Journal of Computer System and Informatics (JoSYC)
ISSN 2714-8912 (media online), ISSN 2714-7150 (media cetak)
Volume 3, No. 2, Februari 2022, Page 89−92
https://ejurnal.seminar-id.com/index.php/josyc
DOI 10.47065/josyc.v3i2.954

Perancangan Aplikasi Deteksi Orisinalitas File Dokumen Menerapkan Algoritma Ripe-MD 128

Samsir

Fakultas Ilmu Komputer dan Teknologi Informasi, Prodi Teknik Informatika, Universitas Budi Darma, Medan, Indonesia
Email Penulis Korespondensi: samsir@gmail.com
Submitted: **11/09/2021**; Accepted: **25/02/2022**; Published: **28/02/2022**

Abstrak−Pendeteksian orisinalitas file dokumen ini dilakukan pada dokumen teks untuk mengetahui perubahan isi teks pada dokumen serta menghindari dari penjiplakan atau plagiarisme dari teks tersebut. Penjiplakan atau plagiarisme berarti mencontoh atau meniru atau mencuri tulisan dan karya orang lain yang kemudian diakui sebagai karangannya sendiri dengan ataupun tanpa seizin penulisnya. Penjiplakan file dokumen ini seringkali diterapkan oleh akademisi baik tingkat sekolah maupun perguruan tinggi. Untuk mengatasi permasalahan maka dibutuhkan suatu solusi untuk mendeteksi keaslian file dokumen tersebut. Proses pendeteksian orisinalitas file dokumen ini menggunakan metode Ripe-MD 120, yang mana output-nya berupa sekumpulan nilai hash yang didapatkan melalui metode tersebut. RIPE-MD dikembangkan untuk proyek RIPE masyarakat eropa. Peningkatan dan pengembangan ini dilakukan karena telah banyak usaha - usaha kriptanalisis yang telah dilakukan terhadap RIPE MD-128 dan berdasarkan informasi yang didapat maka diciptakanlah RIPE MD-160. RIPE MD-128, nilai hash yang dihasilkan hanya memiliki panjang 128 bit yang mampu mendeteksi keaslian sebuah file dokumen gambar.

Kata Kunci: Kriptografi; Orisinalitas; RipedMD 128

Abstract−Detection of the originality of this document file is carried out on text documents to find out changes in the contents of the text in the document and avoid plagiarism or plagiarism of the text. Plagiarism or plagiarism means imitating or imitating or stealing other people's writings and works which are later recognized as one's own writing with or without the permission of the author. Plagiarism of this document file is often applied by academics both at the school and university levels. To overcome the problem, a solution is needed to detect the authenticity of the document file. The process of detecting the originality of this document file uses the Ripe-MD 120 method, where the output is a set of hash values obtained through that method. RIPE-MD was developed for the European Community RIPE project. This improvement and development was carried out because many cryptanalytic efforts had been carried out on RIPE MD-128 and based on the information obtained, RIPE MD-160 was created. RIPE MD-128, the resulting hash value only has a length of 128 bits which is able to detect the authenticity of an image document file.

Keywords: Cryptography; Originality; RipedMD 128

1. PENDAHULUAN

Dewasa ini penggunaan teknologi Internet didunia sudah berkembang pesat. Semua kalangan telah menikmati Internet. Bahkan perkembangan teknologi Internet tersebut semakin memudahkan penggunanya dalam berkomunikasi melalui bermacam-macam media maupun aplikasi. Pemanfaatan teknologi digital telah menjadi kebutuhan dalam era modern saat ini. Komponen yang ada di dalam dunia digital salah satunya adalah *file* dokumen teks. *File* dokumen dalam bentuk digital memudahkan dalam hal penyimpanan, efisien, mudah dicari, bahkan mudah dalam hal penjiplakan.

Salah satu sifat dari data digital adalah mudah untuk dilakukan perubahan dibandingkan dengan data analog. Data digital juga dapat tersimpan secara permanen maupun tidak permanen serta dapat dilakukan proses *editing* atau perubahan berdasarkan keperluan dengan sangat mudah. Proses ini berbeda dengan data *analog*, tulisan tangan ataupun dokumen cetak, perubahan yang dilakukan akan menimbulkan jejak secara nyata pada dokumen tersebut.

