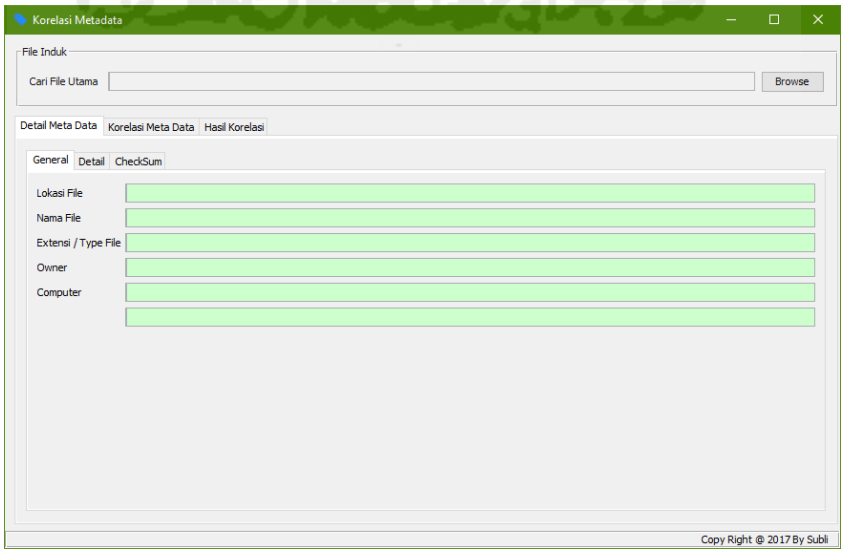
Pada baris 1 untuk filter hasil metadata yang khusus Owner atau pemilik file, pada baris 2 validasi nilai *boolean* di presentasikan dengan *true* untuk pernyataan bernilai benar dan *false* untuk

pernyataan bernilai salah dari file-file yang akan ditemukan, pada baris 3 dan 4 atribut yang di definisikan oleh *interface* untuk menemukan Owner file, pada baris 5 akan diseleksi pencarian dan yang ditampilkan berdasarkan owner file saja dan yang lainnya tidak akan ditampilkan, pada baris 7 direktori file yang dilihat berdasarkan owner filenya, pada baris 8 memilih file ektensi yang dicari berdasarkan dari nilai metadata yang sudah ditemukan, pada baris 9 sebuah kondisi yang akan menemukan file tertentu berdasarkan Owner filenya, pada baris 10 dan 11 5 jika file-file yang ditemukan berdasarkan file owner maka akan ditampilkan, pada baris 12 dan 13 jika tidak ditemukan berdasarkan file ektensi maka tidak akan ditampilkan, begitu juga dengan baris 15 dan 16, pada baris 18 untuk *exception* yang berhubungan dengan *input* dan *output*.

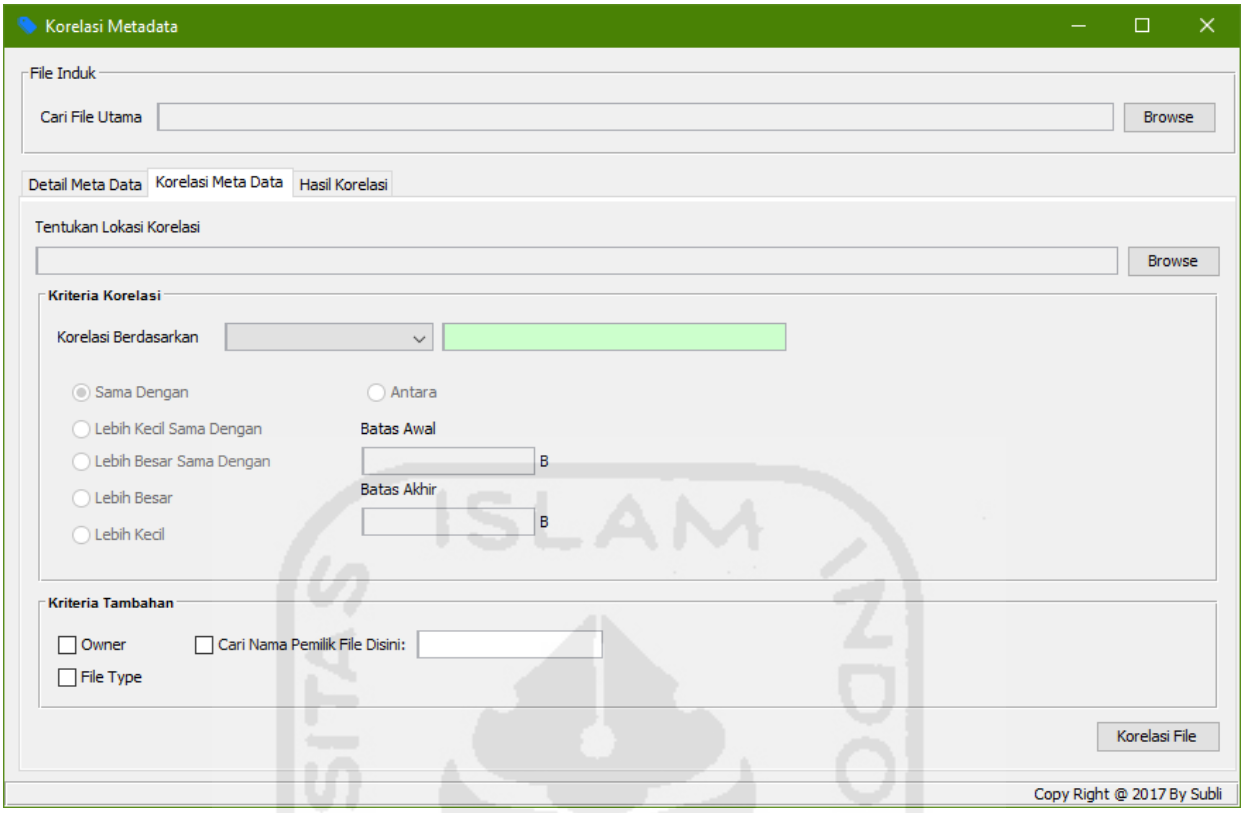
4.3 Pengujian Sistem Metadata Forensik

4.3.1 Awal Program

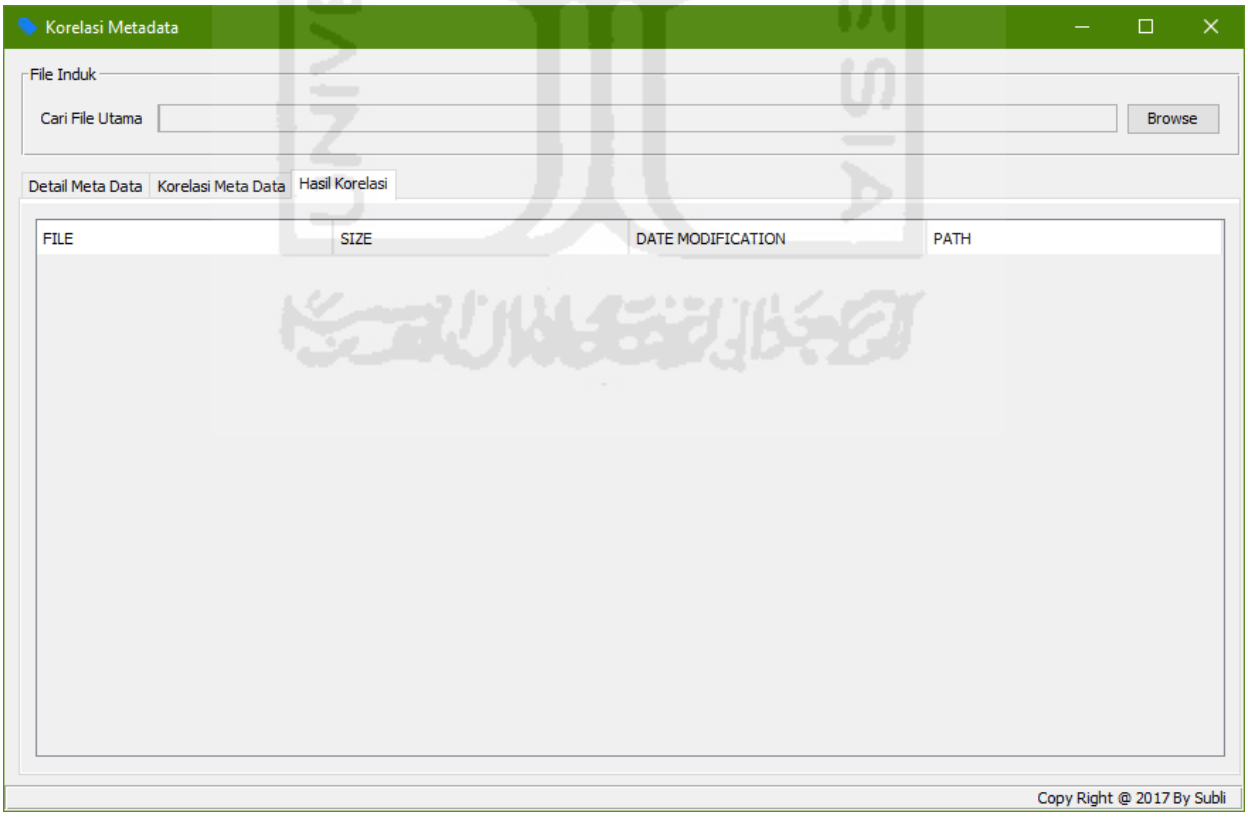
Tampilan awal program sistem metadata forensik ketika baru dijalankan yaitu akan menampilakan Menu File Induk (untuk mencari file utama yang akan di browse) dan Menu Detail Metadata (yang didalamnya ada Menu General untuk melihat metadata file secara umum, Menu Detail untuk melihat metadata file secara lebih jelas lagi dan Menu Checksum untuk melihat nilai hashing suatu file), selanjutnya ketika di select menu berikutnya yatitu Menu Korelasi Metadata (yang didalamnya akan menentukan lokasi tempat pencarian file dan pilihan korelasi file dari empat macam parameter metadata file yaitu tanggal, ukuran, ektensi file dan nama pemilik file) dan Menu Hasil Korelasi (tempat yang sudah disediakan untuk file-file yang sudah ditemukan, berdasarkan pilihan dari menu korelasi file yang sudah ditentukan). Berikut adalah tempilan awal dari ketiga menu masing-masing program ketika baru dijalankan, seperti gambar dibawah ini:



Gambar 4. 10 Tampilan Awal Program Menu Detail Metadata



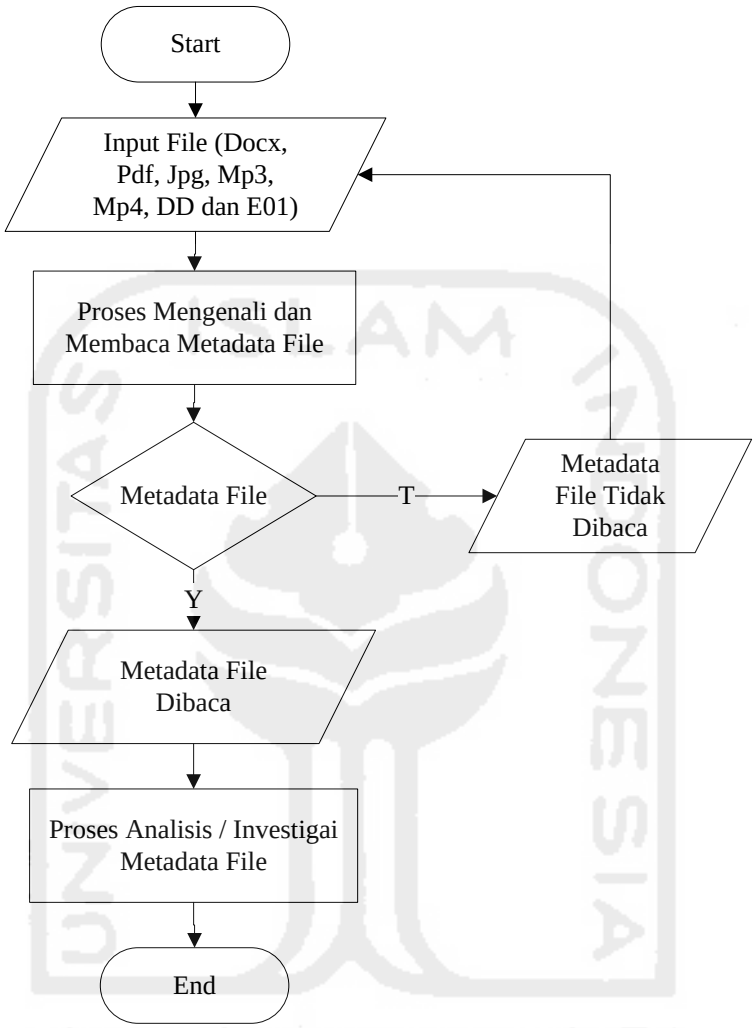
Gambar 4. 11 Tampilan Awal Program Menu Korelasi Metadata



Gambar 4. 12 Tampilan Awal Program Menu Hasil Korelasi

4.3.2 Melihat Karakteristik Metadata File

Berikut dijelaskan secara rinci langkah-langkah penggunaan program aplikasi ini dalam melihat karakteristik metadata file pada gambar *flowchart* dibawah:

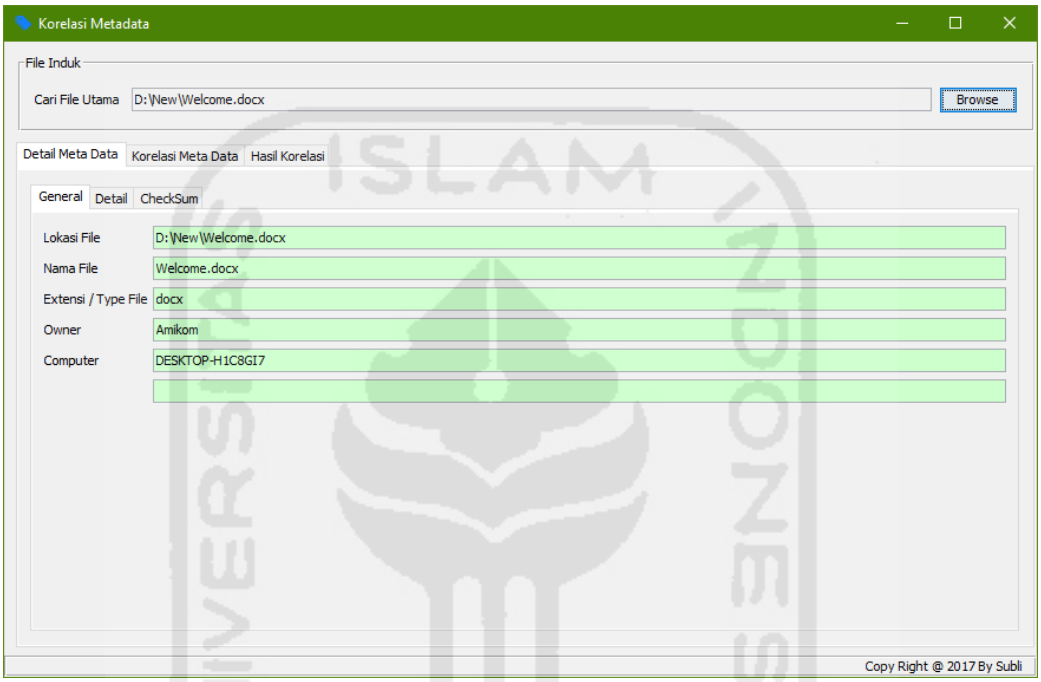


Gambar 4. 13 Flowchart Membaca Karakteristik Metadata File

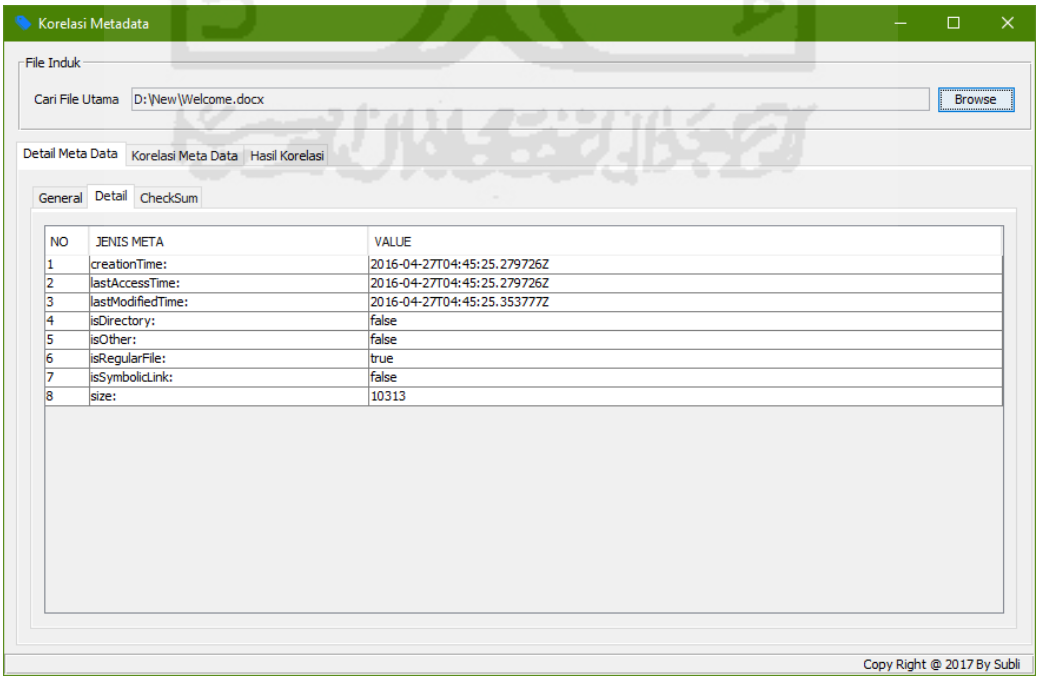
Berikut penjelasan gambar 4.12 *flowchart* dalam pengujian membaca karakteristik metadata file:

1. Pertama-tama dilakukan *start* atau sistem dijalankan
2. Setelah itu dilakukan penginputan file yang akan di baca atau di kenali metadatanya, dimana file yang akan dibaca yaitu file ber-extension Docx, Pdf, Jpg, Mp3, Mp4, DD dan E01
3. Kemudian program akan melakukan pemrosesan file yang telah di inputkan, terdapat kondisi, dimana metadata file yang tidak bisa dibaca akan kembali ke inputan file objek, tetapi metadata file yang dapat terbaca akan langsung ditampilkan metadata filenya
4. Selanjutnya dilakukan sebuah analisis / investigasi terhadap metadata file yang sudah dibaca, dan
5. Terakhir program di tutup atau selesai di jalankan

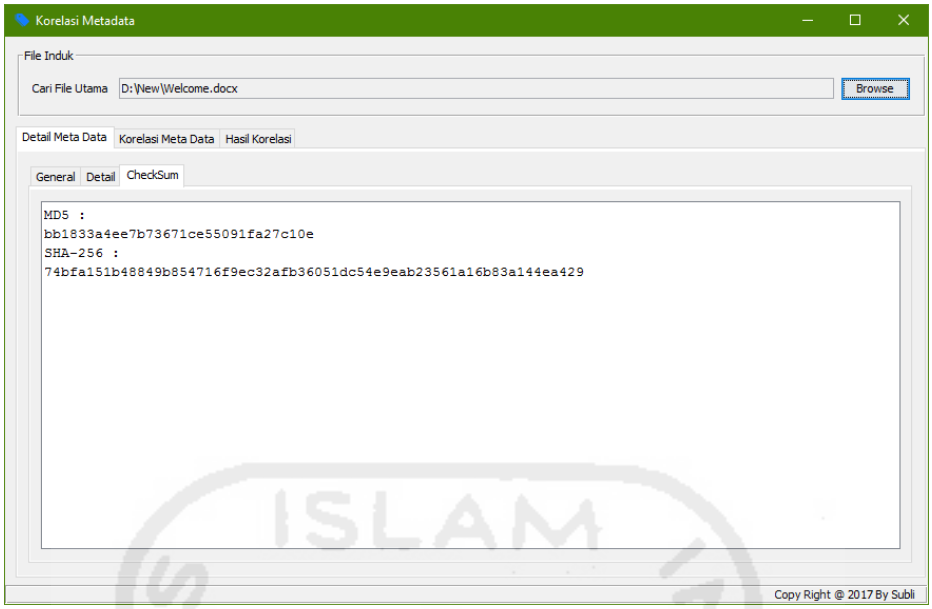
File yang mau di baca metadatanya, terlebih dahulu akan di browse atau di cari sebuah file yang akan diperiksa metadatanya, setelah itu baru kemudian program akan memproses file tersebut sampai ter-identifikasi metadatanya satu persatu, kemudian akan dimunculkan keterangan metadatanya di tabel detail metadata, seperti kita mencoba mem-browse sebuah file Welcome.docx yang ada didalam Folder New Data D Komputer, hasilnya bisa dilihat seperti gambar dibawah ini:



Gambar 4. 14 Metadata General



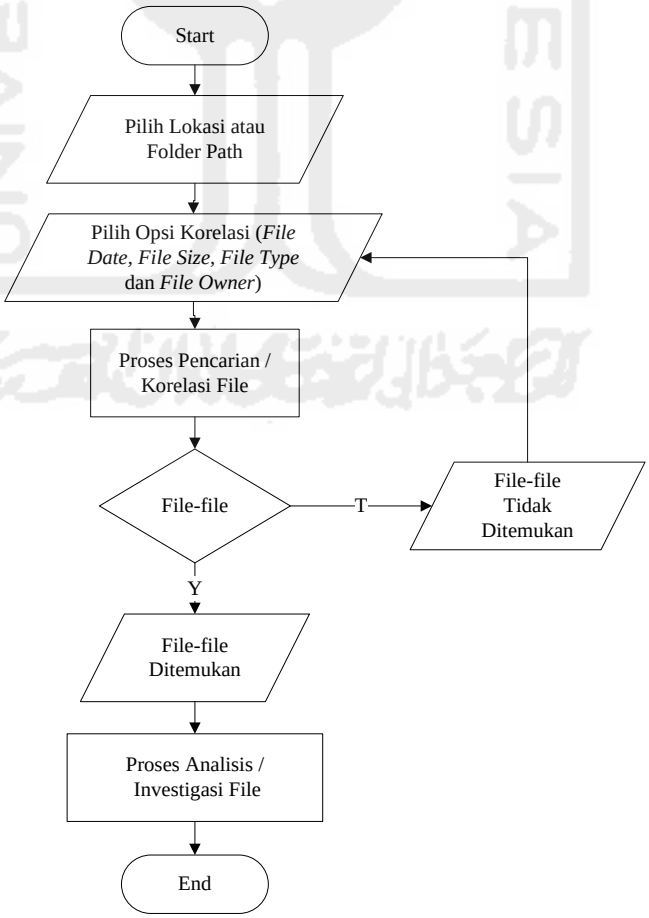
Gambar 4. 15 Metadata Detail



Gambar 4. 16 Metadata Checksum

4.3.3 Melakukan Korelasi File

Berikut dijelaskan secara rinci langkah-langkah penggunaan program aplikasi ini untuk melakukan korelasi file dalam gambar *flowchart* dibawah:



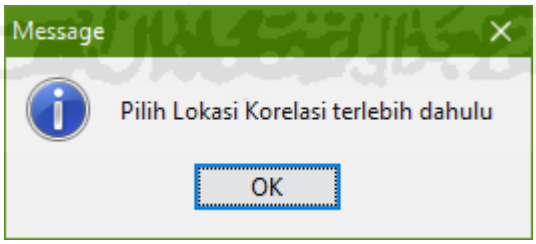
Gambar 4. 17 Flowchart Melakukan Korelasi Metadata File

Berikut penjelasan gambar 4.16 *flowchart* dalam pengujian melakukan korelasi metadata file:

1. Pertama-tama dilakukan *start* atau sistem yang sudah dijalankan tinggal menunggu perintah mulai lagi
2. Dilakukan penginputan lokasi korelasi terlebih dahulu (Data C, Data D atau Data E) atau Foder Path yang ada didalam komputer
3. Setelah itu dilakukan pemilihan metadata file berdasarkan korelasi parameter dari *File Date*, *File Size*, *File Type* dan *File Owner*, kemudian sistem akan melakukan proses menemukan korelasi metadata file yang telah dibuat
4. Terdapat pernyataan atau kondisi dimana terdapat banyak file-file, jika file-file masih belum ditemukan korelasi metadatanya maka sistem akan kembali memilih opsi korelasi metadata file seperti biasa, tetapi apabila file-file sudah ditemukan dari korelasi metadata filenya berdasarkan parameter yang telah dibangun maka akan dilanjutkan ke analisis / investigasi file-file yang sudah ditemukan tersebut, dan
5. Terakhir sistem ini selesai digunakan dan ditutup.

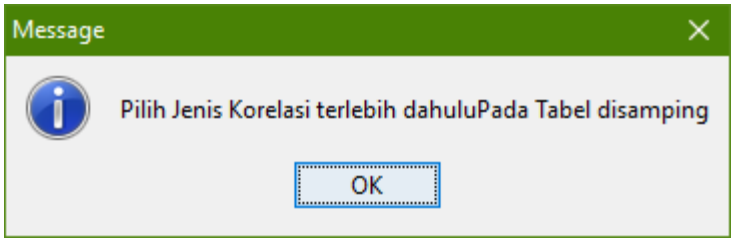
Untuk hasil korelasi metadata file itu sendiri, ada empat jenis yang ditampilkan dari file yaitu Nama sebuah file (*File Name*),Ukuran dari file (*Size*),Tanggal suatu file (*Date*) dan Tempat lokasi atau folder dari suatu file (*Path*).

