Pendeteksi *Error* dengan CRC32 dan Cek Integritas dengan SHA256 pada Aplikasi Pengunduh dan Transfer File

Bagus P. S. Hutomo[1], Hartanto K. Wardana[2], Banu W. Yohanes[3]

Program Studi Sistem Komputer,
Fakultas Teknik Elektronika dan Komputer,
Universitas Kristen Satya Wacana, Salatiga
[1]622013007@student.uksw.edu, [2]hartanto.kusuma@staff.uksw.edu,
[3]banu.yohanes@staff.uksw.edu

Ringkasan

Seiring file yang diunduh dari Internet, maka dibutuhkan pengunduh yang dapat melakukan deteksi *error* dan cek integritas data secara otomatis setelah proses unduh selesai. Selain file unduh, pengecekan integritas juga dilakukan setelah proses transfer file pada jaringan lokal. Hal ini diperlukan untuk memastikan keutuhan dan integritas file yang diunduh maupun ditransfer. Pengecekan keutuhan dan integritas dilakukan dengan mengecek CRC32 dan SHA256 dari file yang diunduh maupun ditransfer. Hasil pengujian menunjukkan file yang tidak diubah dan file yang diubah menghasilkan nilai CRC32 dan SHA256 yang berbeda, sehingga pengecekan CRC32 dan SHA256 dapat digunakan untuk memastikan keutuhan dan integritas file.

Kata kunci: Deteksi *error* transfer file, CRC, Integritas file, SHA

1. Pendahuluan

Untuk mengecek keutuhan dan menjaga integritas dari file yang tersebar di Internet dibutuhkan aplikasi yang dapat mengecek *error* acak ketika file dikirimkan dan mengecek integritas setelah proses unduh selesai. Hal ini penting karena saat proses unduh maupun saat penulisan file pada media penyimpan file bisa mengalami perubahan bit yang disengaja, misalnya adanya *malware*, maupun tidak disengaja, misalnya gangguan pada jaringan. Sehingga file hasil unduh berbeda dari yang disediakan oleh server yang mengakibatkan file tidak bisa dibuka atau ada perubahan file. Selain gangguan saat proses unduh maupun simpan, file dapat diubah isinya atau ditanamkan program yang berbahaya seperti *malware* yang mengakibatkan kerugian.

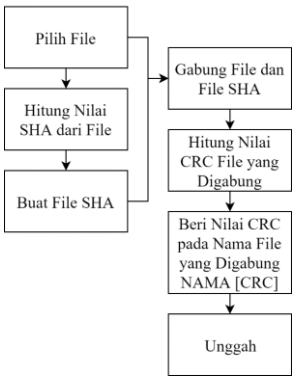
Selain file dari Internet, file juga bisa diperoleh dari jaringan lokal seperti dokumen atau laporan dalam suatu perusahaan. Dalam hal ini integritas dan keamanan data menjadi penting karena file bisa diubah seperti mengganti isi dari dokumen tersebut setelah disimpan di server lokal perusahaan tersebut. Berbagai kasus tersebut dapat diatasi dengan menambahkan fitur pendeteksi *error* dan pengecekan integritas setelah proses unduh dan transfer file, pengecekan *error* dan pengecekan integritas masing-masing dilakukan dengan metode *Cyclic Redundancy Check* (CRC) 32 dan algoritma *Secure Hash Algorithm* (SHA) 256 secara berurutan.

CRC32 digunakan karena bekerja dengan baik untuk mendeteksi *error* acak seperti interferensi jaringan, derau, dan distorsi. Selain itu jumlah *checksum* hanya 8 karakter dalam format heksadesimal sehingga jika ditambahkan pada nama file tidak menjadikan nama file terlalu panjang. SHA256 digunakan karena merupakan memiliki tingkat keamanan yang tinggi dan tingkat *collision* yang lebih rendah dibandingkan beberapa algoritma *hash* versi sebelumnya, MD5 dan SHA1 [1]. Sehingga SHA256 sesuai untuk memberikan sidik jari pada file unduhan yang berjumlah besar di Internet.