Pendeteksian orisinalitas *file* dokumen ini dilakukan pada dokumen teks untuk mengetahui perubahan isi teks pada dokumen serta menghindari dari penjiplakan atau plagiarisme dari teks tersebut. Penjiplakan atau *plagiarisme* berarti mencontoh atau meniru atau mencuri tulisan dan karya orang lain yang kemudian diakui sebagai karangannya sendiri dengan ataupun tanpa seizin penulisnya. Penjiplakan *file* dokumen ini seringkali diterapkan oleh akademisi baik tingkat sekolah maupun perguruan tinggi. Tindakan *plagiat* yang dilakukan oleh siswa maupun mahasiswa ini sangat tidak mencerminkan sikap kreatif dan terpelajar sebagai kaum intelektual. Kadangkala tindak penjiplakan ini dimodifikasi dengan mengganti kata-kata yang mengandung sinonim, dengan maksud agar terlihat berbeda dari pekerjaan teman. Hal semacam ini dapat menimbulkan masalah terhadap evaluasi hasil belajar siswa/mahasiswa. Tindak penjiplakan dapat dilakukan dengan *modify* yang mana dengan mengubah beberapa bagian bahkan keseluruhan, yaitu dengan mengubah kata-kata dengan sinonim.

Untuk mengatasi permasalahan maka dibutuhkan suatu solusi untuk mendeteksi keaslian *file* dokumen tersebut. Proses pendeteksian orisinalitas *file* dokumen ini menggunakan metode Ripe-MD 120, yang mana *output*-nya berupa sekumpulan nilai *hash* yang didapatkan melalui metode tersebut. RIPE-MD dikembangkan untuk proyek RIPE masyarakat eropa. Algoritma ini adalah variasi dari algoritma MD4 yang dirancang untuk menahan serangan kriptanalisis yang ada dan menghasilkan nilai *hash* 128 *bit*. Rotasi dan urutan kata-kata pesan yang dimodifikasi. Selain itu, terdapat dua contoh dalam algoritma, berbeda dalam konstanta, berjalan secara paralel. Setelah setiap blok *output* dari kedua contoh ditambahkan ke *variabel chaining.* Hal ini tampaknya membuat algoritma sangat tahan terhadap pembacaan sandi [1]. Algoritma RIPEMD terdiri dari beberapa varian, di

Journal of Computer System and Informatics (JoSYC)
ISSN 2714-8912 (media online), ISSN 2714-7150 (media cetak)
Volume 3, No. 2, Februari 2022, Page 89−92
https://ejurnal.seminar-id.com/index.php/josyc
DOI 10.47065/josyc.v3i2.954

antaranya adalah RIPEMD-128 dan RIPEMD-160. Algoritma RIPEMD-160 adalah algoritma hash yang merupakan peningkatan dari RIPEMD-128. Peningkatan dan pengembangan ini dilakukan karena telah banyak usaha-usaha kriptanalisis yang telah dilakukan terhadap RIPEMD-128 dan berdasarkan informasi yang didapat maka diciptakanlah RIPEMD-160. RIPEMD-128, nilai *hash* yang dihasilkan hanya memiliki panjang 128 *bit.* Hanya terdapat sedikit perbedaan antara RIPEMD-160 dengan RIPEMD-128.

2. METODOLOGI PENELITIAN

2.1 Orisinalitas

Orisinalitas adalah keaslian atau ketulenan, orisinalitas dapat juga dikatakan sebagai seni mengingat apa yang kamu dengar, tetapi lupa dimana anda mendengarnya. integritas adalah mutu, sifat, atau keadaan yg menunjukkan kesatuan yg utuh sehingga memiliki potensi dan kemampuan yg memancarkan kewibawaan dan kejujuran [5].

2.2 Algoritma Rapid-MD 128

RIPE-MD dikembangkan untuk proyek RIPE Masyarakat Eropa. Algoritma ini adalah variasi dari algoritma MD4, yang dirancang untuk menahan serangan kriptanalisis yang ada dan menghasilkan nilai hash 128 bit. Rotasi dan urutan kata- kata pesan yang dimodifikasi. Selain itu, terdapat dua contoh dalam algoritma, berbeda dalam konstanta, berjalan secara paralel. Setelah setiap blok, Output dari kedua contoh ditambahkan ke variabel chaining. Hal ini tampaknya membuat algoritma sangat tahan terhadap pembacaan sandi [1].

Seperti semua fungsi hash varian MD4, RIPEMD-128 bekerja pada ukuran word 32- bit. Operasi primitifnya adalah [9] :

1. Rotasi bit ke kiri (left-spin)
2. Operasi boolean bitwise (AND, OR, NOT, XOR)
3. Operasi penjumlahan modulo 2^{32} terhadap word dalam *mode two's complement*.

RIPEMD 128 mengkompresi string input dengan panjang sembarang dengan membagi string tersebut menjadi blok-blok yang masing-masing sebesar 512 bit. Setiap blok dibagi menjadi 16 *string* dengan panjang masing-masing subblok adalah 4 *byte*. Kemudian masing-masing *string* 4 *byte* tersebut diubah ke *word* 32-bit dengan metode little-endian. Metode *little-endian* ini digunakan karena metode ini terimplementasikan secara *hardware* ke dalam *prosesor Intel* 80x86 yang umum digunakan. *String* yang menjadi input fungsi ini memiliki panjang sembarang. Jika ternyata panjang Nilai hash hasil proses RIPEMD-128 disimpan dalam 5 *word* 32 bit. Pembentukan kelimanya membentuk struktur umum dari algoritma ini. Isi akhir dari kelima word ini akan digabung menjadi string 128 bit, dengan metode little-endian. Pada awalnya kelima word 32-bit ini diinisialisasi dengan suatu nilai tertentu (*initial vector*). Nilai-nilai tersebut adalah sebagai berikut (dalam heksadesimal).