Selanjutnya untuk melihat korelasi metadata, silahkan browse dulu letaknya, apakah mau di data mana (Data C / Data D) atau di folder mana di setiap data (Data C / Data D). Jika belum di browse dan langsung klik button Korelasi File, maka muncul pesan Pilih Lokasi Korelasi terlebih dahulu, hasilnya seperti gambar berikut:



Gambar 4. 18 Pilih Lokasi Korelasi

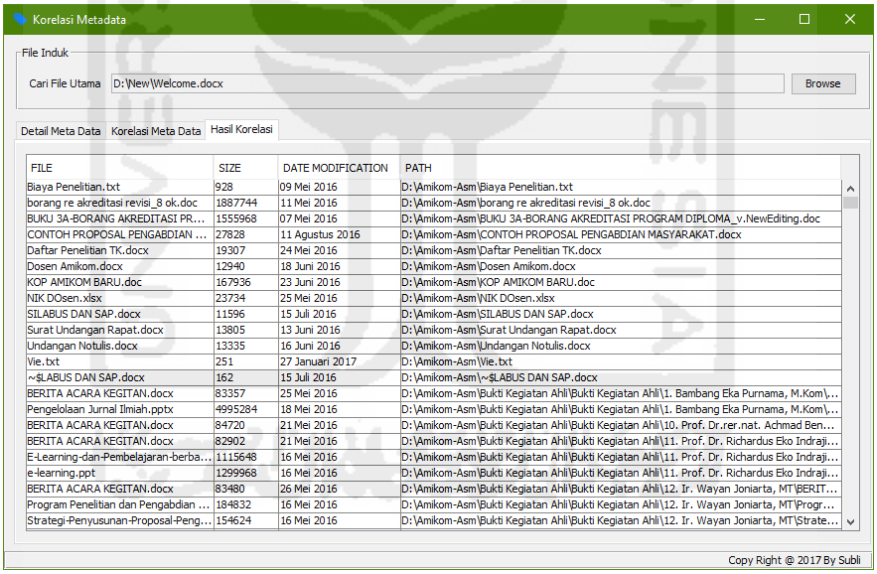
Jika sudah di browse lokasi korelasi metadata yang mau di lihat dan langsung klik button Korelasi File, maka muncul pesan Pilih Jenis Korelasi dahulu pada tabel disamping (parameter korelasi yang mau digunakan yaitu bisa tanggal/date, ukuran/size, ektensi/type atau pemilik/owner), hasilnya seperti gambar berikut:



Gambar 4. 19 Pilih Jenis Korelasi

1. Korelasi Berdasarkan Tanggal (*File Date*)

Jika sudah di pilih salah satunya, misal tanggal dan klik option file yang mau di munculin, apakah tanggalnya **Lebih Besar** dari file Welcome.docx yang sudah di browse, baru kemudian di klik button Korelasi File, maka file-file yang ada di Data D yang tanggalnya lebih besar dari file Welcome.docx akan segera di cari oleh sistem metadata forensik ini, setelah menunggu beberapa saat, maka akan ditemukan banyak sekali file-file yang ada di dalam folder-folder di Data D dan hasilnya seperti gambar berikut:

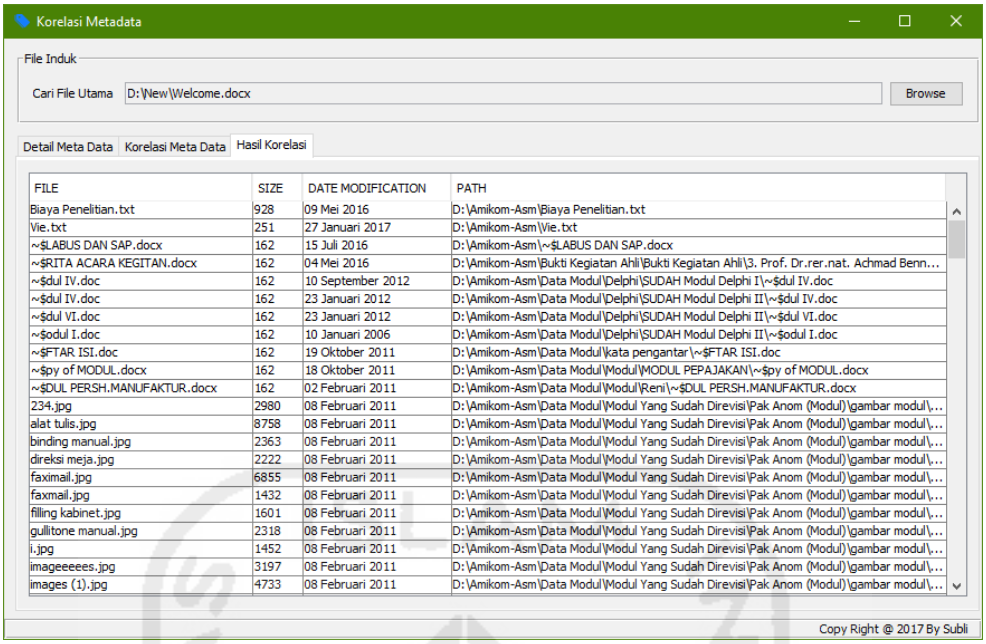


FILE	SIZE	DATE MODIFICATION	PATH
Biaya Penelitian.txt	928	09 Mei 2016	D:\Amikom-Asm\Biaya Penelitian.txt
borang re akreditasi revisi_8 ok.doc	1887744	11 Mei 2016	D:\Amikom-Asm\borang re akreditasi revisi_8 ok.doc
BUKU 3A-BORANG AKREDITASI PR...	1555968	07 Mei 2016	D:\Amikom-Asm\BUKU 3A-BORANG AKREDITASI PROGRAM DIPLOMA_v.NewEditing.doc
CONTOH PROPOSAL PENGABDIAN ...	27828	11 Agustus 2016	D:\Amikom-Asm\CONTOH PROPOSAL PENGABDIAN MASYARAKAT.docx
Daftar Penelitian TK.docx	19307	24 Mei 2016	D:\Amikom-Asm\Daftar Penelitian TK.docx
Dosen Amikom.docx	12940	18 Juni 2016	D:\Amikom-Asm\Dosen Amikom.docx
KOP AMIKOM BARU.doc	167936	23 Juni 2016	D:\Amikom-Asm\KOP AMIKOM BARU.doc
NIK DOsen.xlsx	23734	25 Mei 2016	D:\Amikom-Asm\NIK DOsen.xlsx
SILABUS DAN SAP.docx	11596	15 Juli 2016	D:\Amikom-Asm\SILABUS DAN SAP.docx
Surat Undangan Rapat.docx	13805	13 Juni 2016	D:\Amikom-Asm\Surat Undangan Rapat.docx
Undangan Notulis.docx	13335	16 Juni 2016	D:\Amikom-Asm\Undangan Notulis.docx
Vie.txt	251	27 Januari 2017	D:\Amikom-Asm\Vie.txt
~$LABUS DAN SAP.docx	162	15 Juli 2016	D:\Amikom-Asm\~$LABUS DAN SAP.docx
BERITA ACARA KEGITAN.docx	83357	25 Mei 2016	D:\Amikom-Asm\Bukti Kegiatan Ahli\Bukti Kegiatan Ahli\1. Bambang Eka Purnama, M.Kom\...
Pengelolaan Jurnal Ilmiah.pptx	4995284	18 Mei 2016	D:\Amikom-Asm\Bukti Kegiatan Ahli\Bukti Kegiatan Ahli\1. Bambang Eka Purnama, M.Kom\...
BERITA ACARA KEGITAN.docx	84720	21 Mei 2016	D:\Amikom-Asm\Bukti Kegiatan Ahli\Bukti Kegiatan Ahli\10. Prof. Dr.rer.nat. Achmad Ben...
BERITA ACARA KEGITAN.docx	82902	21 Mei 2016	D:\Amikom-Asm\Bukti Kegiatan Ahli\Bukti Kegiatan Ahli\11. Prof. Dr. Richardus Eko Indraji...
E-Learning-dan-Pembelajaran-berba...	1115648	16 Mei 2016	D:\Amikom-Asm\Bukti Kegiatan Ahli\Bukti Kegiatan Ahli\11. Prof. Dr. Richardus Eko Indraji...
e-learning.ppt	1299968	16 Mei 2016	D:\Amikom-Asm\Bukti Kegiatan Ahli\Bukti Kegiatan Ahli\11. Prof. Dr. Richardus Eko Indraji...
BERITA ACARA KEGITAN.docx	83480	26 Mei 2016	D:\Amikom-Asm\Bukti Kegiatan Ahli\Bukti Kegiatan Ahli\12. Ir. Wayan Joniarta, MT\BERIT...
Program Penelitian dan Pengabdian ...	184832	16 Mei 2016	D:\Amikom-Asm\Bukti Kegiatan Ahli\Bukti Kegiatan Ahli\12. Ir. Wayan Joniarta, MT\Progr...
Strategi-Penyusunan-Proposal-Peng...	154624	16 Mei 2016	D:\Amikom-Asm\Bukti Kegiatan Ahli\Bukti Kegiatan Ahli\12. Ir. Wayan Joniarta, MT\Strate...

Gambar 4. 20 Hasil Korelasi Tanggal

2. Korelasi Berdasarkan Ukuran (*File Size*)

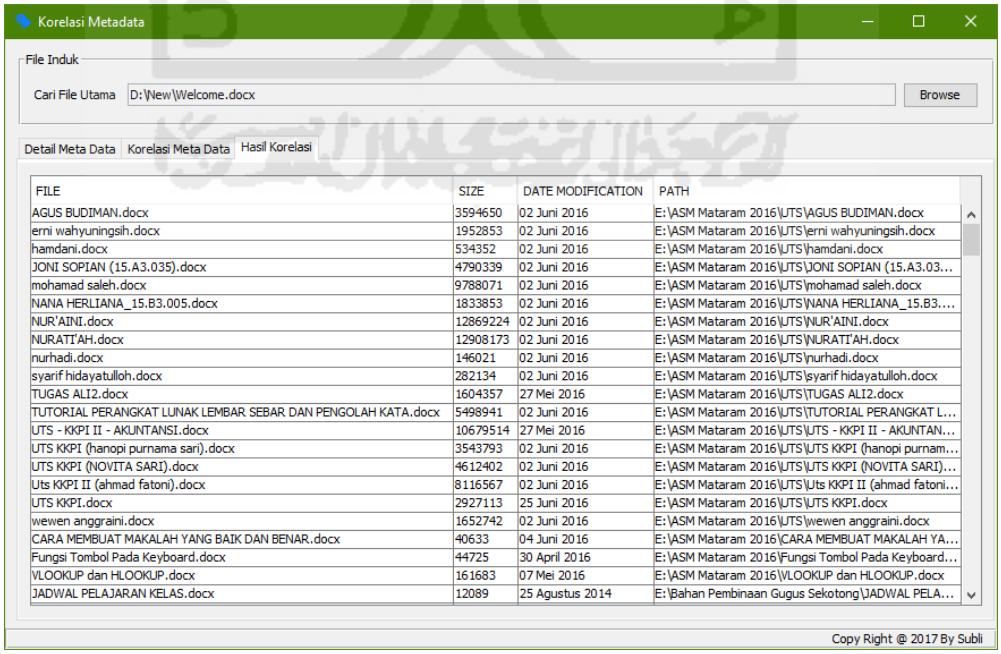
Sama seperti diatas, jika yang di pilih adalah size atau ukuran filenya, dan dibuat option yang dimunculin adalah **Lebih Kecil** dari file Welcome.docx yang sudah di browse, baru kemudian di klik button Korelasi File, maka file-file yang ada di Data D yang ukurannya lebih kecil dari file Welcome.docx akan segera di cari oleh sistem metadata forensik ini, kemudian menunggu beberapa saat, maka akan ditemukan banyak sekali file-file yang ada di dalam folder-folder di Data D dan hasilnya seperti gambar berikut:



Gambar 4. 21 Hasil Korelasi Ukuran

3. Korelasi Berdasarkan Ektensi (*File Type*)

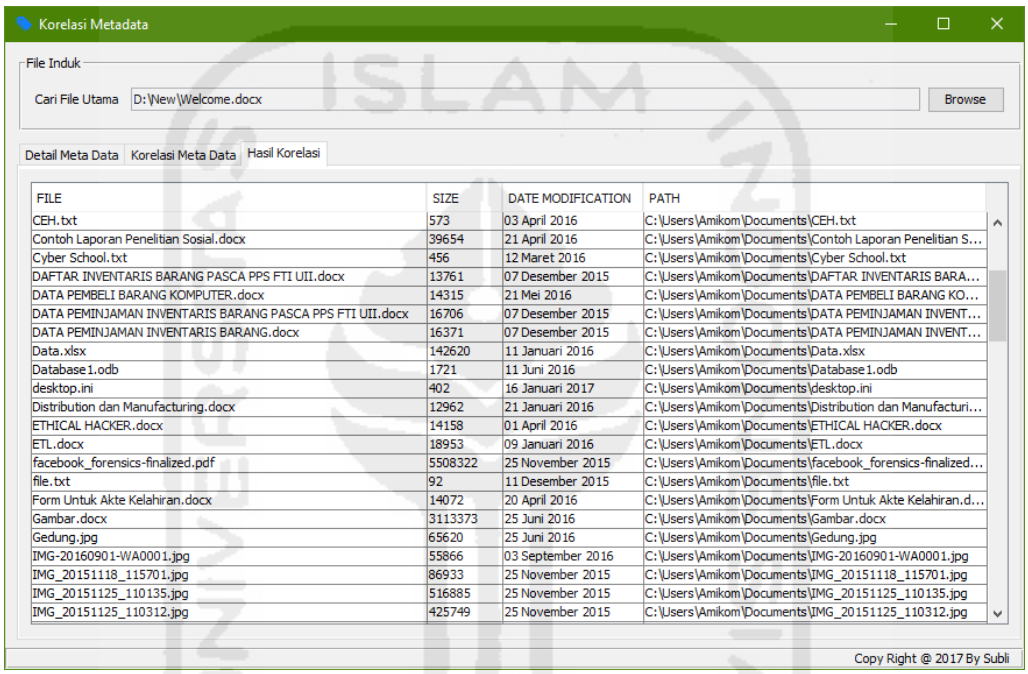
Untuk melihat hasil pencarian berdasarkan korelasi ektensi file yaitu pilih lokasi korelasi dulu, misal di Data E dan langsung dipilih option **File Type**, maka file Welcome.docx yang sudah di browse yang berektensi **.docx** akan segera di cari oleh sistem metadata forensik ini, setelah menunggu beberapa saat, maka akan ditemukan banyak sekali file-file berektensi docx yang ada di dalam folder-folder di Data E dan hasilnya seperti gambar berikut:



Gambar 4. 22 Hasil Korelasi Ektensi File

4. Korelasi Berdasarkan Pemilik (*File Owner*)

Selanjutnya untuk melihat hasil pencarian berdasarkan korelasi pemilik file yaitu pilih lokasi korelasi dulu, misalnya di Documents yang ada di Data C dan langsung dipilih option **Owner**, maka file Welcome.docx yang sudah di browse sebelumnya yang pemilik filenya bernama **Amikom** akan segera di cari oleh sistem metadata forensik ini, kemudian menunggu beberapa saat, maka akan ditemukan banyak sekali file-file atas nama pemilik Amikom yang ada di dalam folder-folder yang di Documents Data C dan hasilnya bisa dilihat seperti gambar berikut:



Gambar 4. 23 Hasil Korelasi Pemilik File

4.4 Analisis Hasil Sistem Metadata Forensik

Untuk melihat sejauh mana hasil kemampuan sistem yang dibangun, perlu dilihat dan dianalisa hasil pengujian metode tersebut. Berikut adalah hasil pengujian dan analisa sistem yang sudah di uji cobakan.

4.4.1 Analisis Hasil Membaca Karakterisitik Metadata File

1. File Dokumen ber-Extensi DOCX

Nama file yang di uji coba dengan sistem metadata forensik ini adalah file Welcome.docx yang berlokasi didalam Folder New yang ada di Data D, kemudian dari pengujian tersebut di dapat hasil analisa metadatanya, berikut bisa di lihat di tabel berikut:

Tabel 4. 9 Hasil Membaca Metadata File Dokumen Welcome.docx

No.	Jenis Metadata	Value
1	Folder Path	D:\New\Welcome.docx
2	Name File	Welcome.docx
3	Type File	docx
4	Owner	Amikom
5	Computer	DESKTOP-H1C8GI7
6	Creation Time	2016-04-27T04:45:25.279726Z
7	Last Access Time	2016-04-27T04:45:25.279726Z
8	Last Modified Time	2016-04-27T04:45:25.353777Z
9	Is Directory	false
10	Is Other	false
11	Is Regular File	true
12	Is Symbolic Link	false
13	Size	10313
14	Checksum MD5	bb1833a4ee7b73671ce55091fa27c10e
15	Checksum SHA-256	74bfa151b48849b854716f9ec32afb3 6051dc 54e9eab23561 a16b83a144ea429

2. File Ebook ber-Extensi PDF

Nama file yang di uji coba dengan sistem metadata forensik ini adalah file Contoh-LR-FD.pdf yang berlokasi di Folder Tesis UII yang ada di Data D, kemudian dari pengujian tersebut di dapat hasil analisa metadatanya, berikut bisa di lihat di tabel berikut:

Tabel 4. 10 Hasil Membaca Metadata File Ebook Contoh-LR-FD.pdf

No.	Jenis Metadata	Value
1	Folder Path	D:\Tesis UII\Contoh-LR-FD.pdf
2	Name File	Contoh-LR-FD.pdf
3	Type File	pdf
4	Owner	Amikom
5	Computer	DESKTOP-H1C8GI7
6	Creation Time	2016-09-18T15:42:27.799967Z
7	Last Access Time	2016-09-18T15:42:27.799967Z
8	Last Modified Time	2016-06-23T02:58:32Z
9	Is Directory	false
10	Is Other	false
11	Is Regular File	true

Lanjutan **Tabel 4. 10** Hasil Membaca Metadata File Ebook Contoh-LR-FD.pdf

No.	Jenis Metadata	Value
12	Is Symbolic Link	false
13	Size	3290304
14	Checksum MD5	35a66b998cd21306211cdbdcf23419c3
15	Checksum SHA-256	fad9bdc81b174492304ea6f5753646f65eda936f623 d05e0fbb8aa3c59f5ad04

3. File Gambar ber-Extensi JPG

Nama file yang di uji coba dengan sistem metadata forensik ini adalah file Subli.jpg yang berlokasi di Folder Foto yang ada di Data E, kemudian dari pengujian tersebut di dapat hasil analisa metadatanya, berikut bisa di lihat di tabel berikut:

Tabel 4. 11 Hasil Membaca Metadata File Gambar Subli.jpg

No.	Jenis Metadata	Value
1	Folder Path	E:\Foto\Subli.jpg
2	Name File	Subli.jpg
3	Type File	jpg
4	Owner	S-1-5-21-2838851640-846236458-583387153-1001
5	Computer	DESKTOP-H1C8GI7
6	Creation Time	2015-10-08T03:00:25.719788Z
7	Last Access Time	2015-10-08T03:00:25.719788Z
8	Last Modified Time	2013-05-27T14:05:33.458215Z
9	Is Directory	false
10	Is Other	false
11	Is Regular File	true
12	Is Symbolic Link	false
13	Size	5117665
14	Checksum MD5	3858ca9b670e9d4577465ab2774a0fa6
15	Checksum SHA-256	26b1b0b0e52e085e30e7e149a2940fb3f198d44eddf \40f822 e201b99c511fb94

4. File Audio ber-Extensi MP3

Nama file yang di uji coba dengan sistem metadata forensik ini adalah file Opick-Alhamdulillah.mp3 yang berlokasi di Folder Mp3\ISLAMI\Opick yang ada di Data E, kemudian dari pengujian tersebut di dapat hasil analisa metadatanya, berikut bisa di lihat di tabel berikut:

Tabel 4. 12 Hasil Membaca Metadata File Audio Opick-Alhamdulillah.mp3

No.	Jenis Metadata	Value
1	Folder Path	E:\Mp3\ISLAMI\Opick\Opick-Alhamdulillah.mp3
2	Name File	Opick-Alhamdulillah.mp3
3	Type File	mp3
4	Owner	S-1-5-21-2838851640-846236458-583387153-1001
5	Computer	DESKTOP-H1C8GI7
6	Creation Time	2014-06-11T08:24:51.186661Z
7	Last Access Time	2014-06-11T08:24:51.186661Z
8	Last Modified Time	2006-06-07T10:32:06Z
9	Is Directory	false
10	Is Other	false
11	Is Regular File	true
12	Is Symbolic Link	false
13	Size	1850203
14	Checksum MD5	edc517e19bb627b7b4804f057b577ae3
15	Checksum SHA-256	952594a2b8864bc1974fa11dd951dba25831dd4 cb657d989 32e7d309377e8ae0

5. File Video ber-Extensi MP4

Nama file yang di uji coba dengan sistem metadata forensik ini adalah file Dewa19-LaskarCinta.mp4 yang berlokasi di Folder MP4 yang ada di Data E, kemudian dari pengujian tersebut di dapat hasil analisa metadatanya, berikut bisa di lihat di tabel berikut:

Tabel 4. 13 Hasil Membaca Metadata File Video Dewa19-LaskarCinta.mp4

No.	Jenis Metadata	Value
1	Folder Path	E:\MP4\Dewa19-LaskarCinta.mp4
2	Name File	Dewa19-LaskarCinta.mp4
3	Type File	mp4
4	Owner	S-1-5-21-2838851640-846236458-583387153-1001
5	Computer	DESKTOP-H1C8GI7
6	Creation Time	2015-05-20T09:36:56.003425Z
7	Last Access Time	2015-05-21T02:40:53.715456Z
8	Last Modified Time	2015-05-20T10:32:02.647291Z
9	Is Directory	false
10	Is Other	false
11	Is Regular File	true
12	Is Symbolic Link	false

Lanjutan **Tabel 4. 13** Hasil Membaca Metadata File Video Dewa19-LaskarCinta.mp4

No.	Jenis Metadata	Value
13	Size	17522975
14	Checksum MD5	d3a0a69207a387db31df0a994c1d203a
15	Checksum SHA-256	74afeff05741861c91bae970d7cbc9dfef03588a8 5760afdfe0468bb2ca7ea55

6. File Akuisisi ber-Extensi DD

Nama file yang di uji coba dengan sistem metadata forensik ini adalah file Metadata.dd yang berlokasi di Folder ProsesImaging yang ada di Data D, kemudian dari pengujian tersebut di dapat hasil analisa metadatanya, berikut bisa di lihat di tabel berikut:

Tabel 4. 14 Hasil Membaca Metadata File Akuisisi Metadata.dd

No.	Jenis Metadata	Value
1	Folder Path	D:\ProsesImaging\Metadata.dd
2	Name File	Metadata.dd
3	Type File	dd
4	Owner	Administrators
5	Computer	DESKTOP-H1C8GI7
6	Creation Time	2017-01-16T15:12:58.767359Z
7	Last Access Time	2017-01-16T15:12:58.767359Z
8	Last Modified Time	2017-01-16T15:17:13.10053Z
9	Is Directory	false
10	Is Other	false
11	Is Regular File	true
12	Is Symbolic Link	false
13	Size	3868623360
14	Checksum MD5	56ec0c24bb966d3a035f0696a1286dbb
15	Checksum SHA-256	a27a203679c3628a9a3413effa42208e48b597c60 a9d067e535 df091fa9643ab

7. File Akuisisi ber-Extensi E01

Nama file yang di uji coba dengan sistem metadata forensik ini adalah file Imaging.E01 yang berlokasi di Folder ProsesImaging yang ada di Data D, kemudian dari pengujian tersebut di dapat hasil analisa metadatanya, berikut bisa di lihat di tabel berikut:

Tabel 4. 15 Hasil Membaca Metadata File Akuisisi Imaging.E01

No.	Jenis Metadata	Value
1	Folder Path	D:\ProsesImaging\Imaging.E01
2	Name File	Imaging.E01
3	Type File	E01
4	Owner	Administrators
5	Computer	DESKTOP-H1C8GI7
6	Creation Time	2017-01-16T15:43:40.929069Z
7	Last Access Time	2017-01-16T15:43:40.929069Z
8	Last Modified Time	2017-01-16T15:48:08.291819Z
9	Is Directory	false
10	Is Other	false
11	Is Regular File	true
12	Is Symbolic Link	false
13	Size	3718986211
14	Checksum MD5	052fe2ccea953ad4e38f720f631deb4e
15	Checksum SHA-256	af6931acd22d13f98515a25186fb3f3cbd7a3505 fd4e955effe6685087488442

Untuk melihat sejauh mana kemapuan aplikasi sistem yang telah dibangun ini, metadata file yang mampu dibaca karakteristiknya tidak hanya pada sebatas tujuh jenis file diatas, tetapi mampu membaca dan mengenali karakteristik metadata jenis file lainnya juga. Berikut dibuat juga tiga macam jenis file yang sudah dibaca karakteristik metadatanya selain dari ketujuh jenis file diatas, yaitu TXT, RAR dan HTML yang dilanjutkan dengan nomor delapan sebagai berikut:

8. File Akuisisi ber-Extensi TXT

Nama file yang di uji coba dengan sistem metadata forensik ini adalah file CEH.txt yang berlokasi di Folder ProsesImaging yang ada di Documents Data C, kemudian dari pengujian tersebut di dapat hasil analisa metadatanya, berikut bisa di lihat di tabel berikut:

Tabel 4. 16 Hasil Membaca Metadata File Text CEH.txt

No.	Jenis Metadata	Value
1	Folder Path	C:\Users\Amikom\Documents\CEH.txt
2	Name File	CEH.txt
3	Type File	txt
4	Owner	Amikom
5	Computer	DESKTOP-H1C8GI7

Lanjutan **Tabel 4. 16** Hasil Membaca Metadata File Text CEH.txt

No.	Jenis Metadata	Value
6	Creation Time	2016-04-03T07:00:16.237292Z
7	Last Access Time	2016-04-03T07:00:16.368379Z
8	Last Modified Time	2016-04-03T07:00:16.368379Z
9	Is Directory	false
10	Is Other	false
11	Is Regular File	true
12	Is Symbolic Link	false
13	Size	573
14	Checksum MD5	248fd1c76cad263cba3ce019eb6dbd06
15	Checksum SHA-256	9a8b6fc593cfbd09c303ef508a4dc0eae68d3a0f 6d9948d92fa48e4b53d8b2d8