2. Perancangan Sistem

2.1. Pembuat File

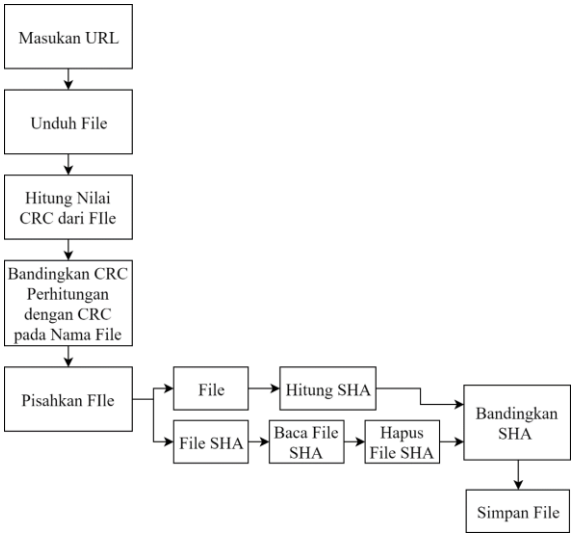
Pada pembuat file dibuat file khusus sebelum file diunggah, yaitu dengan menghitung dan membuat file SHA256 dari file yang dipilih. Kemudian menggabungkan file SHA-256 dengan file yang dipilih. Setelah itu, file dihitung nilai CRC32-nya, kemudian hasil CRC32 ditambahkan pada nama file. Perhitungan CRC32 diimplementasikan menggunakan kode *open-source* [2], sedangkan perhitungan SHA256 diimplementasikan menggunakan class *System.Security.Cryptography.SHA256* dari pustaka atau *application programming interface* (API) .Net 2017 [3], untuk penggabung dan pemisah file menggunakan pustaka DotNetZip [4]. Cara kerja pembuat file ditunjukkan pada Gambar 1.



Gambar 1. Cara kerja pembuat file

2.2. Pengunduh File

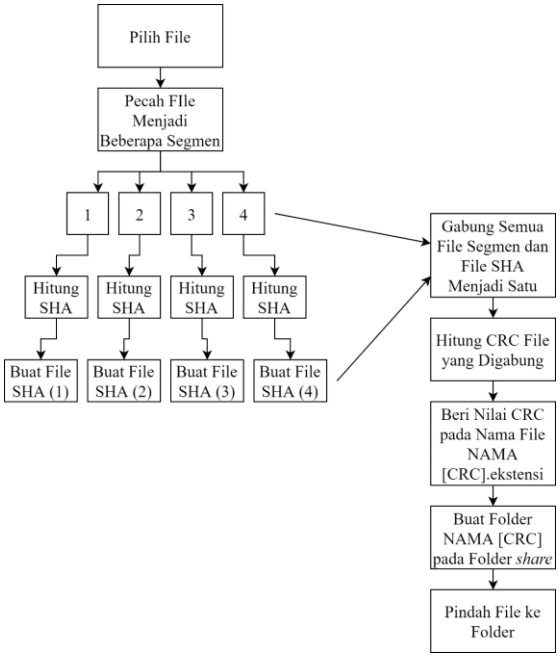
Pada modul pengunduh [5], sebuah file khusus yang sudah dibuat dihitung nilai CRC32 dan dibandingkan dengan CRC32 pada nama file. Jika hasil CRC sama maka tidak terdapat *error* acak ketika transmisi atau penulisan file, dan diberikan status file "OK", jika hasil CRC tidak sama maka terdapat *error* acak dan diberi status "not OK". Selanjutnya file dipisahkan menjadi 2 bagian, yaitu file SHA256 dan file yang dipilih. Selanjutnya pada bagian file dihitung nilai SHA256-nya dan dibandingkan dengan file SHA256 untuk verifikasi. Jika hasil *hash* sama, maka file utuh dikirimkan atau ditulis, dan diberi status file "OK", jika hasil *hash* tidak sama maka terdapat perubahan atau ketidakutuhan file dan diberi status "not OK". Cara kerja pengunduh file ditunjukkan pada Gambar 2.