3. HASIL DAN PEMBAHASAN

Umumnya kerusakan dan perubahan pada *file* terjadi karena adanya perubahan isi dari *file* tersebut sehingga tidak dapat dibuka sama sekali bahkan *file* tersebut rusak mengakibatkan *cluster-cluster* pada *file* khususnya *file* dokumen berubah atau tidak sesuai dengan struktur awalnya. *Cluster-cluster* ini merupakan informasi penting yang dibutuhkan sistem untuk dapat mengenali identitas *file* dokumen serta isi dari *file* dokumen tersebut. Dalam menjaga dan mendeteksi kerusakan *file* dokumen, dapat dibuat sebuah aplikasi yang dapat mendeteksi kerusakan serta integritas data, digunakan suatu cara untuk menghitung suatu nilai terhadap data yang diberikan dan nilai tersebut dikirim bersama-sama data untuk dicek oleh penerima apakah data yang diterima sama dengan aslinya. Untuk mengatasi permasalahan dalam mengotentifikasi *file* dokumen digunakan metode CRC 32. CRC 32 memiliki beberapa varian bergantung pada bilangan *polynomial* yang digunakan dalam proses komputasinya.

3.1 Penerapan Algoritma Rapid-MD 128

Pada algoritma MD-2 dengan melakukan pengujian terhadap objek citra digital, bila dikonversikan ke dalam bit seperti berikut : M = "abc"

"s" : 01100001
'a" : 01100010
"m" : 01100011, maka M : 01100001 01100010 01100010

Dari keterangan diatas diketahui bahwa panjang M = 24 bit, jika dilihat di persamaan 2, panjang pesan yang disimbolkan dengan *l*. Proses berikutnya adalah melakukan *padding* dengan cara menambahkan bit '1' dan sisanya adalah bit '0' sejumlah k. Dimana k merupakan hasil dari persamaan berikut :

$k = l + 1 = 448 \bmod 512$
$k = 24 + 1 = 448 \bmod 512$
$k = 25 = 448 \bmod 512$
$k = 448\text{-}25 = 423$

Journal of Computer System and Informatics (JoSYC)
ISSN 2714-8912 (media online), ISSN 2714-7150 (media cetak)
Volume 3, No. 2, Februari 2022, Page 89−92
https://ejurnal.seminar-id.com/index.php/josyc
DOI 10.47065/josyc.v3i2.954

Maka banyak bit '0' yang ditambahkan sejumlah 423 bit. Setelah itu tambahkan jumlah panjang pada akhir pesan yang di-*padding*. *l* = 24 = 00011000. Sehingga didapati hasil pesan yang *dipadding* :

01100001 01100010 01100011 1 00…0 00…011000

Kemudian melakukan *parsing*, namun dalam contoh kasus ini karena panjang pesan yang di-*padding* tidak melebihi 512 bit, maka hanya menghasilkan 1 blok 512 bit yaitu $M^{(0)}$. Tahap selanjutnya adalah membagi setiap blok 512 bit menjadi 16 blok yang terdiri dari 32 bit, $M_0^{(i)}$, $M_1^{(i)}$, …, $M_{15}^{(i)}$.

$M_0^{(0)}$: 0110 0001 0110 0010 0110 0011 1000 0000
$M_1^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
$M_2^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
$M_3^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
$M_4^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
$M_5^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
$M_6^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
$M_7^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
$M_8^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
$M_9^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
$M_{10}^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
$M_{11}^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
$M_{12}^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
$M_{13}^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
$M_{14}^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
$M_{15}^{(0)}$: 0000 0000 0000 0000 0000 0000 0001 1000

Setelah proses parsing, maka berikutnya melakukan inisial hash value sebagai berikut :

$H_0^{(0)}$ = 6A09E667
$H_1^{(0)}$ = BB67AE85
$H_2^{(0)}$ = 3C6EF372
$H_3^{(0)}$ = A54FF53A
$H_4^{(0)}$ = 510E527F
$H_5^{(0)}$ = 9B056887
$H_6^{(0)}$ = 1F83D9AB
$H_7^{(0)}$ = 5BE0CD19

Kemudian melakukan proses persiapan *message schedule*, tahap ini diawali dengan mengkonveri setiap blok menjadi bilangan hexadesimal seperti berikut ini :