9. File Akuisisi ber-Extensi RAR

Nama file yang di uji coba dengan sistem metadata forensik ini adalah file Subli.rar yang berlokasi di Folder Tesis UII yang ada di Data D, kemudian dari pengujian tersebut di dapat hasil analisa metadatanya, berikut bisa di lihat di tabel berikut:

Tabel 4. 17 Hasil Membaca Metadata File Winrar Subli.rar

No.	Jenis Metadata	Value
1	Folder Path	D:\Tesis UII\Subli.rar
2	Name File	Subli.rar
3	Type File	rar
4	Owner	Amikom
5	Computer	DESKTOP-H1C8GI7
6	Creation Time	2017-03-14T00:31:38.257399Z
7	Last Access Time	2017-03-14T00:31:38.257399Z
8	Last Modified Time	2017-03-14T00:31:38.549597Z
9	Is Directory	false
10	Is Other	false
11	Is Regular File	true
12	Is Symbolic Link	false
13	Size	964224
14	Checksum MD5	ac843814dc81d8f75aa66f1773b5cbb8
15	Checksum SHA-256	37c011aafabe8946361fb40183470a0e24c1375 86b4f92415786e28af711ed22

10. File Akuisisi Ber-Extensi HTML

Nama file yang di uji coba dengan sistem metadata forensik ini adalah file Table.html yang berlokasi di Folder XML yang ada di Data D, kemudian dari pengujian tersebut di dapat hasil analisa metadatanya, berikut bisa di lihat di tabel berikut:

Tabel 4. 18 Hasil Membaca Metadata File HTML Table.html

No.	Jenis Metadata	Value
1	Folder Path	D:\XML\Table.html
2	Name File	Table.html
3	Type File	html
4	Owner	Amikom
5	Computer	DESKTOP-H1C8GI7
6	Creation Time	2015-12-07T02:48:32.830417Z
7	Last Access Time	2015-12-07T02:48:32.830417Z
8	Last Modified Time	2015-12-08T16:21:17.570859Z
9	Is Directory	false
10	Is Other	false
11	Is Regular File	true
12	Is Symbolic Link	false
13	Size	694
14	Checksum MD5	8ebfa04c6bbbf64b8a7967ec54546c0e
15	Checksum SHA-256	1f209514b3583b2d90af48e9c9f3b02d80702b5bf31a70a347f8881fe0d6eb37

4.4.2 Analisis Hasil Melakukan Korelasi File

1. Korelasi Berdasarkan Tanggal (*File Date*)

Option Sama Dengan

Untuk hasil metadata file yang dikorelasi yaitu file Welcome.docx yang metadata tanggalnya berupa “27 April 2016”, yang dilakukan pencarian file-file yang berlokasi di Data D dengan pilihan “Sama Dengan”, maka ditemukan hanya satu file yang tanggalnya sama dengan 27 April 2016 dari metadata tanggal file Welcome.docx yang ada di Data D tersebut. Berikut bisa di lihat hasil analisanya dari tabel 4.19 di bawah ini:

Tabel 4. 19 Hasil Korelasi File Tanggal Opsi Sama Dengan

No.	File Name	Size	Date	Path
1	Welcome.docx	10313	27 April 2016	D:\Welcome.docx

Option Lebih Besar

Untuk hasil metadata file yang dikorelasi yaitu file Welcome.docx yang metadata tanggalnya berupa "27 April 2016", yang dilakukan pencarian file-file yang berlokasi di Data D dengan pilihan "Lebih Besar", maka ditemukan bayak sekali file-file yang tanggalnya lebih besar 27 April 2016 dari metadata tanggal file Welcome.docx yang ada di Data D tersebut, tetapi dari sekian bayak file yang sudah ditemukan, diambil sepuluh sampel di jadikan sebagai analisa pencarian file berdasarkan korelasi tanggal. Berikut bisa di lihat hasil analisanya dari tabel 4.20 di bawah ini:

Tabel 4. 20 Hasil Korelasi File Tanggal Opsi Lebih Besar

No.	File Name	Size	Date	Path
1	COVER.doc	91648	28 Oktober 2016	D:\COVER.doc
2	File	38	08 September 2016	D:\File
3	File Backup.txt	26666	03 Januari 2017	D:\File Backup.txt
4	loging.log	7	05 September 2016	D:\loging.log
5	Membuat dan Membaca File dari Java.docx	202856	08 September 2016	D:\Membuat dan Membaca File dari Java.docx
6	metadata.php	261	10 September 2016	D:\metadata.php
7	TESIS PENGARUH PENERAPAN SISTEM INFORMASI MANAJEMEN.pdf	317296	26 September 2016	D:\TESIS PENGARUH PENERAPAN SISTEM INFORMASI MANAJEMEN.pdf
8	TESIS PERENCANAAN STRATEGIS SISTEM INFORMASI.pdf	1334110	17 September 2016	D:\TESIS PERENCANAAN STRATEGIS SISTEM INFORMASI.pdf

Lanjutan **Tabel 4. 20** Hasil Korelasi File Tanggal Opsi Lebih Besar

No.	File Name	Size	Date	Path
9	TESIS SISTEM INFORMASI.txt	271	26 September 2016	D:\TESIS SISTEM INFORMASI.txt
10	Buku Wisuda 2016 Hal 30-45 baru.cdr	27020269	30 Juni 2016	D:\AMIKOM BOOK \BAH TIAR\Buku Wisuda.cdr

Option Lebih Kecil

Untuk hasil metadata file yang dikorelasi yaitu file Welcome.docx yang metadata tanggalnya berupa "27 April 2016", yang dilakukan pencarian file-file yang berlokasi di Data D dengan pilihan "Lebih Kecil", maka ditemukan bayak sekali file-file yang tanggalnya lebih kecil 27 April 2016 dari metadata tanggal file Welcome.docx yang ada di Data D tersebut, tetapi dari sekian bayak file yang sudah ditemukan, diambil sepuluh sampel di jadikan sebagai analisa pencarian file berdasarkan korelasi tanggal. Berikut bisa di lihat hasil analisanya dari tabel 4.21 di bawah ini:

Tabel 4. 21 Hasil Korelasi File Tanggal Opsi Lebih Kecil

No.	File Name	Size	Date	Path
1	Absen TK.xlsx	302532	14 April 2016	D:\Amikom-Asm\Absen TK.xlsx
2	BUKU 3A PRODI TK-2016 fix-2.docx	695319	25 April 2016	D:\Amikom-Asm\BUKU 3A PRODI TK-2016 fix-2.docx
3	Contoh surat pengantar SMA.doc	481792	11 Maret 2016	D:\Amikom-Asm\Contoh surat pengantar SMA.doc
4	KOP ASM BARU.doc	550912	14 Desember 2015	D:\Amikom-Asm\KOP ASM BARU.doc
5	surat tanpa tes.docx	423143	12 Maret 2016	D:\Amikom-Asm\surat tanpa tes.docx
6	Tesis FIX - ANggun.rar	9871547	21 April 2016	D:\Amikom-Asm\Tesis FIX - ANggun.rar
7	TK SAP N TERBARU.rar	1852891	24 Maret 2016	D:\Amikom-Asm\TK SAP N TERBARU.rar
8	AMIKOM 2015-2016.xls	104960	17 Januari 2016	D:\Amikom-Asm\Absen AMIKOM\AMIKOM 2015-2016.xls
9	Absen AMIKOM-2013-2014.xls	144384	17 Februari 2016	D:\Amikom-Asm\Absen AMIKOM\Absen AMIKOM- 2013-2014.xls
10	Absen AMIKOM-2014-2015.xls	114688	17 Januari 2016	D:\Amikom-Asm\Absen AMIKOM\Absen AMIKOM- 2014-2015.xls

Option Lebih Kecil Sama Dengan

Untuk hasil metadata file yang dikorelasi yaitu file Welcome.docx yang metadata tanggalnya berupa "27 April 2016", yang dilakukan pencarian file-file yang berlokasi di Data D dengan pilihan "Lebih Kecil Sama Dengan", maka ditemukan bayak sekali file-file yang tanggalnya lebih kecil sama dengan 27 April 2016 dari metadata tanggal file Welcome.docx yang ada di Data D tersebut, tetapi dari sekian bayak file yang sudah ditemukan, diambil sepuluh sampel yang dijadikan sebagai analisa pencarian file berdasarkan korelasi tanggal. Berikut bisa di lihat hasil analisanya dari tabel 4.22 di bawah ini:

Tabel 4. 22 Hasil Korelasi File Tanggal Opsi Lebih Kecil Sama Dengan

No.	File Name	Size	Date	Path
1	file.txt	92	07 Oktober 2015	D:\New\file.txt
2	hash.py	19451	10 Oktober 2015	D:\New\hash.py
3	hashing.py	7491	10 Oktober 2015	D:\New\hashing.py
4	Hasil.E03	391248949	29 Mei 2015	D:\New\Hasil.E03
5	image.py	1830	10 Oktober 2015	D:\New\image.py
6	read.py	413	07 Oktober 2015	D:\New\read.py
7	Reference.py	1312	21 Desember 2015	D:\New\Reference.py
8	subli.jpg	5117665	27 Mei 2013	D:\New\subli.jpg
9	Tabel.py	2586	21 Desember 2015	D:\New\Tabel.py
10	Welcome.docx	10313	27 April 2016	D:\New\Welcome.docx

Option Lebih Besar Sama Dengan

Untuk hasil metadata file yang dikorelasi yaitu file Welcome.docx yang metadata tanggalnya berupa "27 April 2016", yang dilakukan pencarian file-file yang berlokasi di Data D dengan option pilihan "Lebih Besar Sama Dengan", maka ditemukan bayak sekali file-file yang tanggalnya lebih besar sama dengan 27 April 2016 dari metadata tanggal file Welcome.docx yang ada di Data D tersebut, tetapi dari sekian bayak file yang sudah ditemukan, diambil sepuluh sampel yang dijadikan sebagai analisa pencarian file berdasarkan korelasi tanggal. Berikut bisa di lihat hasil analisanya dari tabel 4.23 di bawah ini:

Tabel 4. 23 Hasil Korelasi File Tanggal Opsi Lebih Besar Sama Dengan

No.	File Name	Size	Date	Path
1	COVER.doc	91648	28 Oktober 2016	D:\New\COVER.doc
2	File	38	08 September 2016	D:\New\File
3	File Backup.txt	26666	03 Januari 2017	D:\New\File Backup.txt
4	loging.log	7	05 September 2016	D:\New\loging.log
5	Membuat dan Membaca File dari Java.docx	202856	08 September 2016	D:\New\Membuat dan Membaca File dari Java.docx
6	metadata.php	261	10 September 2016	D:\New\metadata.php
7	TESIS PENGARUH PENERAPAN SISTEM INFORMASI MANAJEMEN.pdf	317296	26 September 2016	D:\New\TESIS PENGARUH PENERAPAN SISTEM INFORMASI MANAJEMEN.pdf
8	TESIS PERENCANAAN STRATEGIS SISTEM INFORMASI.pdf	1334110	17 September 2016	D:\New\TESIS PERENCANAAN STRATEGIS SISTEM INFORMASI.pdf
9	TESIS SISTEM INFORMASI.txt	271	26 September 2016	D:\New\TESIS SISTEM INFORMASI.txt
10	Welcome.docx	10313	27 April 2016	D:\New\Welcome.docx

2. Korelasi Berdasarkan Ukuran (*File Size*)

Option Sama Dengan

Untuk hasil metadata file yang dikorelasi yaitu file Welcome.docx yang metadata ukuran filenya berupa “10.313 byte”, yang dilakukan pencarian file-file yang berlokasi di Data D dengan pilihan “Sama Dengan”, maka ditemukan hanya dua file yang ukuran filenya sama dengan 10.313 byte dari metadata ukuran file Welcome.docx yang ada di Data D tersebut. Berikut bisa dilihat hasil analisanya dari tabel 4.24 di bawah ini:

Tabel 4. 24 Hasil Korelasi File Ukuran Opsi Sama Dengan

No.	File Name	Size	Date	Path
1	7seg.lss	10313	04 November 2010	D:\Jadi Satu\Seven Segments\default\7seg.lss
2	Welcome.docx	10313	27 April 2016	D:\New\Welcome.docx

Option Lebih Besar

Untuk hasil metadata file yang dikorelasi yaitu file Welcome.docx yang metadata ukuran filenya berupa "10.313 byte", yang dilakukan pencarian file-file yang berlokasi di Data E dengan pilihan "Lebih Besar", maka ditemukan bayak sekali file-file yang ukuran filenya lebih besar 10.313 byte dari metadata ukuran file Welcome.docx yang ada di Data E tersebut, tetapi dari sekian bayak file yang sudah ditemukan, diambil sepuluh sampel yang dijadikan sebagai analisa pencarian file berdasarkan korelasi ukuran file. Berikut bisa di lihat hasil analisanya dari tabel 4.25 di bawah ini:

Tabel 4. 25 Hasil Korelasi File Ukuran Opsi Lebih Besar

No.	File Name	Size	Date	Path
1	Akhlakul Rasulullah SAW.pptx	11558646	13 Desember 2015	E:\Akhlakul Rasulullah SAW.pptx
2	IMG_20160604_174441.jpg	371024	25 September 2016	E:\IMG_20160604_174441.jpg
3	Kata-Motivasi-Hidup.jpeg	90719	25 September 2016	E:\Kata-Motivasi-Hidup.jpeg
4	Amikom-Asm Mataram.ppt	2509312	02 April 2016	E:\Amikom-Asm Mataram.ppt
5	Koperasi.accdb	520192	16 Juli 2016	E:\Koperasi.accdb
6	[1] PAI.doc	233984	20 September 2010	E:\1. PAI-Prangkat Pmbljran karakter [KTSP]\[1] PAI.doc
7	100 TOKOH.chm	1281966	02 Desember 2005	E:\AGAMA\AL-QURAN dan al-HADITS\100 TOKOH.chm
8	Abu Hanifah.mp3	86207842	11 September 2009	E:\AGAMA\BIOGRAFI 4 IMAM (MP3)\Biografi Imam Abu Hanifah rahimahullah\Abu Hanifah.mp3
9	100_tokoh yg berpengaruh dlm sejarah.chm	1281966	04 Desember 2005	E:\AGAMA\Buku Islami\100_tokoh yg berpengaruh dlm sejarah.chm
10	10 Sebab Di Cintai Allah.ppt	126976	28 Desember 2006	E:\AGAMA\dari akh Riza\10 Sebab Di Cintai Allah.ppt

Option Lebih Kecil

Untuk hasil metadata file yang dikorelasi yaitu file Welcome.docx yang metadata ukuran filenya berupa "10.313 byte", yang dilakukan pencarian file-file yang berlokasi di Data E dengan pilihan "Lebih Kecil", maka ditemukan bayak sekali file-file yang ukuran filenya lebih kecil 10.313 byte dari metadata ukuran file Welcome.docx yang ada di Data E tersebut, tetapi dari sekian bayak file yang sudah ditemukan, diambil sepuluh sampel yang dijadikan sebagai analisa pencarian file berdasarkan korelasi ukuran file. Berikut bisa di lihat hasil analisanya dari tabel 4.26 di bawah ini:

Tabel 4. 26 Hasil Korelasi File Ukuran Opsi Lebih Kecil

No.	File Name	Size	Date	Path
1	desktop.ini	129	23 November 2015	E:\$RECYCLE.BIN\S-1-5-21-316737675-2236937756-87291139-1001\desktop.ini
2	banner_islamicshop.gif	64	21 November 2006	E:\AGAMA\Keluarga Islami\Keluarga Sakinah_files\banner_islamicshop.gif
3	cvjasatama.gif	2539	21 November 2006	E:\AGAMA\Keluarga Islami\Keluarga Sakinah_files\cvjasatama.gif
4	digiquran.gif	2052	21 November 2006	E:\AGAMA\Keluarga Islami\Keluarga Sakinah_files\digiquran.gif
5	frames.js	3098	21 November 2006	E:\AGAMA\Keluarga Islami\Keluarga Sakinah_files\frames.js
6	online.gif	1102	21 November 2006	E:\AGAMA\Keluarga Islami\Keluarga Sakinah_files\online.gif
7	QuranDigital.gif	3072	21 November 2006	E:\AGAMA\Keluarga Islami\Keluarga Sakinah_files\QuranDigital.gif
8	smartquran.gif	2767	21 November 2006	E:\AGAMA\Keluarga Islami\Keluarga Sakinah_files\smartquran.gif
9	spacer.png	218	21 November 2006	E:\AGAMA\Keluarga Islami\Keluarga Sakinah_files\spacer.png
10	top_bar.jpg	589	21 November 2006	E:\AGAMA\Keluarga Islami\Keluarga Sakinah_files\top_bar.jpg

Option Lebih Kecil Sama Dengan

Untuk hasil metadata file yang dikorelasi yaitu file Welcome.docx yang metadata ukuran filenya berupa "10.313 byte", yang dilakukan pencarian file-file yang berlokasi di Data E dengan pilihan "Lebih Kecil Sama Dengan", maka ditemukan bayak sekali file-file yang ukuran filenya lebih kecil sama dengan 10.313 byte dari metadata ukuran file Welcome.docx yang ada di Data E tersebut, tetapi dari sekian bayak file yang sudah ditemukan, diambil sepuluh sampel yang dijadikan sebagai

analisa pencarian file berdasarkan korelasi ukuran file. Berikut bisa di lihat hasil analisanya dari tabel 4.27 di bawah ini:

Tabel 4. 27 Hasil Korelasi File Ukuran Opsi Lebih Kecil Sama Dengan

No.	File Name	Size	Date	Path
1	QuranDigital.gif	3072	21 November 2006	E:\AGAMA\Keluarga Islami\Keluarga Sakinah_files\QuranDigital.gif
2	smartquran.gif	2767	21 November 2006	E:\AGAMA\Keluarga Islami\Keluarga Sakinah_files\smartquran.gif
3	spacer.png	218	21 November 2006	E:\AGAMA\Keluarga Islami\Keluarga Sakinah_files\spacer.png
4	top_bar.jpg	589	21 November 2006	E:\AGAMA\Keluarga Islami\Keluarga Sakinah_files\top_bar.jpg
5	Desktop.ini	78	20 Oktober 2007	E:\AGAMA\Nasehat 1\Desktop.ini
6	Risalah Ramadhan.pdf	0	16 Juli 2012	E:\AGAMA\Ramadhan & Zakat\Risalah Ramadhan.pdf
7	1_1.gif	832	13 Juni 2003	E:\Al Qur'an\Al-quran digital\0-Al-Qur'an & Nabi\AL-QUR'AN DIGITAL\GIF\1\1_1.gif
8	1_2.gif	789	13 Juni 2003	E:\Al Qur'an\Al-quran digital\0-Al-Qur'an & Nabi\AL-QUR'AN DIGITAL\GIF\1\1_2.gif
9	1_3.gif	699	13 Juni 2003	E:\Al Qur'an\Al-quran digital\0-Al-Qur'an & Nabi\AL-QUR'AN DIGITAL\GIF\1\1_3.gif
10	1_4.gif	689	13 Juni 2003	E:\Al Qur'an\Al-quran digital\0-Al-Qur'an & Nabi\AL-QUR'AN DIGITAL\GIF\1\1_4.gif

Option Lebih Besar Sama Dengan

Untuk hasil metadata file yang dikorelasi yaitu file Welcome.docx yang metadata ukuran filenya berupa “10.313 byte”, yang dilakukan pencarian file-file yang berlokasi di Data E dengan pilihan “Lebih Besar Sama Dengan”, maka ditemukan bayak sekali file-file yang ukuran filenya lebih besar sama dengan 10.313 byte dari metadata ukuran file Welcome.docx yang ada di Data E tersebut, tetapi dari sekian bayak file yang sudah ditemukan, diambil sepuluh sampel yang dijadikan sebagai analisa pencarian file berdasarkan korelasi ukuran file. Berikut bisa di lihat hasil analisanya dari tabel 4.28 di bawah ini:

Tabel 4. 28 Hasil Korelasi File Ukuran Opsi Lebih Besar Sama Dengan

No.	File Name	Size	Date	Path
1	Toleransi Seorang Shalahuddin al.doc	33280	23 Oktober 2008	E:\Religius\Islam\DAQWAH\Toleransi Seorang Shalahuddin al.doc
2	TUJUH GOLONGAN YANG AKAN BERNAUNG DI BAWAH.doc	26112	21 November 2008	E:\Religius\Islam\DAQWAH\TUJUH GOLONGAN YANG AKAN BERNAUNG DI BAWAH.doc
3	TUJUH HAL YANG MERUPAKAN SEBAB AKIBAT.doc	30208	08 Januari 2011	E:\Religius\Islam\DAQWAH\TUJUH HAL YANG MERUPAKAN SEBAB AKIBAT.doc
4	Tujuh Macam Pahala.doc	64512	28 Mei 2009	E:\Religius\Islam\DAQWAH\Tujuh Macam Pahala.doc
5	TUJUH POIN SABDA NABI TENTANG DUNIA.doc	28160	08 Januari 2011	E:\Religius\Islam\DAQWAH\TUJUH POIN SABDA NABI TENTANG DUNIA.doc
6	Yaa Allah Berkahilah Kami di Bulan Rajab.doc	35328	19 April 2009	E:\Religius\Islam\DAQWAH\Yaa Allah Berkahilah Kami di Bulan Rajab.doc
7	Yang Manakah Anda.doc	27136	06 Mei 2009	E:\Religius\Islam\DAQWAH\Yang Manakah Anda.doc
8	yyidina muhammad wa aalihi washohbihi wassalim.docx	80715	15 Januari 2011	E:\Religius\Islam\DAQWAH\yyidina muhammad wa aalihi washohbihi wassalim.docx
9	bagaimana_seorang muslim_berpikir.rtf	272040	02 Maret 2004	E:\Religius\Islam\Harun Yahya E-book\bagaimana_seorang muslim_berpikir.rtf
10	Bagaimana_seorangmuslim_berpikir.pdf	2464332	02 Maret 2004	E:\Religius\Islam\Harun Yahya E-book\Bagaimana seorang muslim_berpikir.pdf

Option Antara

Untuk hasil metadata file yang dikorelasi yaitu file Welcome.docx yang metadata ukuran filenya berupa “10.313 byte”, yang dilakukan pencarian file-file yang berlokasi di Data E dengan pilihan “Antara”, dimana ukuran file yang di inputkan dari yang ukuran file yang bernilai batas minimal sampai dengan ukuran file yang bernilai batas maksimal, sehingga file-file yang akan di cari yaitu ukuran file yang ada diantara nilai batas minimal sampai dengan batas maksimal yang telah di inputkan. Dalam pengujian dan analisa ini, di inputkan ukuran file 10.000 byte untuk nilai batas

minimal dan 50.000 byte untuk nilai batas maksimal. Maka ditemukan bayak sekali file-file yang ukuran filenya diantara 10.000 byte sampai dengan 50.000 byte yang ada di Data E tersebut, tetapi dari sekian bayak file yang sudah ditemukan, diambil sepuluh sampel yang dijadikan sebagai analisa pencarian file berdasarkan korelasi ukuran file opsi antara. Berikut bisa di lihat hasil analisanya dari tabel 4.29 di bawah ini:

Tabel 4. 29 Hasil Korelasi File Ukuran Opsi Antara

No.	File Name	Size	Date	Path
1	Al Quran Digital.chw	20318	28 Februari 2012	E:\AGAMA\AL-QURAN dan al-HADITS\Al Quran Digital.chw
2	Share-e qurban.xls	13824	02 November 2012	E:\AGAMA\dari akh Riza\Share-e qurban.xls
3	Peserta HIT.xls	13824	02 November 2012	E:\AGAMA\dari akh Riza\Peserta HIT.xls
4	Orang Tayli.PDF	38074	11 Januari 2001	E:\AGAMA\Fiqih dan Mutiara Hikmah\Orang Tayli.PDF
5	Pic06868.jpg	39479	06 Mei 2002	E:\AGAMA\Islam & Science\Pic06868.jpg
6	Gender Equity In Islam.pdf	45213	27 Mei 2002	E:\AGAMA\Islam, Women & Family\Gender Equity In Islam.pdf
7	Ro'yu ttg muh bin ishaq.pdf	42339	11 Januari 2001	E:\AGAMA\Kajian Islami\Ro'yu ttg muh bin ishaq.pdf
8	Kami Berikan Cobaan Kepadamu.pdf	29045	11 Januari 2001	E:\AGAMA\Kajian Islami\Kami Berikan Cobaan Kepadamu.pdf
9	el-haji.gif	12639	21 November 2006	E:\AGAMA\Keluarga Islami\Keluarga Sakinah_files\el-haji.gif
10	template_css.css	10874	21 November 2006	E:\AGAMA\Keluarga Islami\Keluarga Sakinah_files\template_css.css

3. Korelasi Berdasarkan Ektensi (*File Type*)

Untuk hasil metadata file yang dikorelasi yaitu file Welcome.docx yang metadata ektensi filenya berupa “docx”, yang dilakukan pencarian file-file yang berlokasi di Data E dengan option File Type, maka ditemukan bayak sekali file-file yang ektensi filenya docx dari metadata file type Welcome.docx yang ada di Data E tersebut, tetapi dari sekian bayak file yang sudah ditemukan, diambil sepuluh sampel yang dijadikan sebagai analisa pencarian file berdasarkan korelasi file type. Berikut bisa di lihat hasil analisanya dari tabel 4.30 di bawah ini:

Tabel 4. 30 Hasil Korelasi File Berdasarkan Ektensi File

No.	File Name	Size	Date	Path
1	CARA MEMBUAT MAKALAH YANG BAIK.docx	40633	04 Juni 2016	E:\ASM Mataram 2016\CARA MEMBUAT MAKALAH YANG BAIK.docx
2	Fungsi Tombol Pada Keyboard.docx	44725	30 April 2016	E:\ASM Mataram 2016\Fungsi Tombol Pada Keyboard.docx
3	VLOOKUP dan HLOOKUP.docx	161683	07 Mei 2016	E:\ASM Mataram 2016\VLOOKUP dan HLOOKUP.docx
4	409875558039646.docx	8691	17 September 2016	E:\Backup Android\Telegram\Telegram Documents\409875558039646.docx
5	JADWAL PELAJARAN KELAS.docx	12089	25 Agustus 2014	E:\Bahan Pembinaan Gugus Sekotong\JADWAL PELAJARAN KELAS.docx
6	KALDIK TH. 2014-2015 LOBAR.docx	69002	13 Juni 2014	E:\Bahan Pembinaan Gugus Sekotong\KALDIK TH. 2014-2015 LOBAR.docx
7	Kalender Pendidikan.docx	12321	25 Agustus 2014	E:\Bahan Pembinaan Gugus Sekotong\Kalender Pendidikan.docx
8	PROGRAM SEMESTER GANJIL.docx	13852	25 Agustus 2014	E:\Bahan Pembinaan Gugus Sekotong\PROGRAM SEMESTER GANJIL.docx
9	PROGRAM SEMESTER GENAP.docx	13854	25 Agustus 2014	E:\Bahan Pembinaan Gugus Sekotong\PROGRAM SEMESTER GENAP.docx
10	PROGRAM TAHUNAN.docx	12209	25 Agustus 2014	E:\Bahan Pembinaan Gugus Sekotong\PROGRAM TAHUNAN.docx

4. Korelasi Berdasarkan Pemilik (*File Owner*)

Untuk hasil metadata file yang dikorelasi yaitu file Welcome.docx yang metadata owner atau pemilik filenya berupa “Amikom”, yang dilakukan pencarian file-file yang berlokasi di Data D dengan option Owner, maka ditemukan bayak sekali file-file yang owner atau pemilik filenya Amikom dari metadata owner file Welcome.docx yang ada di Data D tersebut, tetapi dari sekian bayak file yang sudah ditemukan, diambil sepuluh sampel yang dijadikan sebagai analisa pencarian file berdasarkan korelasi owner file. Berikut bisa di lihat hasil analisanya dari tabel 4.31 di bawah ini:

Tabel 4. 31 Hasil Korelasi File Berdasarkan Pemilik File

No.	File Name	Size	Date	Path
1	Tugas CHFI.docx	706260	27 Mei 2016	D:\S2 UII\Tugas CHFI.docx
2	indo_06_13.xls	47616	14 Januari 2016	D:\S2 UII\Semester I\SPK & Business Inteligence\tugas2-14917148\indo_06_13.xls
3	JUMLAH TENAGA MEDIS 2003-2012 (Data yang diolah).xlsx	994225	14 Januari 2016	D:\S2 UII\Semester I\Visualisasi Data BI\JUMLAH TENAGA MEDIS 2003-2012 (Data yang diolah).xlsx
4	Laporan Visualisasi Data.docx	2145635	14 Januari 2016	D:\S2 UII\Semester I\Visualisasi Data BI\Laporan Visualisasi Data.docx
5	tenaga-kesehatan-per-provinsi-2000-2012.csv	176519	12 Januari 2016	D:\S2 UII\Semester I\Visualisasi Data BI\tenaga-kesehatan-per-provinsi-2000-2012.csv
6	JUMLAH TENAGA MEDIS 2003-2012xlsx	1054086	14 Januari 2016	D:\S2 UII\Semester I\JUMLAH TENAGA MEDIS 2003-2012).xlsx
7	Laporan Visualisasi Data.docx	2868488	14 Januari 2016	D:\S2 UII\Semester I\Laporan Visualisasi Data.docx
8	Pelengkap Laporan.docx	13160	18 Januari 2016	D:\S2 UII\Semester III\Olah TKP\Ex UAS Laporan Olah TKP\Pelengkap Laporan.docx
9	Laporan Olah TKP.docx	4448169	31 Januari 2016	D:\S2 UII\Semester III\Olah TKP\Laporan FINAL O-TKP\Laporan Olah TKP.docx
10	Laporan Olah TKP.pdf	2102989	31 Januari 2016	D:\S2 UII\Semester III\Olah TKP\Laporan FINAL O-TKP\Laporan Olah TKP.pdf

5. Korelasi File dari Gabungan Beberapa Jenis Korelasi

Untuk hasil metadata file yang dikorelasi yaitu file Welcome.docx yang metadata ukuran filenya "10313 byte", pemilik filenya berupa "Amikom" dan ektensi file "docx" yang dilakukan pencarian file-file yang berlokasi di Documents Data C dengan option korelasi mulai dari Size dengan option Lebih Besar Sama Dengan, Owner dan File Type maka ditemukan bayak sekali file-file yang metadata size filenya "10313 byte" dengan option Lebih Besar Sama Dengan, pemilik filenya berupa "Amikom" dan ektensi file "docx" dari metadata file Welcome.docx yang ada di Documents Data C, tetapi dari sekian bayak file yang sudah ditemukan, diambil hanya sepuluh

sampel yang dijadikan sebagai analisa pencarian file berdasarkan gabungan dari beberapa jenis korelasi tersebut. Berikut bisa di lihat hasil analisanya dari tabel 4.32 di bawah ini:

Tabel 4. 32 Korelasi File dari Gabungan Beberapa Jenis Korelasi

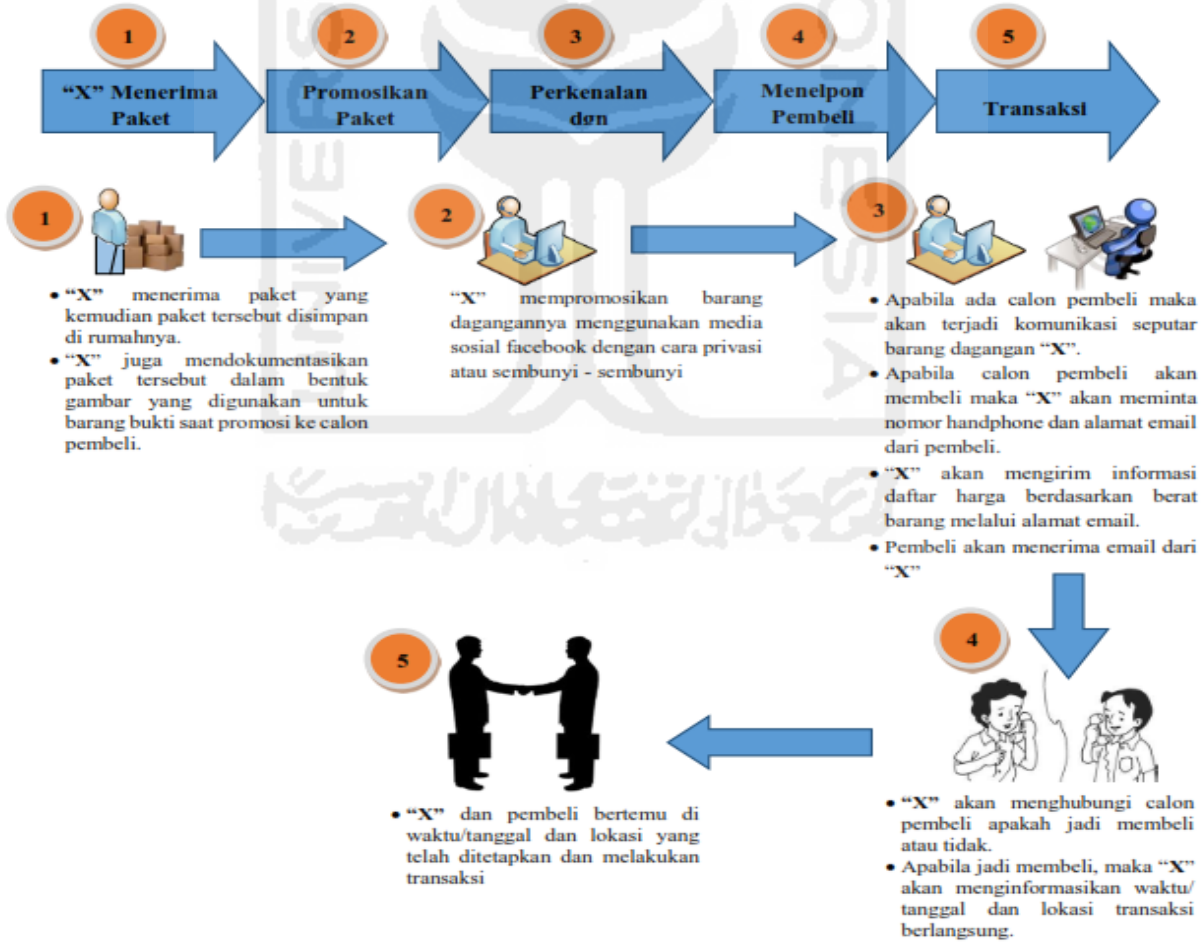
No.	File Name	Size	Date	Path
1	PELATIHAN CEH DAN CHFI.docx	17906	22 Januari 2016	C:\Users\Amikom\Documents\ PELATIHAN CEH DAN CHFI.docx
2	ACARA MAULID.docx	10517 077	02 Juli 2016	C:\Users\Amikom\Documents\ACARA MAULID.docx
3	ACARA NUZULUL.docx	25702 032	01 Juli 2016	C:\Users\Amikom\Documents\ACARA NUZULUL.docx
4	Algoritma SHA Dengan Java.docx	36233	03 September 2016	C:\Users\Amikom\Documents\Algoritma SHA Dengan Java.docx
5	Backpacker Jogja-Lombok.docx	17112	19 Desember 2015	C:\Users\Amikom\Documents\Backpacker Jogja-Lombok.docx
6	Belajar Basic OOP di Python.docx	20164	10 Desember 2015	C:\Users\Amikom\Documents\Belajar Basic OOP di Python.docx
7	Business Intelegence in Blue Bird Corporation.docx	16142	10 Desember 2015	C:\Users\Amikom\Documents\Business Intelegence in Blue Bird Corporation.docx
8	VM VirtualBox dan VirtualBox Extensions Pack.docx	80569 7	23 November 2015	C:\Users\Amikom\Documents\ VM VirtualBox dan VirtualBox Extensions Pack.docx
9	Cara Membagi Partisi Hardisk External Tanpa Software.docx	35837 5	15 Desember 2015	C:\Users\Amikom\Documents\Cara Membagi Partisi Hardisk External Tanpa Software.docx
10	Partisi Hardisk EASEUS Partition Master.docx	29317 7	15 Desember 2015	C:\Users\Amikom\Documents\Partisi Hardisk EASEUS Partition Master.docx

Semua hasil analisa yang sudah ditampilkan dalam tabel-tabel diatas, di dapatkan sebuah metadata file yang dibaca secara umum yang tidak terlau spesifikan dalam pembacaaan metadatanya, contohnya file JPG mempunyai nilai metadata merk kamera waktu pemoretannya dan file MP4 mempunyai nilai metadata *frame rate*, ini tentunya mempunyai nilai metadata yang berbeda dan lebih spesifik lagi, tetapi dalam analisa pembacaan metadata ini yang ditampilkan sama semua yaitu pembacaan metadata file secara umum dan ditemukan juga beberapa file dalam proses korelasi metadata file tersebut dari tampilan hasil korelasi yang dimunculkan itu adalah Nama File (*File Name*), Ukuran File (*Size*), Tanggal File (*Date*) dan Lokasi File (*Path*).

4.5 Studi Kasus dengan Melakukan Pendekatan Metadata

Pada tanggal 28 agustus 2015 “X” mendapat kiriman paket narkoba jenis sabu-sabu untuk di edarkan dan dijual ke wilayah Jawa Barat. Beberapa bulan kemudian transaksi yang dilakukan oleh “X” mulai menembus wilayah Jawa Tengah dan Jawa Timur, karena jarak antara wilayah yang sangat jauh sehingga “X” terpikir untuk melibatkan teknologi informasi dalam aksinya.

Pada bulan September 2015 “X” mulai melibatkan jaringan internet lewat media sosial Facebook untuk memperkenalkan dagangannya kepada calon pembeli secara privasi atau sembunyi-sembunyi. Berawal dari komunikasi media facebook transaksi dilakukan dan apabila calon pembeli tertarik untuk membeli maka “X” akan meminta nomor telepon/handphone dari calon pembeli untuk membahas lebih lanjut tentang harga jual, berat barang yang akan dibeli, dan lokasi transaksi serta untuk memudahkannya maka “X” akan meminta alamat email calon pembeli untuk mengirimkan data-data tersebut adapun dalam kiriman tersebut juga dilampirkan foto dari barang yang akan dibeli oleh calon. Berikut adalah gambar alur proses kasus transaksi narkoba:



Gambar 4. 24 Alur Proses Kasus Transaksi Narkoba

Barang Bukti

Pada kasus transaksi narkoba tersebut, dapat diketahui berbagai jenis barang bukti teknologi informasi yang digunakan, barang bukti tersebut dapat digolongkan menjadi 2 bagian yaitu :

1. Barang Bukti Elektronik

   Barang bukti elektronik yang digunakan adalah :

   a. Handphone berfungsi sebagai alat komunikasi.
   b. Laptop sebagai media untuk mengetik data, edit gambar dan melakukan komunikasi media sosial facebook
   c. Modem sebagai perangkat jaringan internet
   d. Sim Card yang digunakan untuk modem dan Handphone
   e. Thumbdrive Toshiba tempat penyimpanan bukti-bukti kasus transaksi narkoba
2. Barang Bukti Digital

   Barang bukti digital yang digunakan adalah :

   a. Facebook
   b. Email
   c. Dokumen
   d. Gambar

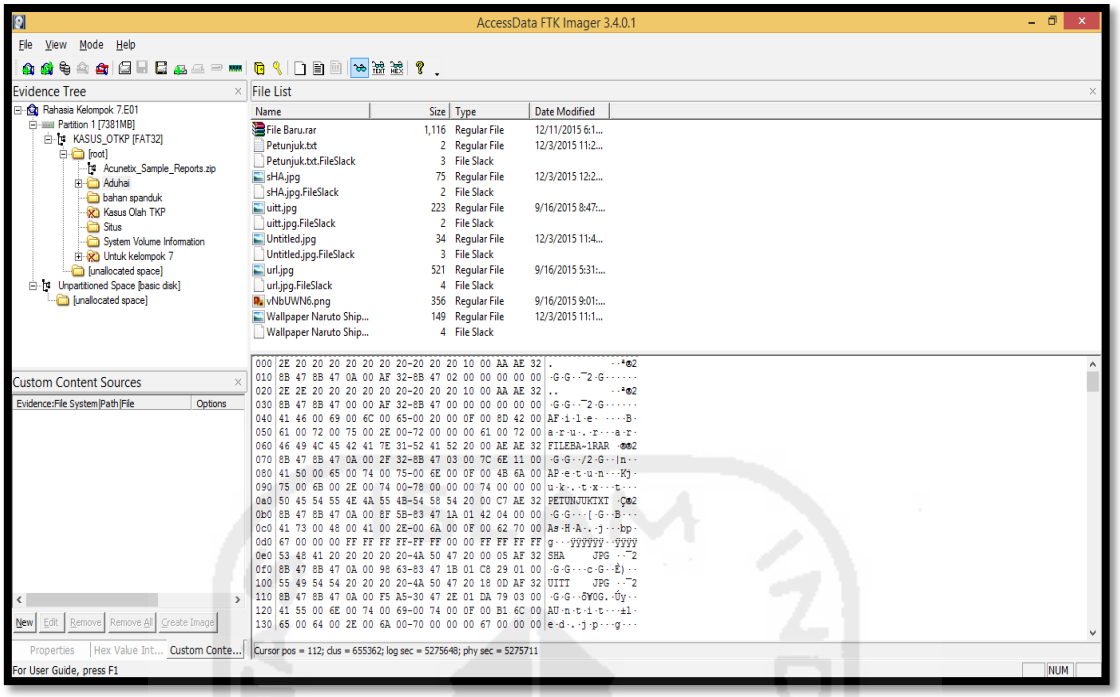
Pekerjan Investigasi

Menemukan keterlibatan “X” yang dapat memberatkannya di Pengadilan, barang bukti yang harus ditemukan adalah :

1. Temukan password di setiap barang bukti
2. File dokumen berupa daftar harga sabu-sabu
3. Gambar sabu-sabu sebagai contoh ilustrasi bukti sabu-sabu dari penjual kepada pembeli

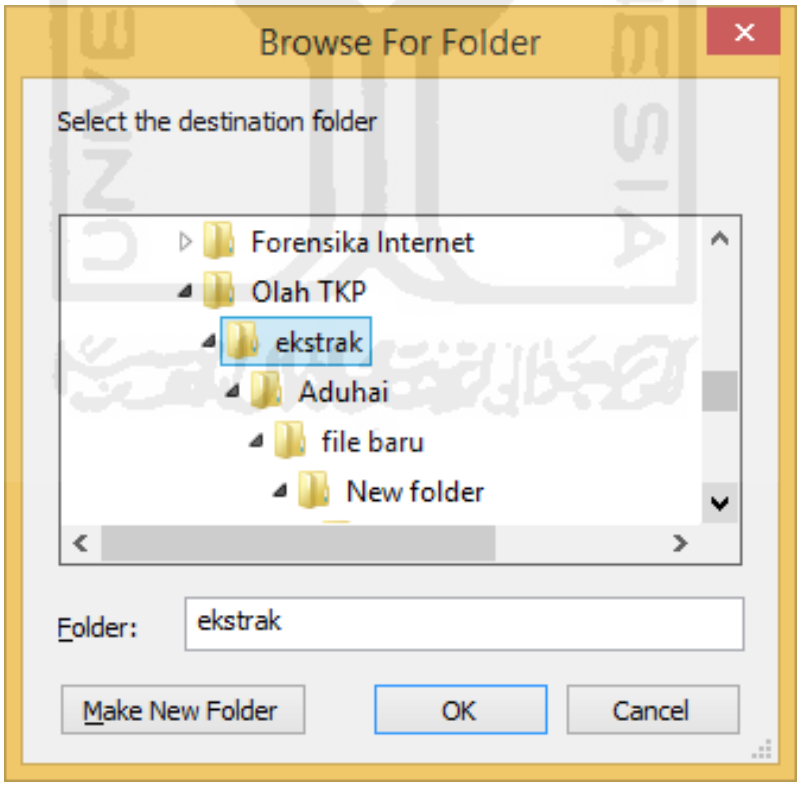
Analisa

1. Langkah pertama membuka bukti digital “Rahasia Kelompok 7.E01” dari barang bukti elektronik *Thumbdrive* Toshiba pada tools FTK Imager 3.4.0.1 seperti gambar 4.25 di bawah ini:



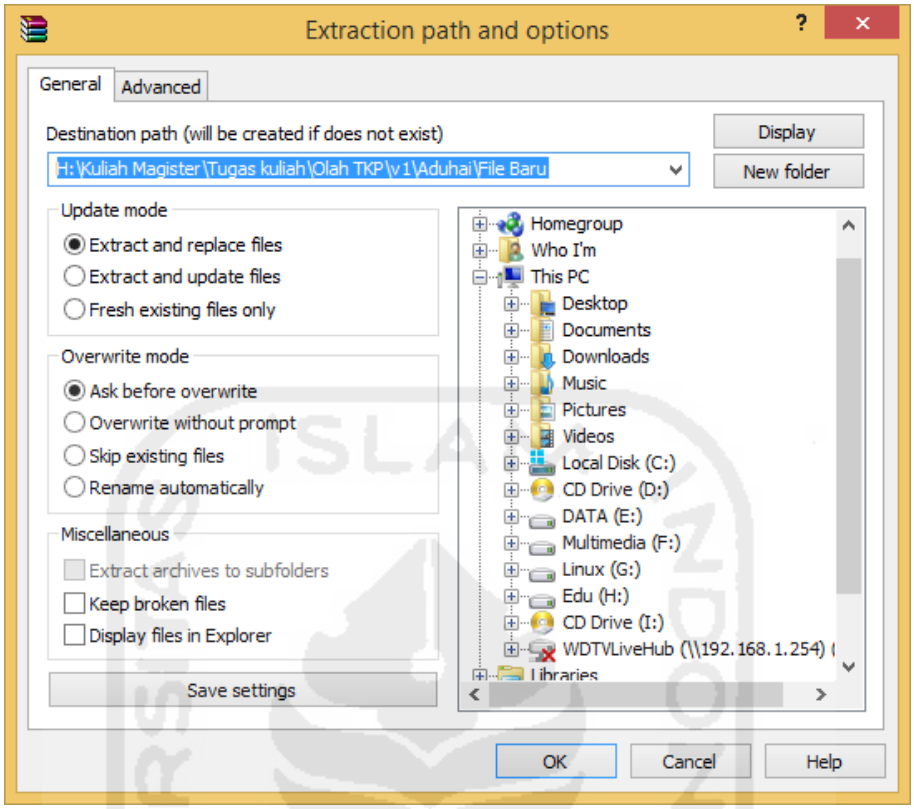
Gambar 4. 25 File Akuisi "Rahasia Kelompok 7.E01"

2. Langkah kedua melakukan *export file* bukti digital ke salah satu folder di Laptop/PC



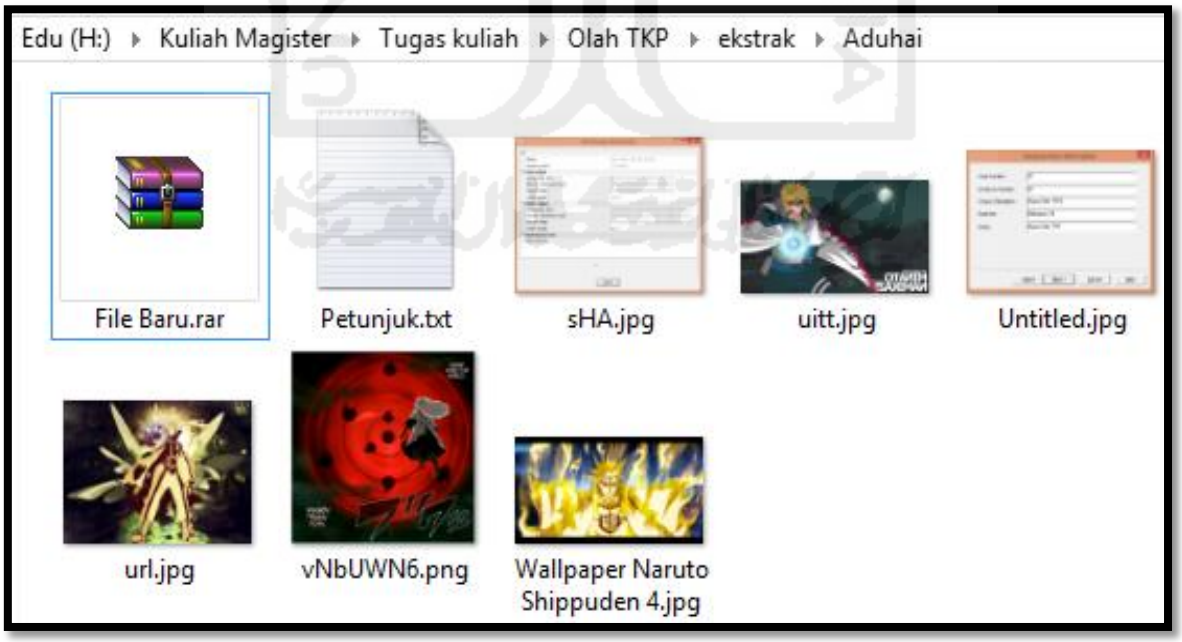
Gambar 4. 26 *Export File* Bukti Digital

3. Proses ekstraksi file Aduhai sperti pada gambar 4. 27



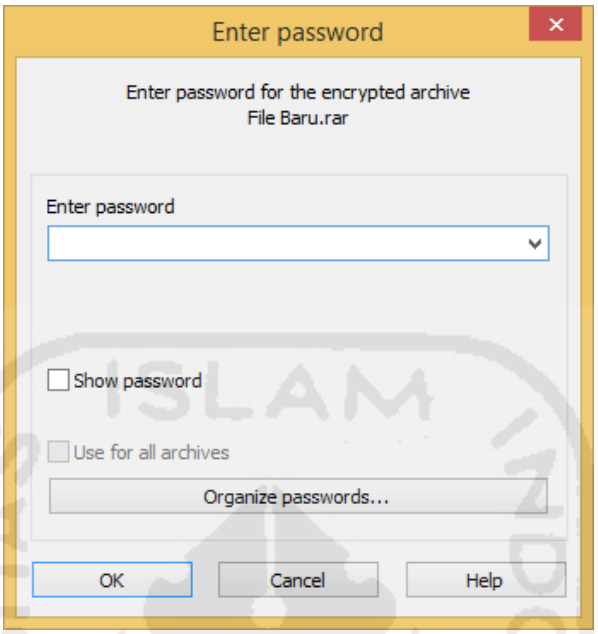
Gambar 4. 27 Proses Ekstraksi File Aduhai