Gambar 2. Cara kerja pengunduh file

2.3. Transfer File pada Sisi *Server*

Pada modul transfer file dipisahkan menjadi 2 bagian, yaitu *server* dan *client*. Pada sisi *server* dibuat file khusus dengan membagi file menjadi beberapa segmen, kemudian menghitung dan membuat file SHA256 dari tiap segmen, selanjutnya semua file digabungkan menjadi satu dan file gabungan dihitung nilai CRC32nya, lalu ditambahkan nilai CRC32 pada nama file gabungan. Pemecah dan penyatu file diimplementasikan dengan kode *open-source*[6]. Cara kerja transfer file pada sisi *server* ditunjukkan oleh Gambar 3.

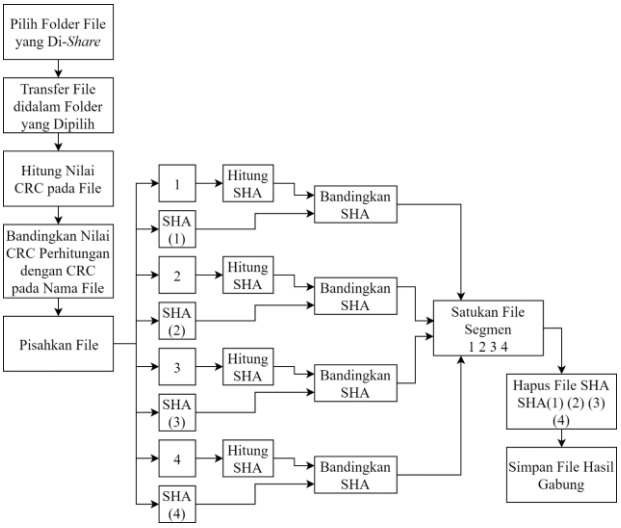


Gambar 3. Cara kerja transfer file pada sisi *server*

2.4. Transfer File pada Sisi *Client*

Modul transfer file di sisi *client* menerima file yang dibuat *server*, dan dicek nilai CRC32 dan dibandingkan dengan CRC32 pada nama file, jika sama status "OK", jika

tidak "not OK". Kemudian file dipisah sehingga ada file segmen dan file SHA256, selanjutnya setiap file segmen dihitung SHA256-nya dan dibandingkan dengan file SHA256 tiap segmen, jika sama status "OK", jika tidak status "not OK". Cara kerja transfer file pada sisi *client* ditunjukkan pada Gambar 4.

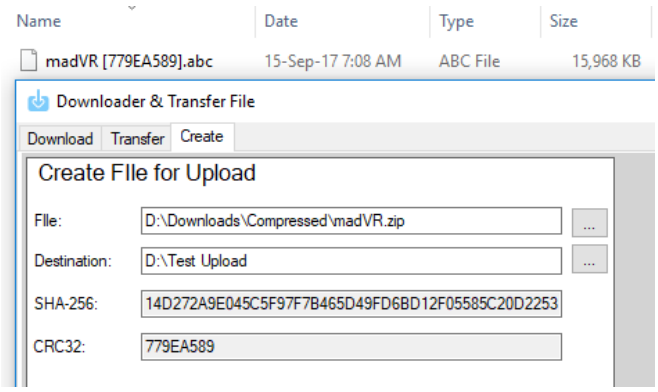


Gambar 4. Cara kerja transfer file pada sisi *client*

3. Pengujian dan Analisis

3.1. Pembuat File

Pada modul pembuat file setelah dibangkitkan nilai SHA256 dari file dan CRC32 dari file gabungan, semuanya ditampilkan pada antarmuka. Kemudian file disalin dan diubah isinya untuk pengujian cek CRC32 dan SHA256. Setelah itu, file diunggah pada file hosting. Antarmuka pengguna pada modul pembuat file ditampilkan pada Gambar 5.