4. Tampilan hasil ekstraksi folder Aduhai



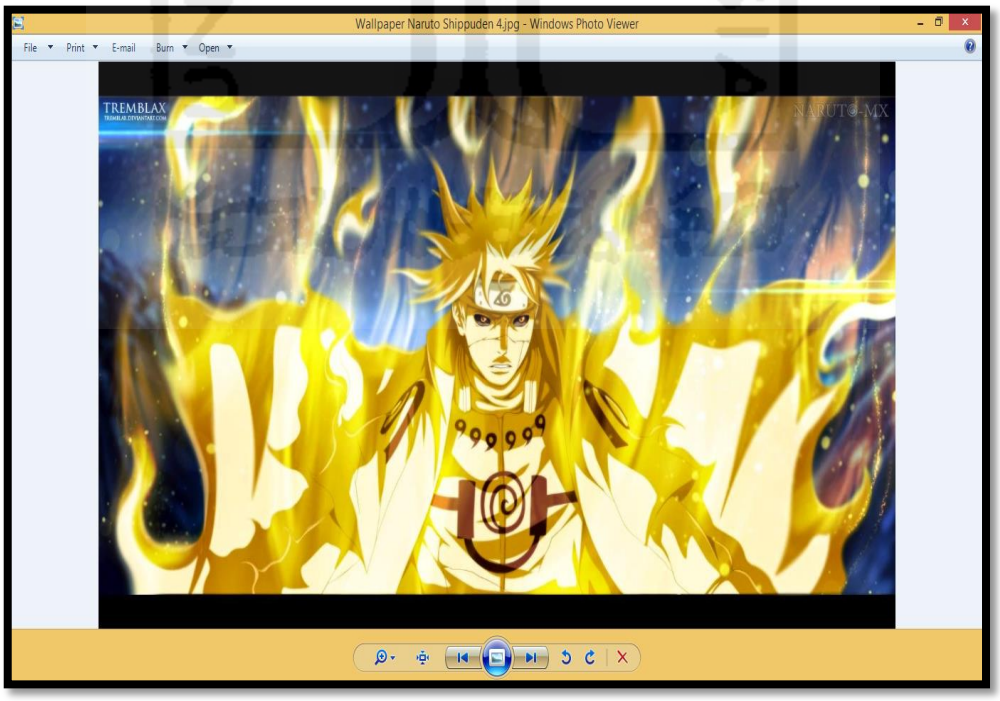
Gambar 4. 28 Hasil Ekstraksi Folder Aduhai

5. Langkah selanjutnya melakukan ekstraksi pada file “Baru.rar” yang ternyata menggunakan *password*



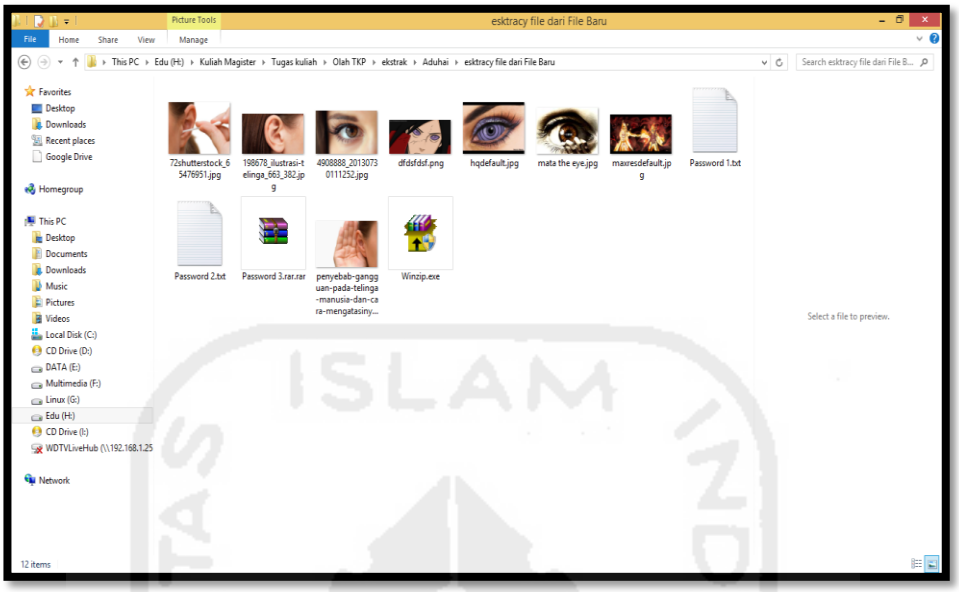
Gambar 4. 29 Enter Password File Baru.rar

6. Langkah berikutnya melakukan pencairan kata kunci atau password “File Baru.rar dengan mencoba kata atau angka yang ada di beberapa file gambar di folder aduhai dan akhirnya menemukan kata *password* Baru.rar “9” seperti terlihat pada gambar 4. 30 dibawah ini:



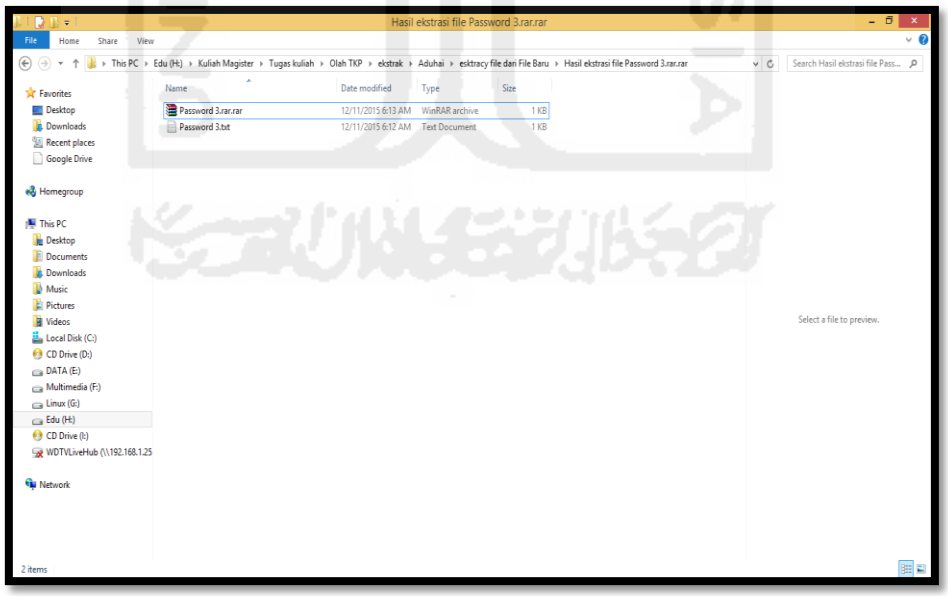
Gambar 4. 30 Kata Password Baru.rar “9”

7. Setelah ditemukan password “File Baru.rar” langkah selanjutnya melakukan ekstraksi dan hasil ektraksi pada file Baru yang terlihat pada gambar 4. 31 dibawah ini:



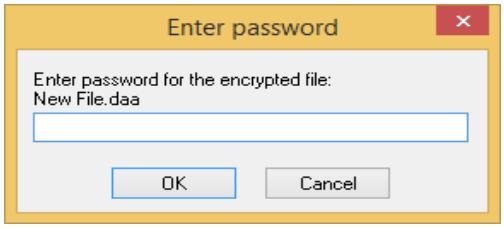
Gambar 4. 31 Hasil Ektraksi File Baru

8. Langkah selanjutnya melakukan pencairan bukti-bukti kasus yang di perintahkan pada saat penugasan investigasi dengan ekstraksi lagi file Baru.rar dengan nama “Password 3.rar.rar” dengan kata kunci dari file Baru.rar tersebut adalah Password 3.rar.rar



Gambar 4. 32 Ektraksi Lagi File Baru.rar

9. Ternyata setelah di eksteraksi file tersebut hanya sebaga file pengecoh examiner dari pelaku kejahatan melainkan bukan bukti yang di cari

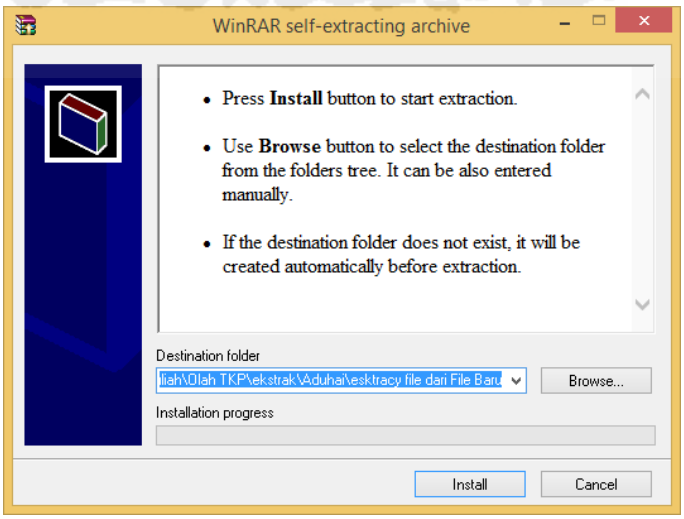


Gambar 4. 33 File Pengecoh Enter Password

10. Langkah selanjutnya mengecek sebuah file berextensi .exe dengan nama file Winzip.exe yang kemungkinan menyembunyikan bukti-bukti yang selama ini di cari

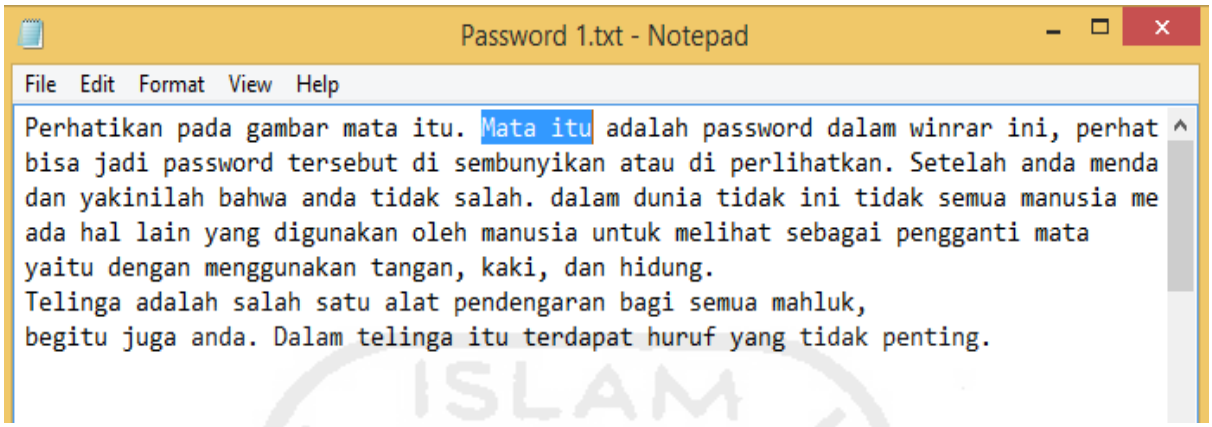
Gambar 4. 34 File Winzip.exe

11. Proses berikutnya instalasi file yang di duga menyimpan bukti-bukti dari kejahatan tersebut dengan file Winzip.exe



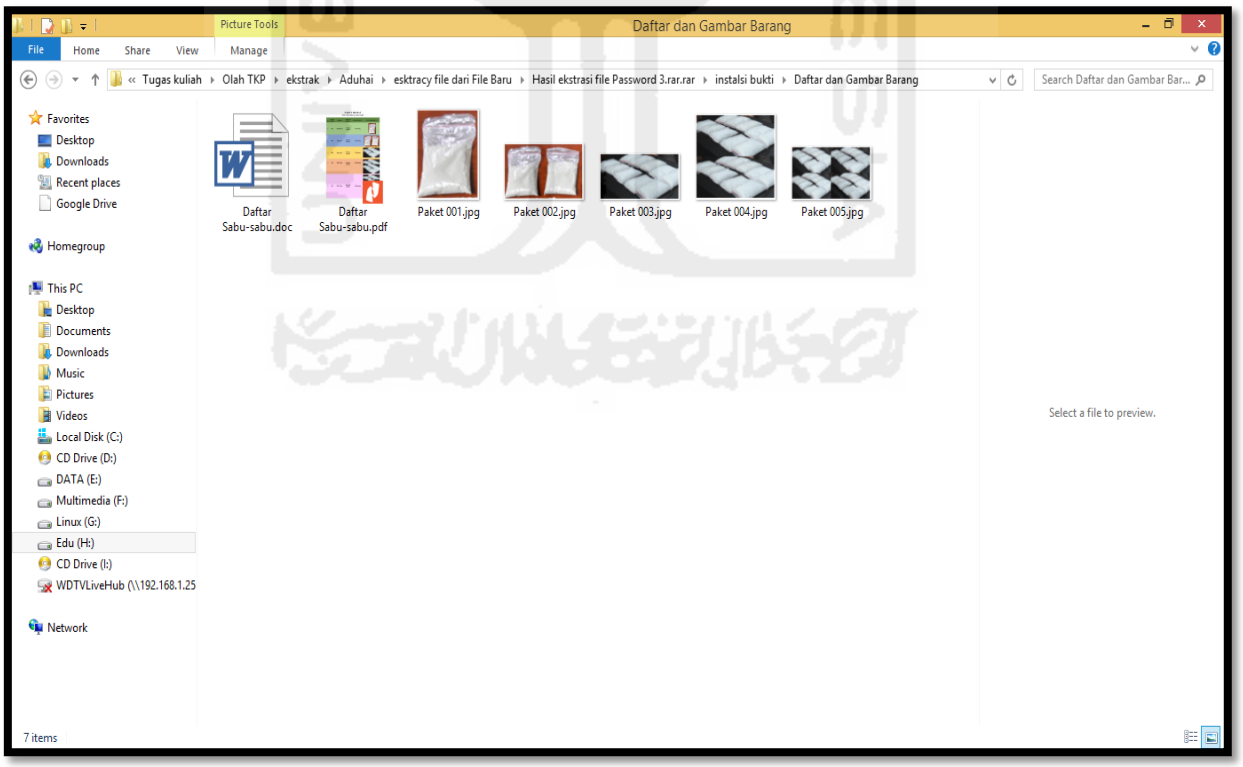
Gambar 4. 35 Instalasi File Winzip.exe

12. Dalam langkah instalasi file Winzip.exe harus memasukan kata kunci. Kata kunci atau password instalasi dari file Winzip.exe adalah “Mata itu”



Gambar 4. 36 Password Instalasi File Winzip.exe

13. Setelah proses installasi ekstraksi file Winzip.exe, maka ditemukan semua bukti-bukti digital yang diperintahkan pada penugasan investigasi yaitu file dokumen harga sabu dan gambar sabu-sabu, seperti pada gambar 4. 37 dibawah ini:



Gambar 4. 37 Daftar dan Gambar Barang

Kesimpulan

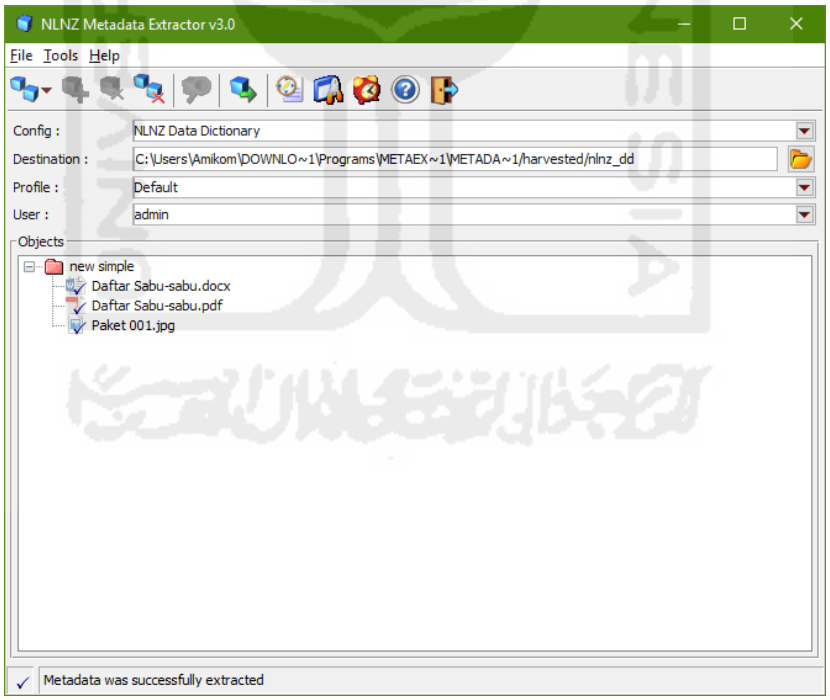
Setelah melakukan proses yang panjang, akhirnya diambil sebuah kesimpulan bahwa: **DI TEMUKAN** keterlibatan "X" dalam predaran dan penjualan narkoba ke beberapa tempat dan jumlah daftar harga sabu beserta foto sabu-sabu di dalam *Thumbdrive* Toshiba yang di akuisisi dan di analisa oleh Tim Investigator.

Melihat Metadata Temuan Bukti-bukti Analisa File "Rahasia Kelompok 7.E01"

Untuk penguatan barang bukti dari hasil analisa file "Rahasia Kelompok 7.E01" diatas, cara lain yang bisa dilakukan adalah pendekatan metadata terhadap temuan bukti-bukti dari analisa file "Rahasia Kelompok 7.E01" dengan tools standar metadata extractor dan metadata forensik yang telah dibangun dari hasil penelitian ini. Adapun temuan bukti-bukti tersebut yang akan di analisa metadatanya adalah sebagai berikut:

1. Melihat Metadata dengan tools Metadata Extractor

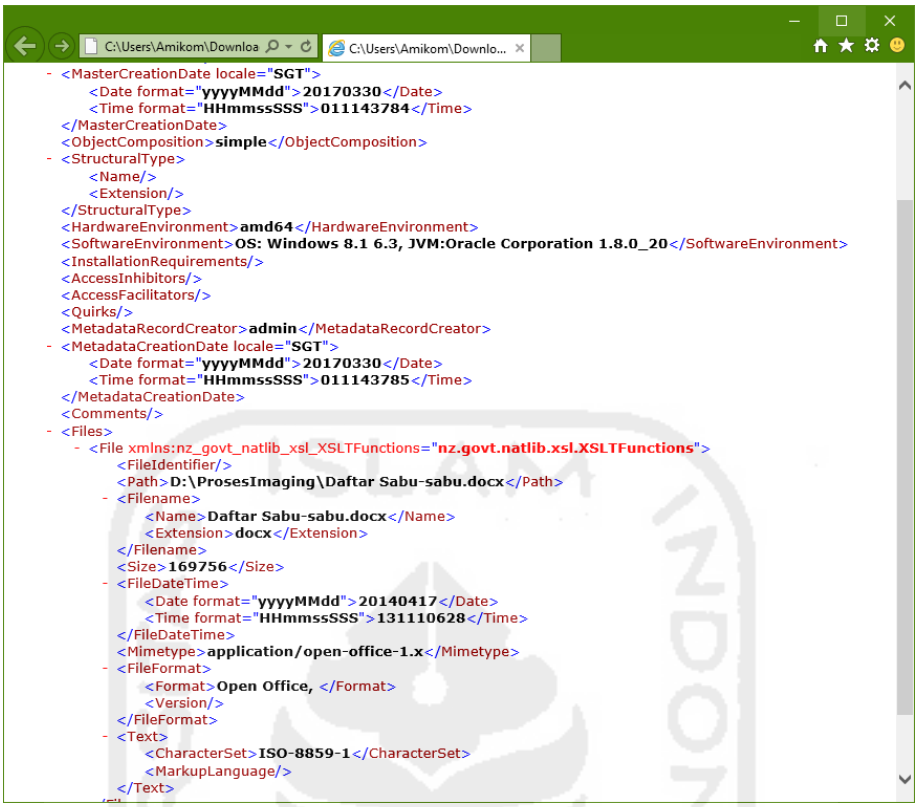
   Langkah pertama membuka Metadata Extractor – browse file yang akan dilihat metadatanya (bisa langsung banyak file), berikut metadata extractor ketika dijalankan:



Gambar 4. 38 Metadata Extractor Ketika Berjalan

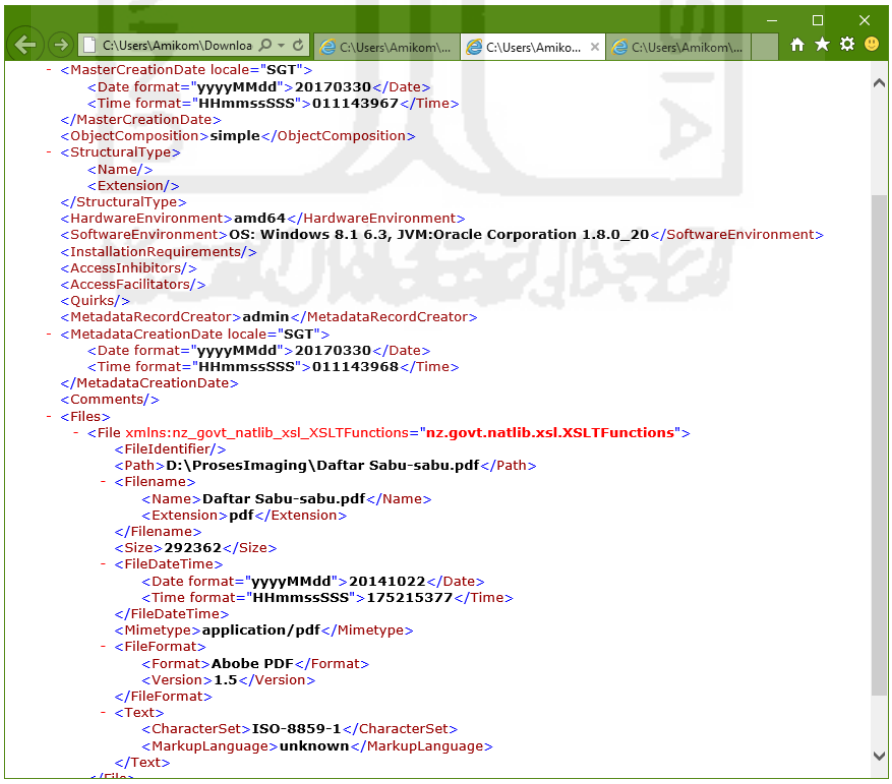
Hasil yang didapat dalam bentuk XML dan di buka dengan menggunakan Internet Explorer. Berikut gambar *screenshot* pada masing-masing temuan bukti file dibawah ini:

a. Temuan Bukti File Dokumen "**Daftar Sabu-sabu.docx**" berikut tampilan metadatanya:



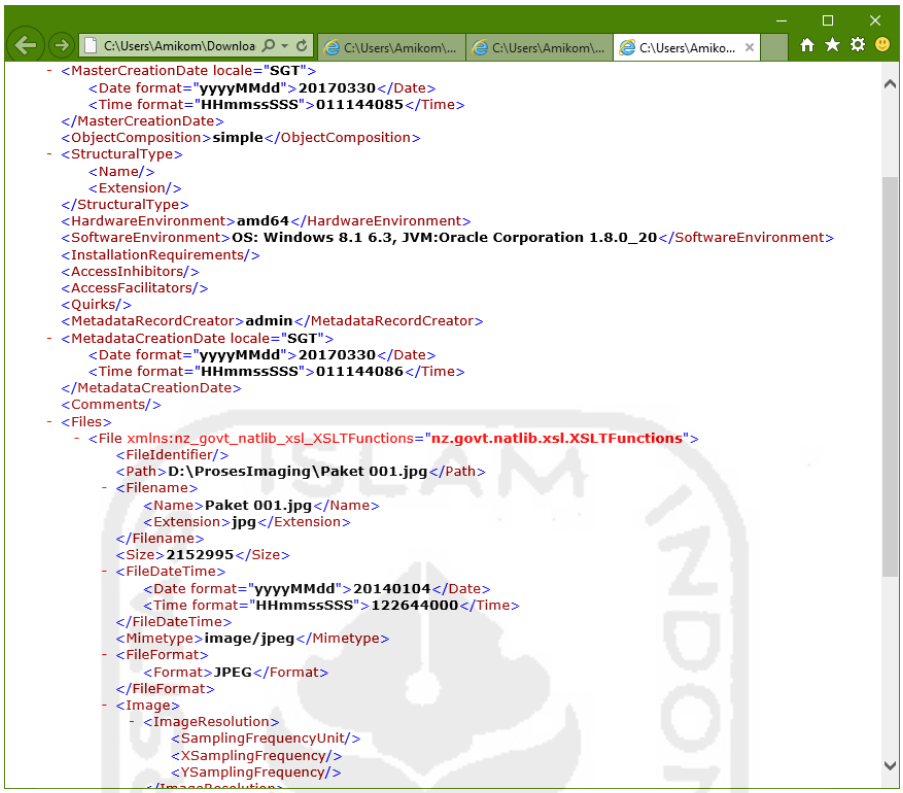
Gambar 4. 39 Hasil Metadata Extractor Daftar Sabu-sabu.docx

b. Temuan Bukti File Dokumen "**Daftar Sabu-sabu.pdf**" berikut tampilan metadatanya:



Gambar 4. 40 Hasil Metadata Extractor Daftar Sabu-sabu.pdf

c. Temuan Bukti File Gambar "**Paket 001.jpg**" berikut tampilan metadatanya:

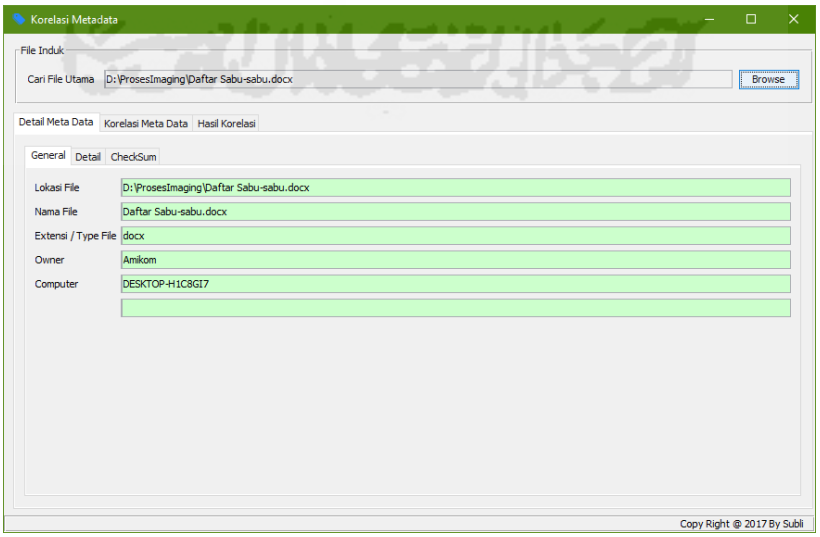


Gambar 4. 41 Hasil Metadata Extractor Paket 001.jpg

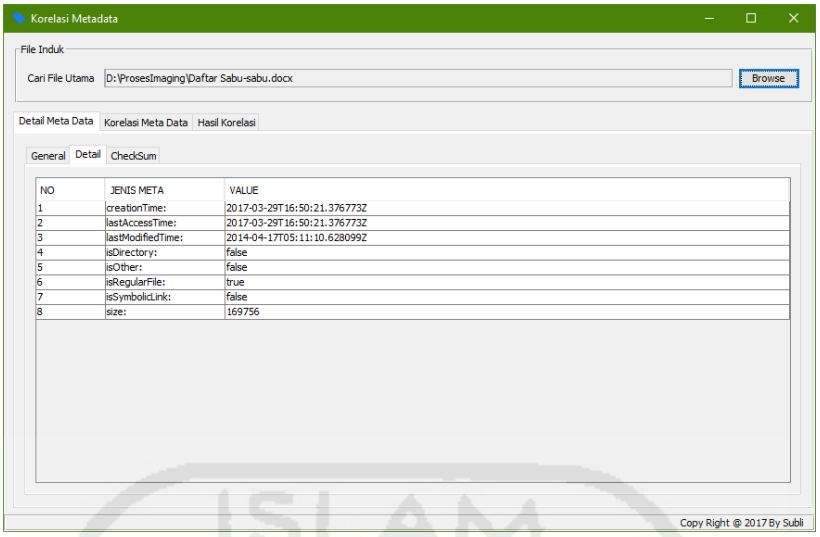
2. Melihat Metadata dengan tools Metadata Forensik

Langkah pertama yang dilakukan adalah membuka tools metadata forensik - *Browse* tempat penyimpanan file dokumen yang akan dilihat metadatanya - Proses beberapa saat nanti muncul Menu Detail Metadata - Hasil outputnya pada masing-masing file dibawah ini:

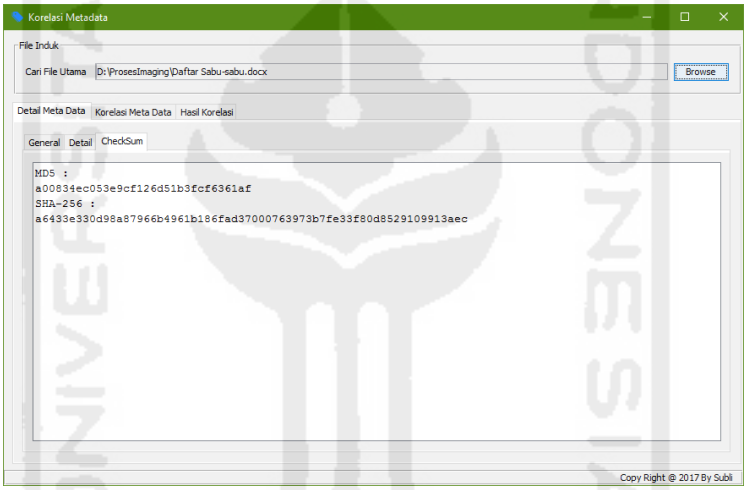
a. Temuan Bukti File Dokumen "**Daftar Sabu-sabu.doc**x" berikut tampilan metadatanya:



Gambar 4. 42 Hasil Metadata Forensik General Daftar Sabu-sabu.docx

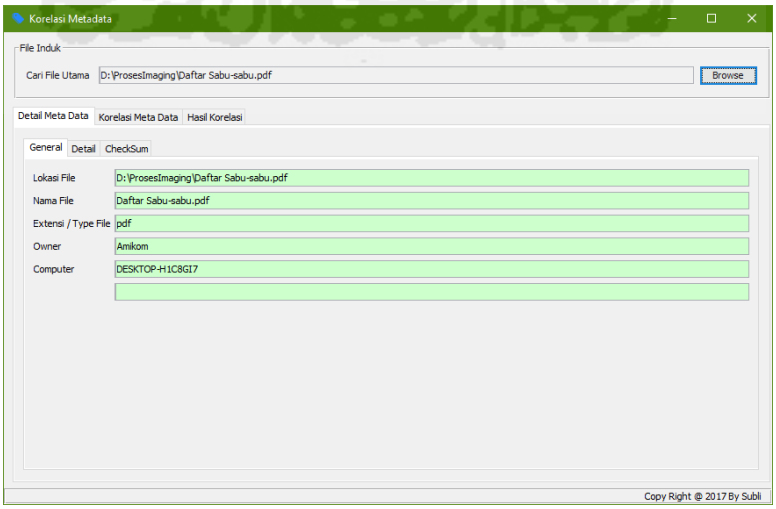


Gambar 4. 43 Hasil Metadata Forensik Detail Daftar Sabu-sabu.docx

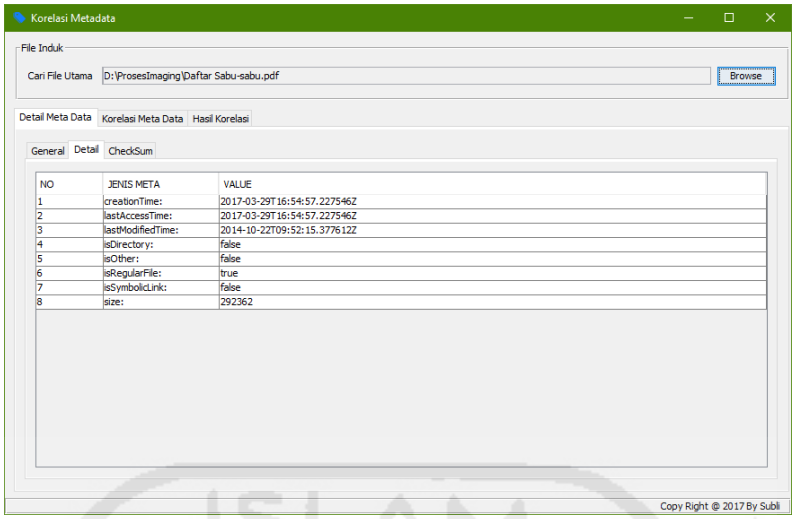


Gambar 4. 44 Hasil Metadata Forensik Checksum Daftar Sabu-sabu.docx

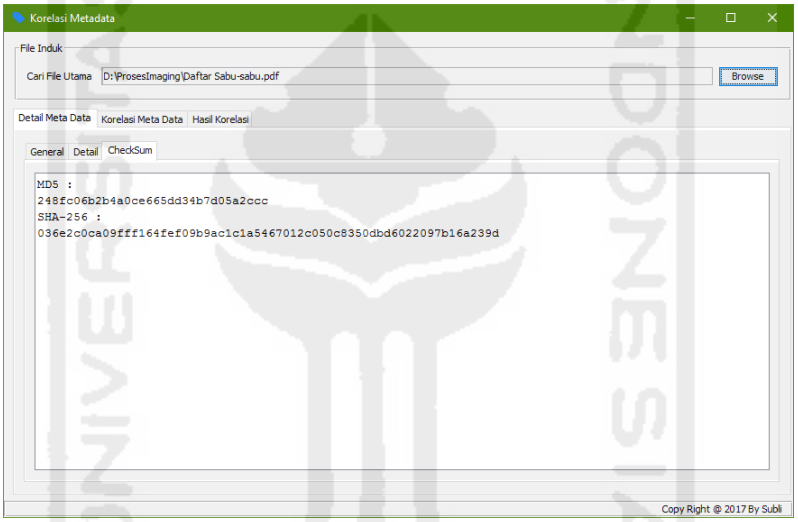
b. Temuan Bukti File Dokumen "**Daftar Sabu-sabu.pdf**" berikut tampilan metadatanya:



Gambar 4. 45 Hasil Metadata Forensik General Daftar Sabu-sabu.pdf

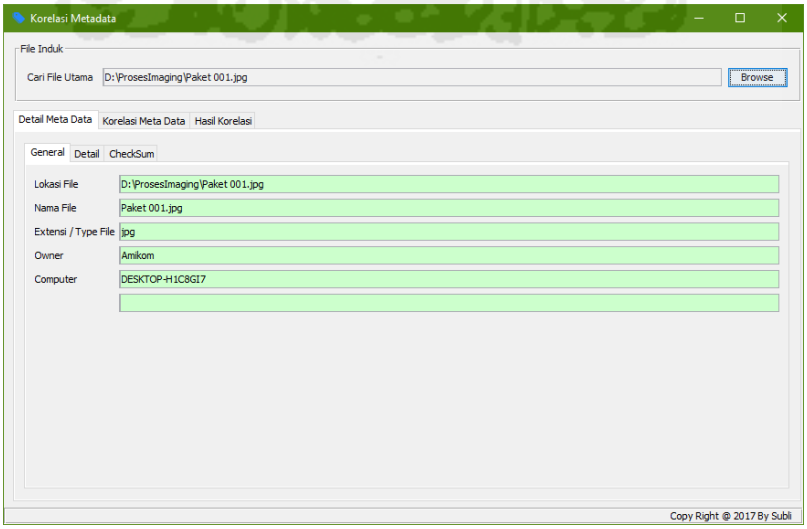


Gambar 4. 46 Hasil Metadata Forensik Detail Daftar Sabu-sabu.pdf

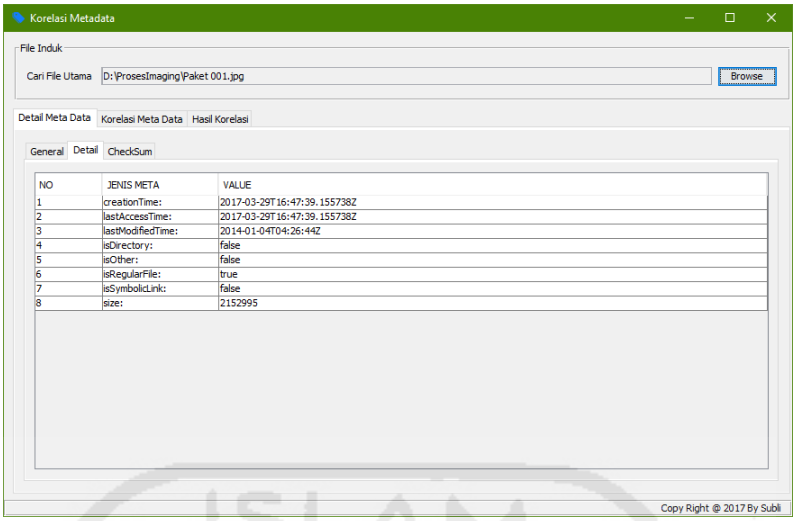


Gambar 4. 47 Hasil Metadata Forensik Checksum Daftar Sabu-sabu.pdf

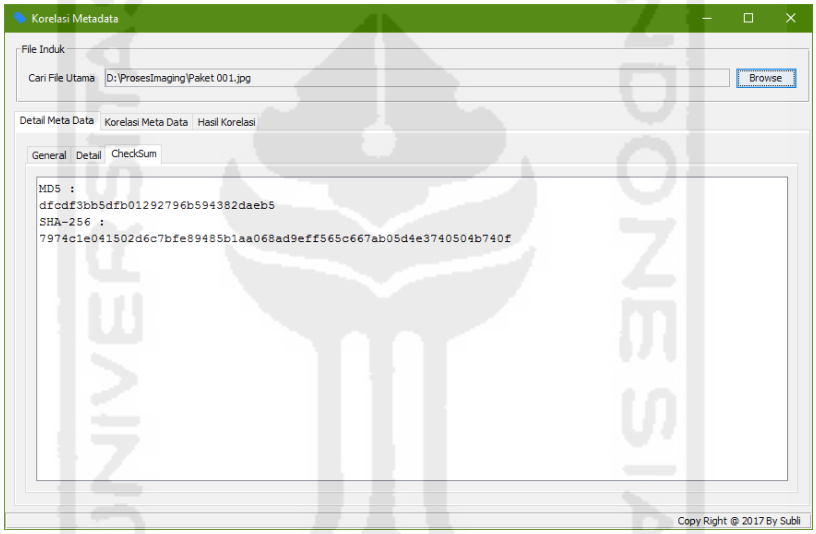
c. Temuan Bukti File Gambar "**Paket 001.jpg**" berikut tampilan metadatanya:



Gambar 4. 48 Hasil Metadata Forensik General Paket 001.jpg



Gambar 4. 49 Hasil Metadata Forensik Detail Paket 001.jpg



Gambar 4. 50 Hasil Metadata Forensik Checksum Paket 001.jpg

Perbandingan Penggunaan Kedua Tools Analisa Metadata File

Pada pembacaan metadata diatas oleh kedua tools Metadata eExtractor dan Metadata Forensik, dapat ditarik kesimpulan perbedaan pembacaan hasil metadatanya. Berikut dapat dilihat pada tabel 4. 33 perbandingan hasil pemeriksaan metadata file gambar Paket 001.jpg dari kedua tools tersebut:

Tabel 4. 33 Hasil Analisa Metadata Kedua Tools

Jenis Metadata	Metadata Extractor	Metadata Forensik
Folder Path	D:\ProsesImaging\Paket 001.jpg	D:\ProsesImaging\Paket 001.jpg
Name File	Paket 001.jpg	Paket 001.jpg
Type File	jpg	jpg

Lanjutan **Tabel 4. 33** Hasil Analisa Metadata Kedua Tools

Jenis Metadata	Metadata Extractor	Metadata Forensik
Owner	-	Amikom
Computer	-	DESKTOP-H1C8GI7
Creation Time	2017030, 011144086	2017-03-29T16:47:39.155738Z
Last Access Time	-	2017-03-29T16:47:39.155738Z
Last Modified Time	20140104, 122644000	2014-01-04T04:26:44Z
Is Directory	-	false
Is Other	-	false
Is Regular File	-	true
Is Symbolic Link	-	false
Size	2152995	2152995
Checksum MD5	-	dfcdf3bb5dfb01292796b594382daeb5
Checksum SHA-256	-	7974c1e041502d6c7bfe89485b1aa068ad9eff565c667ab05d4e3740504b740f
System Type	amd64	-
Jenis OS	Windows 10	-
JVM	Oracle Corporation 1.8.0_20	-

Pada pendekatan metadata dengan kedua tools tersebut, bisa dipastikan file dokumen tersebut adalah file Paket 001.jpg dengan ukuran file 2152995 byte. Terdapat kelebihan dari masing-masing kedua tools tersebut, misalnya metadata extractor dapat membaca metadata system type, jenis OS dan JVM, sedangkan metadata forensik dapat membaca metadata nilai checksum MD5 dan SHA-256.

Bab 5 Kesimpulan dan Saran

5.1 Kesimpulan

Berdasarkan hasil yang didapat dari proses implementasi dan pembahasan maka penelitian analisis metadata forensik untuk proses investigasi digital ini dapat ditarik beberapa kesimpulan yaitu:

1. Penggunaan metadata forensik yang sudah dibangun, semua jenis file yang ada di dalam komputer dapat dilihat detail metadatanya, termasuk tujuh macam file yang sudah dijadikan sampel yaitu DOCX, PDF, JPG, MP3, MP4, DD dan E01.
2. Karakteristik metadata file dapat dipahami secara umum, yaitu dibagi dalam tiga bagian; metadata secara general, metadata detail dan metadata nilai checksumnya. Metadata General terdiri dari lokasi file, nama file, type file, owner dan computer, Metadata Detail mulai dari *CreationTime*, *LastAccessTime*, *LastModifiedTime*, *isDirectory*, *isOther*, *isRegularFile*, *isSymbolicLink* dan *Size*, terakhir Metadata Checksum dengan nilai MD5 dan SHA-256.
3. Setelah melakukan korelasi metadata file, dapat ditemukannya file-file yang ada didalam komputer dari hasil pencarian korelasi berdasarkan parameter dari *Metadata File Date, Size, File Type* dan *Owner* yang ditampilkan dengan *Value File Name, Size, Date* dan *Path*.

5.2 Saran

Adapun saran-saran yang perlu diberikan dengan hasil penelitian ini adalah sebagai berikut:

1. Untuk segi hasil metadata file yang sudah ditemukan, perlu dikembangkan lagi menjadi lebih spesifik pembacaan metadatanya, karena setiap file memiliki masing-masing metadata yang berbeda.
2. Untuk korelasi metadata file dalam melakukan pencarian file-file tidak hanya korelasi berdasarkan parameter dari *Metadata File Date, Size, Type File* dan *Owner*.
3. Metode Graph - Bisa mencari file yang tidak ada dengan file yang ada.
4. Pengembangan dan penelitian lebih lanjut terkait algoritma metadata forensik ini yaitu bisa diketahuinya metadata file yang mana metadata yang belum dimodifikasi dan yang telah dimodifikasi.

Daftar Pustaka

Niso. (2004). Understanding Metadata: NISO Press. Retrieved from www.niso.org

Pendit, Putu Laxman. (2007). *Perpustakaan Digital: Perspektif Perpustakaan Perguruan Tinggi Indonesia*. Jakarta: Sagung Seto

Eko, Aryawan., & Smitdev, Community. (2010). *Metadata Undercover: Sehari Jadi Pakar Telematika*. Jakarta: PT Elex Media Komputindo

Media, Laksmana. (2009). *Membongkar Rahasia Keaslian Foto Berdasarkan Metadata*. Jakarta: PT. TransMedia

Salama et al. (2012). *Metadata Based Forensic Analysis of Digital Information in the Web*. ANNUAL SYMPOSIUM ON INFORMATION ASSURANCE & SECURE KNOWLEDGE MANAGEMENT, JUNE 5-6, 2012, ALBANY, NY

M.P. Roberts & J. Haggerty. (2013). *MetaFor: Metadata Signatures for Automated Remote File Identification in Forensic Investigations*. Proceedings of the European Information Security Multi-Conference (EISMC 2013)

Sriram Raghavan & S V Raghavan. (2013). *AssocGEN: Engine for Analyzing Metadata Based Associations in Digital Evidence*. IEEE Louisville Chapter. 978-1-4799-4061-5/13

Mark Phillips. (2013). *Metadata Analysis at the CommandLine*. Code{4}lib Journal. Issue 19, 2013-01-15

Woods et al. (2013). *Managing and Transforming Digital Forensics Metadata for Digital Collections*. University of North Carolina - School of Library and Information Science 216 Lenoir Drive, CB #3360, 100 Manning Hall Chapel Hill, NC 27599-3360 (919) 962-8366

Suratkar, Khanuja. (2014). *On The Role of Log Based Metadata in Forensic Analysis of Database Attacks*. International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622. International Conference on Industrial Automation and Computing (ICIAC- 12-13th April 2014)

Alanazi, Jones. (2015). *The Value of Metadata in Digital Forensics*. European Intelligence and Security Informatics Conference

Ezz El-Din Hemdan & Manjaiah D.H. (2015). *Forensic Analysis Approach Based on Metadata and Hash Values for Digital Objects in the Cloud*. International Journal of Innovative Research in Computer and Communication Engineering. Vol. 3, Special Issue 7, October 2015

Andy Spore. (2016). *Using Metadata in Litigation*. ProQuest

Marshall, A. M. (2008). Digital Forensics - Digital Evidence in Criminal Investigation. A John Wiley & Sons, Ltd.

Hewlett, Bill., & Packard, Dave. (2013). Spesification Notebook: HP. Retrieved from www.hp.com

Gates, Bill., & Allen, Paul. (2015). Excess: Windows 10. Retrieved from https://www.microsoft.com/en-us/software-download/windows10

Rita, Susilawati, S.S. (2006). MENGENAL METADATA SEBAGAI SEBUAH ALAT INVESTASI DATA. Rekomendasi tentang pemanfaatan dokumen ISO 19115 sebagai standar metadata nasional di Indonesia, Tim Kerja Standar Metadata, Pusat Sistem Jaringan dan Standardisasi data Spasial, Badan Koordinasi Survey dan Pemetaan Nasional, 2006.

Silviana, Arlis, Rita., & Mahendra, Saputra, Riyan. (2014). PENGEMBANGAN MODUL KONVERSI METADATA SPOT 5 VIRTUAL RECEPTION SESUAI FORMAT ISO 19115/19139: Seminar Nasional Penginderaan Jauh 2014

Universitas Ciputra. (2016). Pengertian Korelasi dan Macam-macam Korelasi. Retrieved from http://ciputrauceo.net/blog/2016/5/16/pengertian-korelasi-dan-macam-macam-korelasi

Satria, Wahono Romi. (2016) *Research Methodology*. Retrieved from http://romisatriawahono.net

Hasibuan, Zainal A. (2007). *Metodologi Penelitian Pada Bidang Ilmu Komputer dan Teknologi Informasi : Konsep, Teknik dan Aplikasi*. Fakultas Ilmu Komputer Universitas Indonesia