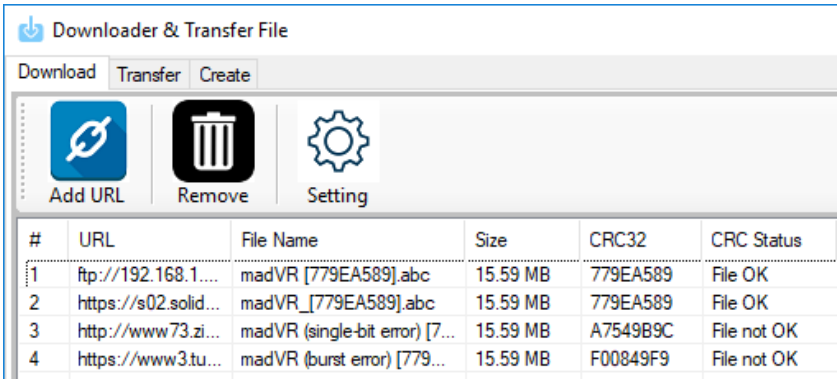


Gambar 5. Antarmuka modul pembuat file

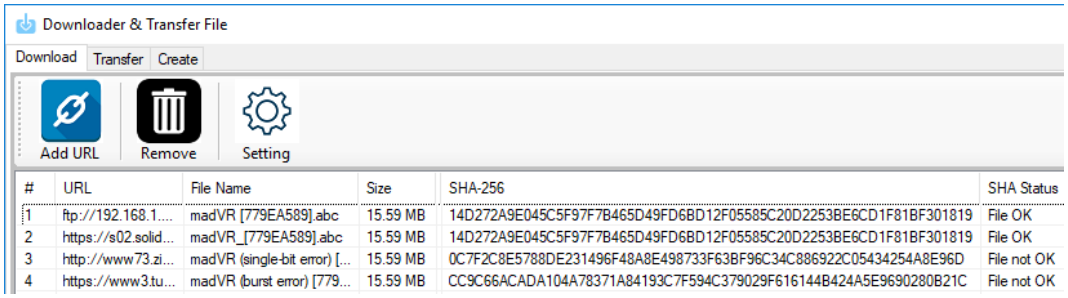
3.2. Pengunduh File

Pada modul pengunduh file, file hasil unggahan akan diunduh dan hasil dari cek nilai CRC32 dan SHA256 ditampilkan oleh program. Untuk file yang tidak diubah CRC status menunjukkan "FIle OK", sementara file mengalami *error* acak ketika transmisi atau penulisan menunjukkan "File not OK", antarmukanya ditampilkan pada Gambar 6. Untuk hasil cek SHA256 jika file tidak diubah, maka hasil SHA status menunjukkan

"File OK", dan jika file diubah, maka status menunjukkan "File not OK", antarmuka hasil cek SHA256 terdapat pada Gambar 7.



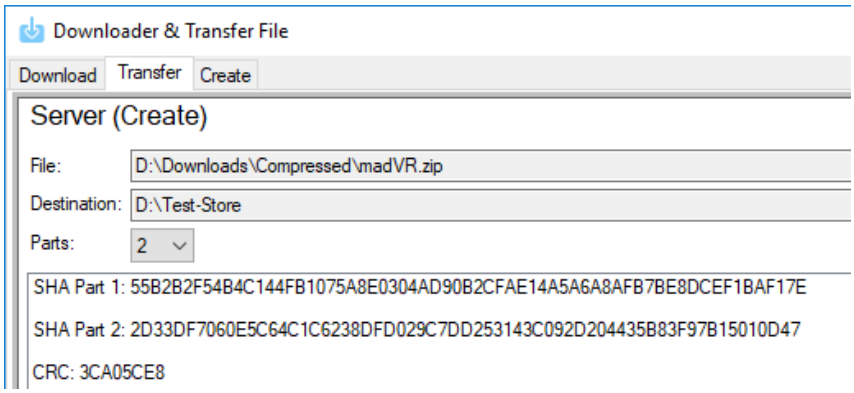
Gambar 6. Antarmuka hasil pengujian cek CRC32 pada pengunduh