Lampiran

Source Code Pembuatan Aplikasi Sistem Metadata Forensik

```
import java.io.File;
import java.io.FileFilter;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.attribute.BasicFileAttributes;
import java.nio.file.attribute.FileOwnerAttributeView;
import java.nio.file.attribute.UserPrincipal;
import java.security.MessageDigest;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Map;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
/**
 * @author Amikom
 */
public class Home extends javax.swing.JFrame {
    /**
     * Creates new form Home
     */
    public Home() {
        initComponents();
        setIconImage(new javax.swing.ImageIcon(this.getClass().getResource("/tag.png")).getImage());
        tabelMeta.setModel(tModel);
        tabelHasil.setModel(tModelHasil);
        disablekorelasi(false);
        setLocationRelativeTo(null);
        jPanel12.setVisible(false);
    }
    private void initComponents() {
        jFileChooser1 = new javax.swing.JFileChooser();
        buttonGroup1 = new javax.swing.ButtonGroup();
        jLabel5 = new javax.swing.JLabel();
        jFileChooser2 = new javax.swing.JFileChooser();
        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        txtNamaFile = new javax.swing.JTextField();
        jButton1 = new javax.swing.JButton();
        jTabbedPane2 = new javax.swing.JTabbedPane();
        jPanel2 = new javax.swing.JPanel();
        jTabbedPane1 = new javax.swing.JTabbedPane();
        jPanel5 = new javax.swing.JPanel();
        jLabel3 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel7 = new javax.swing.JLabel();
        jLabel8 = new javax.swing.JLabel();
        txtLokasi = new javax.swing.JTextField();
        txtNamaFile2 = new javax.swing.JTextField();
        txtTypeFile = new javax.swing.JTextField();
        txtAuthors = new javax.swing.JTextField();
        txtComputer = new javax.swing.JTextField();
        txtOwner = new javax.swing.JTextField();
        jPanel6 = new javax.swing.JPanel();
        jScrollPane1 = new javax.swing.JScrollPane();
        tabelMeta = new javax.swing.JTable();
        jPanel7 = new javax.swing.JPanel();
        jScrollPane3 = new javax.swing.JScrollPane();
        txtSUM = new javax.swing.JTextArea();
        jPanel3 = new javax.swing.JPanel();
        jButton2 = new javax.swing.JButton();
        jLabel2 = new javax.swing.JLabel();
        txtPath = new javax.swing.JTextField();
```

```
jButton3 = new javax.swing.JButton();
jPanel8 = new javax.swing.JPanel();
jLabel10 = new javax.swing.JLabel();
txtKorelasi = new javax.swing.JTextField();
cmbKorelasi = new javax.swing.JComboBox();
jPanel11 = new javax.swing.JPanel();
rbantara = new javax.swing.JRadioButton();
jLabel11 = new javax.swing.JLabel();
jLabel12 = new javax.swing.JLabel();
txtAntaraAwal = new javax.swing.JFormattedTextField();
jLabel14 = new javax.swing.JLabel();
txtAntaraAkhir = new javax.swing.JFormattedTextField();
jLabel15 = new javax.swing.JLabel();
jPanel12 = new javax.swing.JPanel();
rbantara2 = new javax.swing.JRadioButton();
jLabel16 = new javax.swing.JLabel();
tglAntaraAwal = new com.toedter.calendar.JDateChooser();
jLabel17 = new javax.swing.JLabel();
tglAnataraAkhir = new com.toedter.calendar.JDateChooser();
jPanel13 = new javax.swing.JPanel();
rbSama = new javax.swing.JRadioButton();
rbSamaKecil = new javax.swing.JRadioButton();
rbSamaBesar = new javax.swing.JRadioButton();
rbBesar = new javax.swing.JRadioButton();
rbKecil = new javax.swing.JRadioButton();
jPanel9 = new javax.swing.JPanel();
chkAuthor = new javax.swing.JCheckBox();
chkFiletype = new javax.swing.JCheckBox();
chkOwner = new javax.swing.JCheckBox();
txtOwnerManual = new javax.swing.JTextField();
jPanel4 = new javax.swing.JPanel();
jScrollPane2 = new javax.swing.JScrollPane();
tabelHasil = new javax.swing.JTable();
jPanel10 = new javax.swing.JPanel();
jLabel13 = new javax.swing.JLabel();
jLabel5.setText("jLabel5");
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setTitle("Korelasi Metadata");
 jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder(javax.swing.BorderFactory.cr
 eateEtchedBorder(), "File Induk"));
jLabel1.setText("Cari File Utama");
txtNamaFile.setEditable(false);
jButton1.setText("Browse");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(
 jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel1Layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(jLabel1)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(txtNamaFile)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jButton1)
        .addContainerGap())
);
jPanel1Layout.setVerticalGroup(
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel1Layout.createSequentialGroup()
        .addContainerGap()
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel1)
            .addComponent(txtNamaFile,               javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(jButton1))
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
);
jLabel3.setText("Lokasi File ");
jLabel6.setText("Nama File");
jLabel4.setText("Extensi / Type File");
jLabel7.setText("Owner");
```

```
  jLabel8.setText("Computer");
  txtLokasi.setEditable(false);
  txtLokasi.setBackground(new java.awt.Color(204, 255, 204));
  txtNamaFile2.setEditable(false);
  txtNamaFile2.setBackground(new java.awt.Color(204, 255, 204));
  txtTypeFile.setEditable(false);
  txtTypeFile.setBackground(new java.awt.Color(204, 255, 204));
  txtAuthors.setEditable(false);
  txtAuthors.setBackground(new java.awt.Color(204, 255, 204));
  txtComputer.setEditable(false);
  txtComputer.setBackground(new java.awt.Color(204, 255, 204));
  txtOwner.setEditable(false);
  txtOwner.setBackground(new java.awt.Color(204, 255, 204));
  javax.swing.GroupLayout jPanel5Layout = new javax.swing.GroupLayout(jPanel5);
  jPanel5.setLayout(jPanel5Layout);
  jPanel5Layout.setHorizontalGroup(
 Panel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
      .addGroup(jPanel5Layout.createSequentialGroup()
          .addContainerGap()
   .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
              .addComponent(jLabel3)
              .addComponent(jLabel6)
              .addComponent(jLabel4)
              .addComponent(jLabel7)
              .addComponent(jLabel8))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
         .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
   )
              .addComponent(txtLokasi,       javax.swing.GroupLayout.DEFAULT_SIZE,       723,
   Short.MAX_VALUE)
              .addComponent(txtNamaFile2)
              .addComponent(txtTypeFile)
              .addComponent(txtAuthors)
              .addComponent(txtComputer)
              .addComponent(txtOwner))
         .addContainerGap())
  );
  jPanel5Layout.setVerticalGroup(
jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
      .addGroup(jPanel5Layout.createSequentialGroup()
          .addContainerGap()

   .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
              .addComponent(jLabel3)
              .addComponent(txtLokasi,                    javax.swing.GroupLayout.PREFERRED_SIZE,
   javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
         .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
   E)
              .addComponent(jLabel6)
              .addComponent(txtNamaFile2,                    javax.swing.GroupLayout.PREFERRED_SIZE,
   javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
         .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
   E)
              .addComponent(jLabel4)
              .addComponent(txtTypeFile,                javax.swing.GroupLayout.PREFERRED_SIZE,
   javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
        )
              .addComponent(jLabel7)
              .addComponent(txtAuthors,                javax.swing.GroupLayout.PREFERRED_SIZE,
   javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
         .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
   E)
              .addComponent(jLabel8)
              .addComponent(txtComputer,                javax.swing.GroupLayout.PREFERRED_SIZE,
   javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
          .addComponent(txtOwner,                    javax.swing.GroupLayout.PREFERRED_SIZE,
   javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
         .addContainerGap(191, Short.MAX_VALUE))
  );
  jTabbedPane1.addTab("General", jPanel5);
  tabelMeta.setModel(new javax.swing.table.DefaultTableModel(
```

```
        new Object [][] {
            {null, null, null},
            {null, null, null},
            {null, null, null},
            {null, null, null}
        },
        new String [] {
            "Title 1", "Title 2", "Title 3"
        }
    ));
    jScrollPane1.setViewportView(tabelMeta);
    javax.swing.GroupLayout jPanel6Layout = new javax.swing.GroupLayout(jPanel6);
    jPanel6.setLayout(jPanel6Layout);
    jPanel6Layout.setHorizontalGroup(
jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel6Layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jScrollPane1,        javax.swing.GroupLayout.DEFAULT_SIZE,        815,
     Short.MAX_VALUE)
            .addContainerGap())
    );
    jPanel6Layout.setVerticalGroup(
        jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel6Layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jScrollPane1,        javax.swing.GroupLayout.DEFAULT_SIZE,        330,
     Short.MAX_VALUE)
            .addContainerGap())
    );
    jTabbedPane1.addTab("Detail", jPanel6);
    txtSUM.setEditable(false);
    txtSUM.setColumns(20);
    txtSUM.setRows(5);
    txtSUM.setAutoscrolls(false);
    jScrollPane3.setViewportView(txtSUM);
    javax.swing.GroupLayout jPanel7Layout = new javax.swing.GroupLayout(jPanel7);
    jPanel7.setLayout(jPanel7Layout);
    jPanel7Layout.setHorizontalGroup(
     jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel7Layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jScrollPane3,        javax.swing.GroupLayout.DEFAULT_SIZE,        815,
     Short.MAX_VALUE)
            .addContainerGap())
    );
    jPanel7Layout.setVerticalGroup(
  jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel7Layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jScrollPane3,        javax.swing.GroupLayout.DEFAULT_SIZE,        330,
     Short.MAX_VALUE)
            .addContainerGap())
    );
    jTabbedPane1.addTab("CheckSum", jPanel7);
    javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
    jPanel2.setLayout(jPanel2Layout);
    jPanel2Layout.setHorizontalGroup(
     jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jTabbedPane1)
            .addContainerGap())
    );
    jPanel2Layout.setVerticalGroup(
     jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jTabbedPane1)
            .addContainerGap())
    );
    jTabbedPane2.addTab("Detail Meta Data", jPanel2);
    jButton2.setText("Korelasi File");
    jButton2.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton2ActionPerformed(evt);
        }
    });
```

```
jLabel2.setText("Tentukan Lokasi Korelasi");
txtPath.setEditable(false);
jButton3.setText("Browse");
jButton3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton3ActionPerformed(evt);
    }
});


 jPanel8.setBorder(javax.swing.BorderFactory.createTitledBorder(javax.swing.BorderFactory.cr
 eateEtchedBorder(),                                "Kriteria                                Korelasi",
 javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
 javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font("Arial", 1, 10)));
jLabel10.setText("Korelasi Berdasarkan");
txtKorelasi.setEditable(false);
txtKorelasi.setBackground(new java.awt.Color(204, 255, 204));
cmbKorelasi.setModel(new javax.swing.DefaultComboBoxModel(new String[] { " ", "Creation Time",
 "Last Modified Time", "Last Access Time", "Size" }));
cmbKorelasi.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        cmbKorelasiActionPerformed(evt);
    }
});
buttonGroup1.add(rbantara);
rbantara.setText("Antara");
rbantara.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        rbantaraActionPerformed(evt);
    }
});
jLabel11.setText("Batas Awal");
jLabel12.setText("Batas Akhir");
txtAntaraAwal.setFormatterFactory(new          javax.swing.text.DefaultFormatterFactory(new
 javax.swing.text.NumberFormatter(new java.text.DecimalFormat("#0"))));
jLabel14.setText("B");
txtAntaraAkhir.setFormatterFactory(new          javax.swing.text.DefaultFormatterFactory(new
 javax.swing.text.NumberFormatter(new java.text.DecimalFormat("#0"))));
jLabel15.setText("B");
javax.swing.GroupLayout jPanel11Layout = new javax.swing.GroupLayout(jPanel11);
jPanel11.setLayout(jPanel11Layout);
jPanel11Layout.setHorizontalGroup(
 jPanel11Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel11Layout.createSequentialGroup()
        .addContainerGap(
 .addGroup(jPanel11Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
 .addGroup(jPanel11Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
 false)
                .addComponent(rbantara,              javax.swing.GroupLayout.DEFAULT_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGroup(jPanel11Layout.createSequentialGroup()
 .addGroup(jPanel11Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel11)
                    .addComponent(jLabel12))
                .addGap(98, 98, 98)))
            .addGroup(jPanel11Layout.createSequentialGroup()
 .addGroup(jPanel11Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING,
 false)
                .addComponent(txtAntaraAkhir,  javax.swing.GroupLayout.Alignment.LEADING,
 javax.swing.GroupLayout.DEFAULT_SIZE, 123, Short.MAX_VALUE)
                .addComponent(txtAntaraAwal,  javax.swing.GroupLayout.Alignment.LEADING))
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
 .addGroup(jPanel11Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel14)
                .addComponent(jLabel15))))
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
);
jPanel11Layout.setVerticalGroup(
    jPanel11Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel11Layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(rbantara)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(jLabel11)
      .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
 .addGroup(jPanel11Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
```

```
            .addComponent(txtAntaraAwal,            javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(jLabel14))
        .addGap(3, 3, 3)
        .addComponent(jLabel12)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
 .addGroup(jPanel11Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(txtAntaraAkhir,           javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(jLabel15))
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
);
buttonGroup1.add(rbantara2);
rbantara2.setText("Antara");
jLabel16.setText("Dari Tanggal/jam");
tglAntaraAwal.setDate(new Date());
tglAntaraAwal.setDateFormatString("dd MM yyyy hh:mm:ss");
jLabel17.setText("Sampai Tangal/Jam");
tglAnataraAkhir.setDate(new Date());
tglAnataraAkhir.setDateFormatString("dd MM yyyy hh:mm:ss");
javax.swing.GroupLayout jPanel12Layout = new javax.swing.GroupLayout(jPanel12);
jPanel12.setLayout(jPanel12Layout);
jPanel12Layout.setHorizontalGroup(
 jPanel12Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel12Layout.createSequentialGroup()
        .addContainerGap()
 .addGroup(jPanel12Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(tglAntaraAwal,     javax.swing.GroupLayout.DEFAULT_SIZE,     159,
 Short.MAX_VALUE)
            .addGroup(jPanel12Layout.createSequentialGroup()
 .addGroup(jPanel12Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(rbantara2)
                    .addComponent(jLabel16)
                    .addComponent(jLabel17))
                .addGap(0, 0, Short.MAX_VALUE))
            .addComponent(tglAnataraAkhir,           javax.swing.GroupLayout.DEFAULT_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        .addContainerGap())
);
jPanel12Layout.setVerticalGroup(
 jPanel12Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel12Layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(rbantara2)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jLabel16)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(tglAntaraAwal,                javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE,              javax.swing.GroupLayout.PREFERRED_SIZE)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jLabel17)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(tglAnataraAkhir,              javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
);
buttonGroup1.add(rbSama);
rbSama.setSelected(true);
rbSama.setText("Sama Dengan");
buttonGroup1.add(rbSamaKecil);
rbSamaKecil.setText("Lebih Kecil Sama Dengan");
buttonGroup1.add(rbSamaBesar);
rbSamaBesar.setText("Lebih Besar Sama Dengan");
buttonGroup1.add(rbBesar);
rbBesar.setText("Lebih Besar");
buttonGroup1.add(rbKecil);
rbKecil.setText("Lebih Kecil");
javax.swing.GroupLayout jPanel13Layout = new javax.swing.GroupLayout(jPanel13);
jPanel13.setLayout(jPanel13Layout);
jPanel13Layout.setHorizontalGroup(
 jPanel13Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel13Layout.createSequentialGroup()
        .addContainerGap()
 .addGroup(jPanel13Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(rbSama)
            .addComponent(rbKecil)
            .addComponent(rbBesar)
```

```
            .addComponent(rbSamaBesar)
            .addComponent(rbSamaKecil))
        .addContainerGap(40, Short.MAX_VALUE))
);
jPanel13Layout.setVerticalGroup(
 jPanel13Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
 jPanel13Layout.createSequentialGroup()
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(rbSama)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(rbSamaKecil)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(rbSamaBesar)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(rbBesar)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(rbKecil)
        .addContainerGap())
);
javax.swing.GroupLayout jPanel8Layout = new javax.swing.GroupLayout(jPanel8);
jPanel8.setLayout(jPanel8Layout);
jPanel8Layout.setHorizontalGroup(
 jPanel8Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel8Layout.createSequentialGroup()
        .addGap(10,                                    10,                                    10)
 .addGroup(jPanel8Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
            .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
 jPanel8Layout.createSequentialGroup()
                .addComponent(jPanel13,              javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE,               javax.swing.GroupLayout.PREFERRED_SIZE)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jPanel11,              javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE,               javax.swing.GroupLayout.PREFERRED_SIZE)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jPanel12,              javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
 jPanel8Layout.createSequentialGroup()
                .addComponent(jLabel10)
                .addGap(18, 18, 18)
                .addComponent(cmbKorelasi,    javax.swing.GroupLayout.PREFERRED_SIZE,    148,
 javax.swing.GroupLayout.PREFERRED_SIZE)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(txtKorelasi,    javax.swing.GroupLayout.PREFERRED_SIZE,    244,
 javax.swing.GroupLayout.PREFERRED_SIZE)))
        .addContainerGap(260, Short.MAX_VALUE))
);
jPanel8Layout.setVerticalGroup(
 jPanel8Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel8Layout.createSequentialGroup()
        .addContainerGap()
 .addGroup(jPanel8Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel10)
            .addComponent(txtKorelasi,               javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(cmbKorelasi,               javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE,              javax.swing.GroupLayout.PREFERRED_SIZE))
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
 .addGroup(jPanel8Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jPanel13,                    javax.swing.GroupLayout.DEFAULT_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(jPanel11,                    javax.swing.GroupLayout.DEFAULT_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(jPanel12,                    javax.swing.GroupLayout.DEFAULT_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        .addContainerGap())
);
 jPanel9.setBorder(javax.swing.BorderFactory.createTitledBorder(javax.swing.BorderFactory.cr
 eateEtchedBorder(),                          "Kriteria                              Tambahan",
 javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
 javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font("Arial", 1, 10)));
chkAuthor.setText("Owner");
chkAuthor.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        chkAuthorActionPerformed(evt);
    }
```

```
});
chkFiletype.setText("File Type");
chkOwner.setText("Cari Nama Pemilik File Disini:");
chkOwner.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        chkOwnerMouseClicked(evt);
    }
    public void mouseReleased(java.awt.event.MouseEvent evt) {
        chkOwnerMouseReleased(evt);
    }
});
javax.swing.GroupLayout jPanel9Layout = new javax.swing.GroupLayout(jPanel9);
jPanel9.setLayout(jPanel9Layout);
jPanel9Layout.setHorizontalGroup(
 jPanel9Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel9Layout.createSequentialGroup()
        .addContainerGap()
 .addGroup(jPanel9Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel9Layout.createSequentialGroup()
                .addComponent(chkAuthor)
                .addGap(40, 40, 40)
                .addComponent(chkOwner)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(txtOwnerManual,  javax.swing.GroupLayout.PREFERRED_SIZE,  132,
 javax.swing.GroupLayout.PREFERRED_SIZE))
            .addComponent(chkFiletype))
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
);
jPanel9Layout.setVerticalGroup(
 jPanel9Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel9Layout.createSequentialGroup()
        .addContainerGap()
 .addGroup(jPanel9Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(chkAuthor)
            .addComponent(chkOwner)
            .addComponent(txtOwnerManual,           javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE,              javax.swing.GroupLayout.PREFERRED_SIZE))
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(chkFiletype)
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
);
javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
jPanel3.setLayout(jPanel3Layout);
jPanel3Layout.setHorizontalGroup(
 jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel3Layout.createSequentialGroup()
        .addContainerGap()
 .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jPanel8,                  javax.swing.GroupLayout.DEFAULT_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addComponent(txtPath)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jButton3))
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addComponent(jLabel2)
                .addGap(0, 0, Short.MAX_VALUE))
            .addComponent(jPanel9,                  javax.swing.GroupLayout.DEFAULT_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
 jPanel3Layout.createSequentialGroup()
                .addGap(0, 0, Short.MAX_VALUE)
                .addComponent(jButton2)))
        .addContainerGap())
);
jPanel3Layout.setVerticalGroup(
 jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
 jPanel3Layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(jLabel2)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
 .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(txtPath,                  javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(jButton3))
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
        .addComponent(jPanel8,                          javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE,                  javax.swing.GroupLayout.PREFERRED_SIZE)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jPanel9,                          javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE,                  javax.swing.GroupLayout.PREFERRED_SIZE)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jButton2)
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
);
jTabbedPane2.addTab("Korelasi Meta Data", jPanel3);
tabelHasil.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {
        {null, null, null, null},
        {null, null, null, null},
        {null, null, null, null},
        {null, null, null, null}
    },
    new String [] {
        "Title 1", "Title 2", "Title 3", "Title 4"
    }
));
jScrollPane2.setViewportView(tabelHasil);
javax.swing.GroupLayout jPanel4Layout = new javax.swing.GroupLayout(jPanel4);
jPanel4.setLayout(jPanel4Layout);
jPanel4Layout.setHorizontalGroup(
 jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel4Layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(jScrollPane2,        javax.swing.GroupLayout.DEFAULT_SIZE,        840,
 Short.MAX_VALUE)
        .addContainerGap())
);
jPanel4Layout.setVerticalGroup(
 jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel4Layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(jScrollPane2,        javax.swing.GroupLayout.DEFAULT_SIZE,        380,
 Short.MAX_VALUE)
        .addContainerGap())
);
jTabbedPane2.addTab("Hasil Korelasi", jPanel4);
jPanel10.setBorder(javax.swing.BorderFactory.createEtchedBorder());
jLabel13.setText("Copy Right @ 2017 By Subli");
javax.swing.GroupLayout jPanel10Layout = new javax.swing.GroupLayout(jPanel10);
jPanel10.setLayout(jPanel10Layout);
jPanel10Layout.setHorizontalGroup(
    jPanel10Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
 jPanel10Layout.createSequentialGroup()
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(jLabel13)
        .addContainerGap())
);
jPanel10Layout.setVerticalGroup(
 jPanel10Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addComponent(jLabel13)
);
javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
        .addContainerGap()
 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
            .addComponent(jPanel1,                      javax.swing.GroupLayout.DEFAULT_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(jTabbedPane2)))
    .addComponent(jPanel10,                             javax.swing.GroupLayout.DEFAULT_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(jPanel1,                      javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
      .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
            .addComponent(jTabbedPane2)
          .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jPanel10,                    javax.swing.GroupLayout.PREFERRED_SIZE,
     javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
    );
    pack();
}
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    kosong();
    if (jFileChooser1 == null) {
        jFileChooser1 = new JFileChooser();
        jFileChooser1.addChoosableFileFilter(new MyCostumFilter());
        jFileChooser1.setAcceptAllFileFilterUsed(false);
    }
    int a = jFileChooser1.showOpenDialog(this);
    if (a == JFileChooser.APPROVE_OPTION) {
        File file = jFileChooser1.getSelectedFile();
        txtNamaFile.setText(file.getAbsolutePath());
        getMetaData(file, file.getName());
    }
}
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    if (txtPath.getText().isEmpty()) {
        JOptionPane.showMessageDialog(rootPane, "Pilih Lokasi Korelasi terlebih dahulu");
    } else {
        if (rbSama.isSelected()) {
            perbandingan = 1;
        } else if (rbBesar.isSelected()) {
            perbandingan = 3;
        } else if (rbKecil.isSelected()) {
            perbandingan = 2;
        } else if (rbSamaBesar.isSelected()) {
            perbandingan = 5;
        } else if (rbantara.isSelected()) {
            perbandingan = 6;
            try {
                sizeawal = Integer.parseInt(txtAntaraAwal.getText());
                sizeakhir = Integer.parseInt(txtAntaraAkhir.getText());
            } catch (Exception ex) {
                JOptionPane.showMessageDialog(null, "Input Size Awal atau Size akhir salah");
            }
        } else if (rbantara2.isSelected()) {
            perbandingan = 7;
        } else {
            perbandingan = 4;
        }
        if(rbantara.isSelected()        &&        (txtAntaraAwal.getText().isEmpty()        ||
    txtAntaraAkhir.getText().isEmpty())){
            JOptionPane.showMessageDialog(null,  "Isi  Size  awal  dan  atau  size  akhir  terlebih
    dahulu");
        }else
        if (chkOwner.isSelected() && txtOwnerManual.getText().isEmpty()) {
            JOptionPane.showMessageDialog(null, "Inputkan Nama Terlebih Dahulu");
        } else if (cmbKorelasi.getSelectedIndex() == 2) {
            jTabbedPane2.setSelectedIndex(2);
            if (chkAuthor.isSelected()) {
                if (chkFiletype.isSelected()) {
                    if (chkOwner.isSelected()) {
                        getHasil(txtPath.getText(), tglauthorextensiownerFilter, true);
                        walk(txtPath.getText(), tglauthorextensiownerFilter);
                    } else {
                        getHasil(txtPath.getText(), tglauthorextensiFilter, true);
                        walk(txtPath.getText(), tglauthorextensiFilter);
                    }
                } else {
                    if (chkOwner.isSelected()) {
                        getHasil(txtPath.getText(), tglauthorownerFilter, true);
                        walk(txtPath.getText(), tglauthorownerFilter);
                    } else {
                        getHasil(txtPath.getText(), tglauthorFilter, true);
                        walk(txtPath.getText(), tglauthorFilter);
                    }
                }
            } else {
                if (chkFiletype.isSelected()) {
                    if (chkOwner.isSelected()) {
                        getHasil(txtPath.getText(), tglextensiownerFilter, true);
```

```
                walk(txtPath.getText(), tglextensiownerFilter);
            } else {
                getHasil(txtPath.getText(), tglextensiFilter, true);
                walk(txtPath.getText(), tglextensiFilter);
            }
        } else {
            if (chkOwner.isSelected()) {
                getHasil(txtPath.getText(), tglownerFilter, true);
                walk(txtPath.getText(), tglownerFilter);
            } else {
                getHasil(txtPath.getText(), tglFilter, true);
                walk(txtPath.getText(), tglFilter);
            }
        }
    }
    if(tabelHasil.getRowCount()<=0){
        JOptionPane.showMessageDialog(null, "Tidak ada file yang ditemukan");
    }
} else if (cmbKorelasi.getSelectedIndex() == 4) {
    jTabbedPane2.setSelectedIndex(2);
    if (chkAuthor.isSelected()) {
        if (chkFiletype.isSelected()) {
            if (chkOwner.isSelected()) {
                getHasil(txtPath.getText(), sizeauthorextensiownerFilter, true);
                walk(txtPath.getText(), sizeauthorextensiownerFilter);
            } else {
                getHasil(txtPath.getText(), sizeauthorextensiFilter, true);
                walk(txtPath.getText(), sizeauthorextensiFilter);
            }
        } else {
            if (chkOwner.isSelected()) {
                getHasil(txtPath.getText(), sizeauthorownerFilter, true);
                walk(txtPath.getText(), sizeauthorownerFilter);
            } else {
                getHasil(txtPath.getText(), sizeauthorFilter, true);
                walk(txtPath.getText(), sizeauthorFilter);
            }
        }
    } else {
        if (chkFiletype.isSelected()) {
            if (chkOwner.isSelected()) {
                getHasil(txtPath.getText(), sizeextensiownerFilter, true);
                walk(txtPath.getText(), sizeextensiownerFilter);
            } else {
                getHasil(txtPath.getText(), sizeextensiFilter, true);
                walk(txtPath.getText(), sizeextensiFilter);
            }
        } else {
            if (chkOwner.isSelected()) {
                getHasil(txtPath.getText(), sizeownerFilter, true);
                walk(txtPath.getText(), sizeownerFilter);
            } else {
                getHasil(txtPath.getText(), sizeFilter, true);
                walk(txtPath.getText(), sizeFilter);
            }
        }
    }
     if(tabelHasil.getRowCount()<=0){
        JOptionPane.showMessageDialog(null, "Tidak ada file yang ditemukan");
    }
} else {
    jTabbedPane2.setSelectedIndex(2);
    if (chkAuthor.isSelected()) {
        if (chkFiletype.isSelected()) {
            if (chkOwner.isSelected()) {
                getHasil(txtPath.getText(), authorextensiowner, true);
                walk(txtPath.getText(), authorextensiowner);
            } else {
                getHasil(txtPath.getText(), authorextensi, true);
                walk(txtPath.getText(), authorextensi);
            }
        } else {
            if (chkOwner.isSelected()) {
                getHasil(txtPath.getText(), authorowner, true);
                walk(txtPath.getText(), authorowner);
            } else {
                getHasil(txtPath.getText(), author, true);
```

```
                        walk(txtPath.getText(), author);
                    }
                }
            } else {
                if (chkFiletype.isSelected()) {
                    if (chkOwner.isSelected()) {
                        getHasil(txtPath.getText(), extensiowner, true);
                        walk(txtPath.getText(), extensiowner);
                    } else {
                        getHasil(txtPath.getText(), extensi, true);
                        walk(txtPath.getText(), extensi);
                    }
                } else {
                    if (chkOwner.isSelected()) {
                        getHasil(txtPath.getText(), owner, true);
                        walk(txtPath.getText(), owner);
                    } else {
                        jTabbedPane2.setSelectedIndex(1);
                        JOptionPane.showMessageDialog(rootPane, "Pilih Jenis Korelasi terlebih
     dahulu"
                                + "Pada ComboBox diatas");
                    }
                }
            }
             if(tabelHasil.getRowCount()<=0){
                JOptionPane.showMessageDialog(null, "Tidak ada file yang ditemukan");
            }
        }
    }
}
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    jFileChooser2 = new JFileChooser();
    jFileChooser2.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
    jFileChooser2.setAcceptAllFileFilterUsed(false);
    int a = jFileChooser2.showOpenDialog(this);
    if (a == JFileChooser.APPROVE_OPTION) {
        File file = jFileChooser2.getSelectedFile();
        txtPath.setText(file.getAbsolutePath());
    }
}
private void chkOwnerMouseClicked(java.awt.event.MouseEvent evt) {
    if (chkOwner.isSelected()) {
        txtOwnerManual.setEnabled(true);
    } else {
        txtOwnerManual.setEnabled(false);
    }
}
private void chkOwnerMouseReleased(java.awt.event.MouseEvent evt) {
    if (chkOwner.isSelected()) {
        txtOwnerManual.setEnabled(true);
    } else {
        txtOwnerManual.setEnabled(false);
    }
}
private void cmbKorelasiActionPerformed(java.awt.event.ActionEvent evt) {
    if (cmbKorelasi.getSelectedIndex() == 0) {
        disablekorelasi(false);
    } else {
        disablekorelasi(true);
        if (cmbKorelasi.getSelectedIndex() == 1) {
            txtKorelasi.setText(cmbKorelasi.getSelectedItem().toString()     +     ":     "     +
     tabelMeta.getModel().getValueAt(0, 2).toString());
            jPanel11.setVisible(false);
            jPanel12.setVisible(true);
        } else if (cmbKorelasi.getSelectedIndex() == 2) {
            txtKorelasi.setText(cmbKorelasi.getSelectedItem().toString()     +     ":     "     +
     tabelMeta.getModel().getValueAt(1, 2).toString());
            jPanel11.setVisible(false);
            jPanel12.setVisible(true);
        } else if (cmbKorelasi.getSelectedIndex() == 3) {
            txtKorelasi.setText(cmbKorelasi.getSelectedItem().toString()     +     ":     "     +
     tabelMeta.getModel().getValueAt(2, 2).toString());
            jPanel11.setVisible(false);
            jPanel12.setVisible(true);
        } else if (cmbKorelasi.getSelectedIndex() == 4) {
            txtKorelasi.setText(cmbKorelasi.getSelectedItem().toString()     +     ":     "     +
     tabelMeta.getModel().getValueAt(7, 2).toString());
```

```
            jPanel11.setVisible(true);
            jPanel12.setVisible(false);
        }
    }
}
private void rbantaraActionPerformed(java.awt.event.ActionEvent evt) {
}
private void chkAuthorActionPerformed(java.awt.event.ActionEvent evt) {
}
public static void main(String args[]) {
    try {
        for              (javax.swing.UIManager.LookAndFeelInfo              info              :
     javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Windows".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
     java.util.logging.Logger.getLogger(Home.class.getName()).log(java.util.logging.Level.SEVERE
     , null, ex);
    }               catch            (InstantiationException              ex)               {
     java.util.logging.Logger.getLogger(Home.class.getName()).log(java.util.logging.Level.SEVERE
     , null, ex);
    }               catch              (IllegalAccessException              ex)               {
     java.util.logging.Logger.getLogger(Home.class.getName()).log(java.util.logging.Level.SEVERE
     , null, ex);
    }          catch          (javax.swing.UnsupportedLookAndFeelException          ex)          {
     java.util.logging.Logger.getLogger(Home.class.getName()).log(java.util.logging.Level.SEVERE
     , null, ex);
    }
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new Home().setVisible(true);
        }
    });
}
private javax.swing.ButtonGroup buttonGroup1;
private javax.swing.JCheckBox chkAuthor;
private javax.swing.JCheckBox chkFiletype;
private javax.swing.JCheckBox chkOwner;
private javax.swing.JComboBox cmbKorelasi;
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JFileChooser jFileChooser1;
private javax.swing.JFileChooser jFileChooser2;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel11;
private javax.swing.JLabel jLabel12;
private javax.swing.JLabel jLabel13;
private javax.swing.JLabel jLabel14;
private javax.swing.JLabel jLabel15;
private javax.swing.JLabel jLabel16;
private javax.swing.JLabel jLabel17;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel10;
private javax.swing.JPanel jPanel11;
private javax.swing.JPanel jPanel12;
private javax.swing.JPanel jPanel13;
private javax.swing.JPanel jPanel2;
private javax.swing.JPanel jPanel3;
private javax.swing.JPanel jPanel4;
private javax.swing.JPanel jPanel5;
private javax.swing.JPanel jPanel6;
private javax.swing.JPanel jPanel7;
private javax.swing.JPanel jPanel8;
private javax.swing.JPanel jPanel9;
private javax.swing.JScrollPane jScrollPane1;
```

```
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JScrollPane jScrollPane3;
private javax.swing.JTabbedPane jTabbedPane1;
private javax.swing.JTabbedPane jTabbedPane2;
private javax.swing.JRadioButton rbBesar;
private javax.swing.JRadioButton rbKecil;
private javax.swing.JRadioButton rbSama;
private javax.swing.JRadioButton rbSamaBesar;
private javax.swing.JRadioButton rbSamaKecil;
private javax.swing.JRadioButton rbantara;
private javax.swing.JRadioButton rbantara2;
private javax.swing.JTable tabelHasil;
private javax.swing.JTable tabelMeta;
private com.toedter.calendar.JDateChooser tglAnataraAkhir;
private com.toedter.calendar.JDateChooser tglAntaraAwal;
private javax.swing.JFormattedTextField txtAntaraAkhir;
private javax.swing.JFormattedTextField txtAntaraAwal;
private javax.swing.JTextField txtAuthors;
private javax.swing.JTextField txtComputer;
private javax.swing.JTextField txtKorelasi;
private javax.swing.JTextField txtLokasi;
private javax.swing.JTextField txtNamaFile;
private javax.swing.JTextField txtNamaFile2;
private javax.swing.JTextField txtOwner;
private javax.swing.JTextField txtOwnerManual;
private javax.swing.JTextField txtPath;
private javax.swing.JTextArea txtSUM;
private javax.swing.JTextField txtTypeFile;
String header[] = {"NO", "JENIS META", "VALUE"};
DefaultTableModel tModel = new DefaultTableModel(header, 0);
long size = 0, lastmodifide = 0, lastaccess = 0, creationtime = 0, sizeawal = 0, sizeakhir = 0,
        tglawal = 0, tglakhir = 0;
public void getMetaData(File f, String namafile) {
    tModel = new DefaultTableModel(header, 0);
    Path file = Paths.get(f.getAbsolutePath());
    try {
        FileInfo filein = new FileInfo(f);
        try {
            txtOwner.setText(filein.getOwner());
        } catch (ParseException ex) {
        }
        txtLokasi.setText(f.getPath());
        txtNamaFile2.setText(namafile);
        txtTypeFile.setText(getFileExtension(f));
        txtComputer.setText(getComputerName());
        FileOwnerAttributeView   ownerAttributeView   =   Files.getFileAttributeView(f.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = ownerAttributeView.getOwner();
     txtAuthors.setText(owner.getName().substring(owner.getName().lastIndexOf("\\") + 1));
        String checksum = "MD5 :\n" + getChecksum(f.getPath(), "MD5")
                + "\nSHA-256 :\n" + getChecksum(f.getPath(), "SHA-256");
        txtSUM.setText(checksum);
        BasicFileAttributes attr = Files.readAttributes(file, BasicFileAttributes.class);
        String data1[] = {"1", "creationTime: ", attr.creationTime().toString()};
        tModel.addRow(data1);
        creationtime = attr.creationTime().toMillis();
        String data8[] = {"2", "lastAccessTime: ", attr.lastAccessTime().toString()};
        tModel.addRow(data8);
        lastaccess = attr.lastAccessTime().toMillis();
        String data2[] = {"3", "lastModifiedTime: ", attr.lastModifiedTime().toString()};
        tModel.addRow(data2);
        lastmodifide = attr.lastModifiedTime().toMillis();
        String data3[] = {"4", "isDirectory: ", String.valueOf(attr.isDirectory())};
        tModel.addRow(data3);
        String data4[] = {"5", "isOther: ", String.valueOf(attr.isOther())};
        tModel.addRow(data4);
        String data5[] = {"6", "isRegularFile: ", String.valueOf(attr.isRegularFile())};
        tModel.addRow(data5);
        String data6[] = {"7", "isSymbolicLink: ", String.valueOf(attr.isSymbolicLink())};
        tModel.addRow(data6);
        String data7[] = {"8", "size: ", String.valueOf(attr.size())};
        size = attr.size();
        tModel.addRow(data7);
    } catch (IOException ex) {
    }
    tabelMeta.setModel(tModel);
    AturLebarKolom alk = new AturLebarKolom(tabelMeta);
```

```
    alk.adjustColumns();
}
String headerHasil[] = {"NAME FILE", "SIZE", "DATE", "FOLDER PATH"};
DefaultTableModel tModelHasil = new DefaultTableModel(headerHasil, 0);
SimpleDateFormat sdf = new SimpleDateFormat("dd MMMM YYYY");
public void walk(String path, FileFilter dd) {
    File root = new File(path);
    File[] list = root.listFiles();
    if (list == null) {
        return;
    }
    for (File f : list) {
        if (f.isDirectory()) {
            walk(f.getAbsolutePath(), dd);
            getHasil(f.getAbsolutePath(), dd, false);
        }
    }
}
public void getHasil(String dirPath, FileFilter ff, boolean cek) {
    if (cek) {
        tModelHasil = new DefaultTableModel(headerHasil, 0);
    }
    File dir = new File(dirPath);
    File[] files = dir.listFiles(ff);
    try {
        for (File aFile : files) {
            String data[] = {aFile.getName(), String.valueOf(aFile.length()),
                sdf.format(new Date(aFile.lastModified())), aFile.getPath()};
            tModelHasil.addRow(data);
        }
    } catch (Exception ex) {
        System.out.println("Folder : " + dirPath + " Hasil = " + files);
    }
    tabelHasil.setModel(tModelHasil);
    AturLebarKolom alk = new AturLebarKolom(tabelHasil);
    alk.adjustColumns();
}
int perbandingan = 1;
FileFilter sizeFilter = new FileFilter() {
    public boolean accept(File file) {
        if (file.isFile()) {
            if (file.length() == size && perbandingan == 1) {
                return true;
            } else if (file.length() < size && perbandingan == 2) {
                return true;
            } else if (file.length() > size && perbandingan == 3) {
                return true;
            } else if (file.length() <= size && perbandingan == 4) {
                return true;
            } else if (file.length() >= size && perbandingan == 5) {
                return true;
            } else if ((file.length() >= sizeawal && file.length() <= sizeakhir) && perbandingan
     == 6) {
                return true;
            } else {
                return false;
            }
        } else {
            return false;
        }
    }
};
FileFilter tglFilter = new FileFilter() {
    public boolean accept(File file) {
        if (file.isFile()) {
            if (file.lastModified() == lastmodifide && perbandingan == 1) {
                return true;
            } else if (file.lastModified() < lastmodifide && perbandingan == 2) {
                return true;
            } else if (file.lastModified() > lastmodifide && perbandingan == 3) {
                return true;
            } else if (file.lastModified() <= lastmodifide && perbandingan == 4) {
                return true;
            } else if (file.lastModified() >= lastmodifide && perbandingan == 5) {
                return true;
            } else if ((file.lastModified() >= tglawal && file.lastModified() <= tglakhir) &&
     perbandingan == 7) {
```

```
                return true;
            } else {
                return false;
            }

        } else {
            return false;
        }
    }
};
FileFilter extensi = new FileFilter() {
    @Override
    public boolean accept(File file) {
        if (file.isFile()) {
            return getFileExtension(file).equalsIgnoreCase(txtTypeFile.getText().trim());
        } else {
            return false;
        }
    }
};
FileFilter author = new FileFilter() {
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtAuthors.getText().trim())) {
                    return true;
                } else {
                    return false;
                }
            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
FileFilter owner = new FileFilter() {
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtOwnerManual.getText().trim())) {
                    return true;
                } else {
                    return false;
                }

            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
FileFilter sizeauthorFilter = new FileFilter() {
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtAuthors.getText().trim())) {
                    if (file.length() == size && perbandingan == 1) {
                        return true;
```

```
                    } else if (file.length() < size && perbandingan == 2) {
                        return true;
                    } else if (file.length() > size && perbandingan == 3) {
                        return true;
                    } else if (file.length() <= size && perbandingan == 4) {
                        return true;
                    } else if (file.length() >= size && perbandingan == 5) {
                        return true;
                    } else if ((file.length() >= sizeawal && file.length() <= sizeakhir) &&
     perbandingan == 6) {
                        return true;
                    } else {
                        return false;
                    }
                } else {
                    return false;
                }

            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
FileFilter sizeauthorextensiFilter = new FileFilter() {
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtAuthors.getText().trim())) {
                    if (getFileExtension(file).equalsIgnoreCase(txtTypeFile.getText().trim())) {
                        if (file.length() == size && perbandingan == 1) {
                            return true;
                        } else if (file.length() < size && perbandingan == 2) {
                            return true;
                        } else if (file.length() > size && perbandingan == 3) {
                            return true;
                        } else if (file.length() <= size && perbandingan == 4) {
                            return true;
                        } else if (file.length() >= size && perbandingan == 5) {
                            return true;
                        } else if ((file.length() >= sizeawal && file.length() <= sizeakhir) &&
     perbandingan == 6) {
                            return true;
                        } else {
                            return false;
                        }
                    } else {
                        return false;
                    }
                } else {
                    return false;
                }

            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
FileFilter sizeauthorextensiownerFilter = new FileFilter() {
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
```

```
                if (dd.equalsIgnoreCase(txtAuthors.getText().trim())) {
                    if (getFileExtension(file).equalsIgnoreCase(txtTypeFile.getText().trim())) {
                        if (dd.equalsIgnoreCase(txtOwnerManual.getText().trim())) {
                            if (file.length() == size && perbandingan == 1) {
                                return true;
                            } else if (file.length() < size && perbandingan == 2) {
                                return true;
                            } else if (file.length() > size && perbandingan == 3) {
                                return true;
                            } else if (file.length() <= size && perbandingan == 4) {
                                return true;
                            } else if (file.length() >= size && perbandingan == 5) {
                                return true;
                            } else if ((file.length() >= sizeawal && file.length() <= sizeakhir)
     && perbandingan == 6) {
                                return true;
                            } else {
                                return false;
                            }
                        } else {
                            return false;
                        }
                    } else {
                        return false;
                    }
                } else {
                    return false;
                }
            } catch (IOException ex) {
                return false;
            }

        }
};
FileFilter sizeauthorownerFilter = new FileFilter() {
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtAuthors.getText().trim())) {
                    if (dd.equalsIgnoreCase(txtOwnerManual.getText().trim())) {
                        if (file.length() == size && perbandingan == 1) {
                            return true;
                        } else if (file.length() < size && perbandingan == 2) {
                            return true;
                        } else if (file.length() > size && perbandingan == 3) {
                            return true;
                        } else if (file.length() <= size && perbandingan == 4) {
                            return true;
                        } else if (file.length() >= size && perbandingan == 5) {
                            return true;
                        } else if ((file.length() >= sizeawal && file.length() <= sizeakhir) &&
     perbandingan == 6) {
                            return true;
                        } else {
                            return false;
                        }
                    } else {
                        return false;
                    }
                } else {
                    return false;
                }
            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
```

```
    }
};
FileFilter sizeextensiownerFilter = new FileFilter() {
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (getFileExtension(file).equalsIgnoreCase(txtTypeFile.getText().trim())) {
                    if (dd.equalsIgnoreCase(txtOwnerManual.getText().trim())) {
                        if (file.length() == size && perbandingan == 1) {
                            return true;
                        } else if (file.length() < size && perbandingan == 2) {
                            return true;
                        } else if (file.length() > size && perbandingan == 3) {
                            return true;
                        } else if (file.length() <= size && perbandingan == 4) {
                            return true;
                        } else if (file.length() >= size && perbandingan == 5) {
                            return true;
                        } else if ((file.length() >= sizeawal && file.length() <= sizeakhir) &&
     perbandingan == 6) {
                            return true;
                        } else {
                            return false;
                        }
                    } else {
                        return false;
                    }
                } else {
                    return false;
                }
            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }

    }
};
FileFilter sizeextensiFilter = new FileFilter() {
    @Override
    public boolean accept(File file) {
        if (file.isFile()) {
            if (getFileExtension(file).equalsIgnoreCase(txtTypeFile.getText().trim())) {
                if (file.length() == size && perbandingan == 1) {
                    return true;
                } else if (file.length() < size && perbandingan == 2) {
                    return true;
                } else if (file.length() > size && perbandingan == 3) {
                    return true;
                } else if (file.length() <= size && perbandingan == 4) {
                    return true;
                } else if (file.length() >= size && perbandingan == 5) {
                    return true;
                }  else  if  ((file.length()  >=  sizeawal  &&  file.length()  <=  sizeakhir)  &&
     perbandingan == 6) {
                    return true;
                } else {
                    return false;
                }
            } else {
                return false;
            }
        } else {
            return false;
        }
    }
};
FileFilter sizeownerFilter = new FileFilter() {
    public boolean accept(File file) {
```

```
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {

                if (dd.equalsIgnoreCase(txtOwnerManual.getText().trim())) {
                    if (file.length() == size && perbandingan == 1) {
                        return true;
                    } else if (file.length() < size && perbandingan == 2) {
                        return true;
                    } else if (file.length() > size && perbandingan == 3) {
                        return true;
                    } else if (file.length() <= size && perbandingan == 4) {
                        return true;
                    } else if (file.length() >= size && perbandingan == 5) {
                        return true;
                    } else if ((file.length() >= sizeawal && file.length() <= sizeakhir) &&
     perbandingan == 6) {
                        return true;
                    } else {
                        return false;
                    }
                } else {
                    return false;
                }
            } else {
                return false;
            }

        } catch (IOException ex) {
            return false;
        }

    }
};
FileFilter tglauthorFilter = new FileFilter() {
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtAuthors.getText().trim())) {
                    if (file.lastModified() == lastmodifide && perbandingan == 1) {
                        return true;
                    } else if (file.lastModified() < lastmodifide && perbandingan == 2) {
                        return true;
                    } else if (file.lastModified() > lastmodifide && perbandingan == 3) {
                        return true;
                    } else if (file.lastModified() <= lastmodifide && perbandingan == 4) {
                        return true;
                    } else if (file.lastModified() >= lastmodifide && perbandingan == 5) {
                        return true;
                    } else if ((file.lastModified() >= tglawal && file.lastModified() <= tglakhir)
     && perbandingan == 7) {
                        return true;
                    } else {
                        return false;
                    }
                } else {
                    return false;
                }

            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
FileFilter tglauthorextensiFilter = new FileFilter() {
```

```
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtAuthors.getText().trim())) {
                    if (getFileExtension(file).equalsIgnoreCase(txtTypeFile.getText().trim())) {
                        if (file.lastModified() == lastmodifide && perbandingan == 1) {
                            return true;
                        } else if (file.lastModified() < lastmodifide && perbandingan == 2) {
                            return true;
                        } else if (file.lastModified() > lastmodifide && perbandingan == 3) {
                            return true;
                        } else if (file.lastModified() <= lastmodifide && perbandingan == 4) {
                            return true;
                        } else if (file.lastModified() >= lastmodifide && perbandingan == 5) {
                            return true;
                        } else if ((file.lastModified() >= tglawal && file.lastModified() <=
     tglakhir) && perbandingan == 7) {
                            return true;
                        } else {
                            return false;
                        }
                    } else {
                        return false;
                    }
                } else {
                    return false;
                }

            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
FileFilter tglauthorextensiownerFilter = new FileFilter() {
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtAuthors.getText().trim())) {
                    if (dd.equalsIgnoreCase(txtOwnerManual.getText().trim())) {
                        if
     (getFileExtension(file).equalsIgnoreCase(txtTypeFile.getText().trim())) {
                            if (file.lastModified() == lastmodifide && perbandingan == 1) {
                                return true;
                            } else if (file.lastModified() < lastmodifide && perbandingan == 2) {
                                return true;
                            } else if (file.lastModified() > lastmodifide && perbandingan == 3) {
                                return true;
                            } else if (file.lastModified() <= lastmodifide && perbandingan == 4)
     {
                                return true;
                            } else if (file.lastModified() >= lastmodifide && perbandingan == 5)
     {
                                return true;
                            } else if ((file.lastModified() >= tglawal && file.lastModified() <=
     tglakhir) && perbandingan == 7) {
                                return true;
                            } else {
                                return false;
                            }
                        } else {
                            return false;
                        }
                    } else {
                        return false;
```

```
                    }
                } else {
                    return false;
                }

            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
FileFilter tglextensiownerFilter = new FileFilter() {
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {

                if (dd.equalsIgnoreCase(txtOwnerManual.getText().trim())) {
                    if (getFileExtension(file).equalsIgnoreCase(txtTypeFile.getText().trim())) {
                        if (file.lastModified() == lastmodifide && perbandingan == 1) {
                            return true;
                        } else if (file.lastModified() < lastmodifide && perbandingan == 2) {
                            return true;
                        } else if (file.lastModified() > lastmodifide && perbandingan == 3) {
                            return true;
                        } else if (file.lastModified() <= lastmodifide && perbandingan == 4) {
                            return true;
                        } else if (file.lastModified() >= lastmodifide && perbandingan == 5) {
                            return true;
                        } else if ((file.lastModified() >= tglawal && file.lastModified() <=
     tglakhir) && perbandingan == 7) {
                            return true;
                        } else {
                            return false;
                        }
                    } else {
                        return false;
                    }

                } else {
                    return false;
                }

            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
FileFilter tglauthorownerFilter = new FileFilter() {
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtAuthors.getText().trim())) {
                    if (dd.equalsIgnoreCase(txtOwnerManual.getText().trim())) {
                        if (file.lastModified() == lastmodifide && perbandingan == 1) {
                            return true;
                        } else if (file.lastModified() < lastmodifide && perbandingan == 2) {
                            return true;
                        } else if (file.lastModified() > lastmodifide && perbandingan == 3) {
                            return true;
                        } else if (file.lastModified() <= lastmodifide && perbandingan == 4) {
                            return true;
                        } else if (file.lastModified() >= lastmodifide && perbandingan == 5) {
```

```
                            return true;
                        } else if ((file.lastModified() >= tglawal && file.lastModified() <=
    tglakhir) && perbandingan == 7) {
                            return true;
                        } else {
                            return false;
                        }
                    } else {
                        return false;
                    }
                } else {
                    return false;
                }

            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
FileFilter tglownerFilter = new FileFilter() {
    public boolean accept(File file) {
        FileOwnerAttributeView ownerAttributeView = Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtOwnerManual.getText().trim())) {
                    if (file.lastModified() == lastmodifide && perbandingan == 1) {
                        return true;
                    } else if (file.lastModified() < lastmodifide && perbandingan == 2) {
                        return true;
                    } else if (file.lastModified() > lastmodifide && perbandingan == 3) {
                        return true;
                    } else if (file.lastModified() <= lastmodifide && perbandingan == 4) {
                        return true;
                    } else if (file.lastModified() >= lastmodifide && perbandingan == 5) {
                        return true;
                    } else if ((file.lastModified() >= tglawal && file.lastModified() <= tglakhir)
     && perbandingan == 7) {
                        return true;
                    } else {
                        return false;
                    }
                } else {
                    return false;
                }
            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
FileFilter tglextensiFilter = new FileFilter() {
    @Override
    public boolean accept(File file) {
        if (file.isFile()) {
            if (getFileExtension(file).equalsIgnoreCase(txtTypeFile.getText().trim())) {
                if (file.lastModified() == lastmodifide && perbandingan == 1) {
                    return true;
                } else if (file.lastModified() < lastmodifide && perbandingan == 2) {
                    return true;
                } else if (file.lastModified() > lastmodifide && perbandingan == 3) {
                    return true;
                } else if (file.lastModified() <= lastmodifide && perbandingan == 4) {
                    return true;
                } else if (file.lastModified() >= lastmodifide && perbandingan == 5) {
                    return true;
                } else if ((file.lastModified() >= tglawal && file.lastModified() <= tglakhir) &&
     perbandingan == 7) {
                    return true;
```

```
                } else {
                    return false;
                }
            } else {
                return false;
            }
        } else {
            return false;
        }
    }
};
FileFilter authorextensi = new FileFilter() {
    @Override
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtAuthors.getText().trim())) {
                    return
     getFileExtension(file).equalsIgnoreCase(txtTypeFile.getText().trim());
                } else {
                    return false;
                }
            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
FileFilter authorextensiowner = new FileFilter() {
    @Override
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtAuthors.getText().trim())) {
                    if (dd.equalsIgnoreCase(txtOwner.getText().trim())) {
                        return
     getFileExtension(file).equalsIgnoreCase(txtTypeFile.getText().trim());
                    } else {
                        return false;
                    }
                } else {
                    return false;
                }
            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
FileFilter authorowner = new FileFilter() {

    @Override
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtAuthors.getText().trim())) {
                    if (dd.equalsIgnoreCase(txtOwner.getText().trim())) {
                        return true;
```

```
                    } else {
                        return false;
                    }
                } else {
                    return false;
                }
            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
FileFilter extensiowner = new FileFilter() {
    @Override
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtOwner.getText().trim())) {
                    return
     getFileExtension(file).equalsIgnoreCase(txtTypeFile.getText().trim());
                } else {
                    return false;
                }
            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
private String getFileExtension(File file) {
    String name = file.getName();
    try {
        return name.substring(name.lastIndexOf(".") + 1);
    } catch (Exception e) {
        return "";
    }
}
public byte[] createChecksum(String filename, String type) throws Exception {
    InputStream fis = new FileInputStream(filename);
    byte[] buffer = new byte[1024];
    MessageDigest complete = MessageDigest.getInstance(type);
    int numRead;
    do {
        numRead = fis.read(buffer);
        if (numRead > 0) {
            complete.update(buffer, 0, numRead);
        }
    } while (numRead != -1);

    fis.close();
    return complete.digest();
}
public String getChecksum(String filename, String type) {
    String result = "";
    byte[] b;
    try {
        b = createChecksum(filename, type);
        for (int i = 0; i < b.length; i++) {
            result += Integer.toString((b[i] & 0xff) + 0x100, 16).substring(1);
        }
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(null, "GAGAL cek CheckSum");
    }
    return result;
}
private String getComputerName() {
    Map<String, String> env = System.getenv();
    if (env.containsKey("COMPUTERNAME")) {
```

```
            return env.get("COMPUTERNAME");
        } else if (env.containsKey("HOSTNAME")) {
            return env.get("HOSTNAME");
        } else {
            return "Unknown Computer";
        }
    }
    public void kosong() {
        txtAuthors.setText(null);
        txtComputer.setText(null);
        txtLokasi.setText(null);
        txtNamaFile2.setText(null);
        txtOwner.setText(null);
        txtPath.setText(null);
        txtSUM.setText(null);
        txtTypeFile.setText(null);
        txtAntaraAkhir.setText(null);
        txtAntaraAwal.setText(null);
        tModel = new DefaultTableModel(header, 0);
        tabelMeta.setModel(tModel);
        tModelHasil = new DefaultTableModel(headerHasil, 0);
        tabelHasil.setModel(tModelHasil);
    }
    private void disablekorelasi(boolean status) {
        txtKorelasi.setText("");
        txtKorelasi.setEnabled(status);
        rbSama.setEnabled(status);
        rbBesar.setEnabled(status);
        rbKecil.setEnabled(status);
        rbSamaBesar.setEnabled(status);
        rbSamaKecil.setEnabled(status);
        rbantara.setEnabled(status);
        rbantara2.setEnabled(status);
        txtAntaraAwal.setEnabled(status);
        txtAntaraAkhir.setEnabled(status);
        tglAnataraAkhir.setDate(new Date());
        tglAntaraAwal.setDate(new Date());
        rbSama.setSelected(status);
    }
}
```