Gambar 7. Antarmuka hasil pengujian cek SHA256 pada pengunduh

3.3. Transfer File pada Sisi *Server*

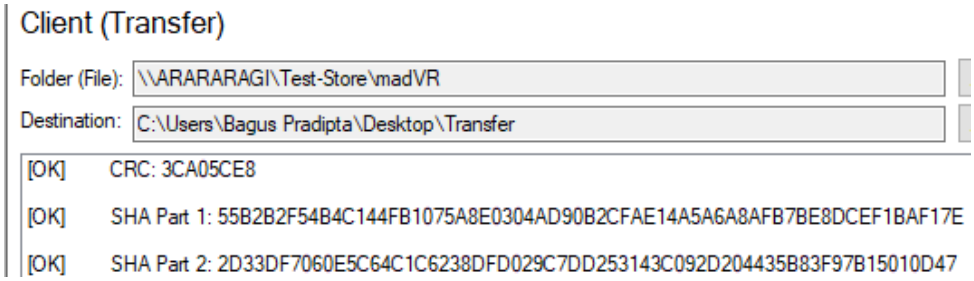
Pada *server* setelah file dibuat muncul nilai SHA256 untuk tiap segmen dan nilai CRC32 dari file setelah digabung ditampilkan pada antarmuka. Selanjutnya file disalin dan isi dari file segmen diubah untuk pengujian cek nilai CRC32 dan SHA256. Contoh tampilan pada modul transfer file di sisi *server* ditampilkan pada Gambar 8.



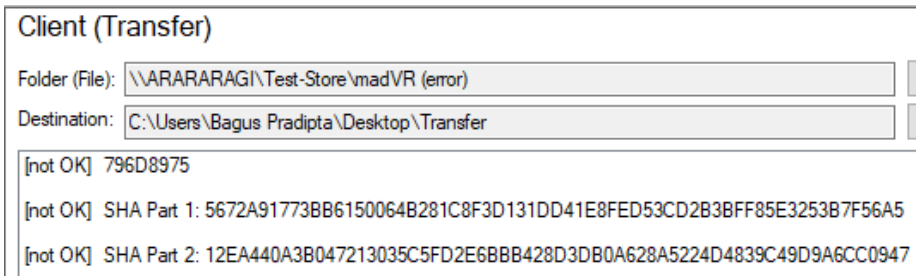
Gambar 8. Antarmuka hasil pengujian transfer file pada sisi *server*

3.4. Transfer File pada Sisi *Client*

Pada *client* hasil cek CRC32 dan SHA-256 ditampilkan oleh program. Hasil cek file yang tidak diubah menunjukkan status CRC32 dan SHA-257 "OK", hasil cek ditunjukkan Gambar 9 dan hasil cek file yang diubah menunjukkan status CRC32 dan SHA-256 "not OK" ditunjukkan Gambar 10.



Gambar 9. Antarmuka hasil pengujian transfer file yang tidak diubah pada sisi *client*



Gambar 10. Antarmuka hasil pengujian transfer file yang diubah pada sisi *client*

4. Kesimpulan

Metode CRC32 dan algoritma SHA256 dapat digunakan untuk mendeteksi *error* acak pada transmisi atau penulisan file serta perubahan pada file secara berurutan. Untuk lanjutan dapat dibuat perbandingan karakteristik dan performa kedua metode dalam menjaga integritas data. Karena file yang bisa diproses oleh aplikasi adalah file khusus yang dibuat oleh aplikasi, proteksi untuk keamanan data bisa dijamin selama file dikelola oleh administrator sistem.

Daftar Pustaka

[1] S. Marc, B. Elie, K. Pierre, A. Ange, M. Yarik, P. B. Alex, B. Clement, Announcing the first SHA1 collision, Google Security Blog [Online] https://security. googleblog.com/2017/02/announcing-first-sha1-collision.html, diakses tanggal 11 September 2017.

[2] G. Damien, DamienGKit/CSharp/DamienG.Library/Security/Cryptography/, [Online], https://github.com/damieng/DamienGKit/blob/master/CSharp/DamienG.Library/Security/Cryptography/Crc32.cs, diakses tanggal 7 Juni 2017.

[3] SHA256 Class, .NET Framework 4.7.2 [Online], https://msdn.microsoft.com/en-us/library/system.security.cryptography.sha256(v=vs.110).aspx, diakses tanggal 16 Juni 2017.

[4] DotNetZip - Zip and Unzip in C#, VB, any .NET language, CodePlex Archive Open Source Project Archive [Online], https://dotnetzip.codeplex.com/, diakses tanggal 20 Juni 2017.

[5] subena22jf, subena22jf/multipart.cs, GitHubGist, [Online], https://gist. github.com /subena22jf/3358b8609966203502a5, diakses tanggal 6 Agustus 2017.

[6] K. Vamshi, Split and Merge files in C# - C# Corner, [Online], http://www.c-sharpcorner.com/UploadFile/a72401/split-and-merge-files-in-C-Sharp/, diakses tanggal 10 Agustus 2017.