$M_0^{(0)}$: 61626380
$M_1^{(0)}$: 00000000
$M_2^{(0)}$: 00000000
$M_3^{(0)}$: 00000000
$M_4^{(0)}$: 00000000
$M_5^{(0)}$: 00000000
$M_6^{(0)}$: 00000000
$M_7^{(0)}$: 00000000
$M_8^{(0)}$: 00000000
$M_9^{(0)}$: 00000000
$M_{10}^{(0)}$: 00000000
$M_{11}^{(0)}$: 00000000
$M_{12}^{(0)}$: 00000000
$M_{13}^{(0)}$: 00000000
$M_{14}^{(0)}$: 00000000
$M_{15}^{(0)}$: 00000018

Nilai yang dimiliki $M_i^{(j)}$ menjadi nilai $W_i^{(j)}$, sehingga $W_0^{(0)} = M_0^{(0)}$ dan seterusnya hingga $W_{63}^{(i-1)}$, dimana i adalah jumlah blok 512 bit. Selanjutnya melakukan inisialisasi variabel a,b,c,…,h, dimana nilai dari setiap variabel diambil dari *initial hash value* sehingga a = $H_0^{(0)}$, b = $H_1^{(0)}$, c = $H_2^{(0)}$, …., h = $H_7^{(0)}$, selanjutnya dilakukan proses komputasi fungsi dari r = 0 sampai r = 63.

a b c d e f G h

Init: 6A09E667 BB67AE85 3C6EF372 A54FF53A 510E527F 9B05688C 1F83D9AB 5BE0CD19
t = 0 5D6AEBCD 6A09E667 BB67AE85 3C6EF372 FA2A4622 510E527F 9B05688C 1F83D9AB
t = 1 5A6AD9AD 5D6AEBCD 6A09E667 BB67AE85 78CE7989 FA2A4622 510E527F 9B05688C
t = 2 C8C347A7 5A6AD9AD 5D6AEBCD 6A09E667 F92939EB 78CE7989 FA2A4622 510E527F
…. …… …… …….. …….. …… ……. …… ……

Journal of Computer System and Informatics (JoSYC)
ISSN 2714-8912 (media online), ISSN 2714-7150 (media cetak)
Volume 3, No. 2, Februari 2022, Page 89−92
https://ejurnal.seminar-id.com/index.php/josyc
DOI 10.47065/josyc.v3i2.954

t = 63 506E3058 D39A2165 04D24D6C B85E2CE9 5EF50F24 FB121210 948D25B6 961F4894

Maka tahap berikutnya adalah menjumlahkan variabel a, b, c, d, e, f, g, h yang sudah dikomputasi dengan *initial hash value*

$H_0^{(0)}$ = 506e3058 + 6a09e667 = ba7816bf
$H_1^{(0)}$ = d39a2165 + bb67ae85 = bf01cfea
$H_2^{(0)}$ = 04d24d6c + 3c6ef372 = 414140de
$H_3^{(0)}$ = b85e2ce9 + a54ff53a = 5dae2223
$H_4^{(0)}$ = 5ef50f24 + 510e527f = b00361a3
$H_5^{(0)}$ = fb121210 + 9b056887 = 96177a9c
$H_6^{(0)}$ = 948d25b6 + 1f83d9ab = b410ff61
$H_7^{(0)}$ = 961f4894 + 5be0cd19 = f20015ad, terakhir gabungkan $H_0 – H_7$

ba7816bf bf01cfea 414140de 5dae2223 b00361a3 96177a9c b410ff61 f20015ad, Sehingga didapat *hash value* atau *message digest* dari M adalah : ba7816bf bf01cfea 414140de 5dae2223 b00361a3 96177a9c b410ff61 f20015ad

4. KESIMPULAN

Berdasarkan hasil pengujian aplikasi pemeriksa keaslian file dokumen maka dapat disimpulkan bahwa Metode riped-md 128 dapat juga digunakan untuk mendeteksi isi *file* dokumen walaupun nama *file* nya berbeda. Aplikasi ini dapat meminimalisir pengguna (*user*) terinfeksi *malware* dari *file* yang diterima karena telah dimodifikasi.

REFERENCES

[1] AR.Abdurrasyid, 2017, "*Perbandingan Algoritma Gosudarstvennyi Standard (GOSH) dan Ripe Message Digest (RIPE-MD) Pada Hashing Kode Booking Tiket Pesawat",* Jurnal Sistem Informasi Kaputama, Vol. 1, Nomor 1, ISSN : 2548-9712.
[2] http://Berbag_iilmu.com/2014/06/pengertian-perancangan.html, diakses tanggal 27 Juni 2019.
[3] http://id.wikipedia.org.wiki. *Aplikasi*. Diakses tanggal 27 Juni 2019
[4] https://www.psychologymania.com/2013/04/pengertian-deteksi-dini.html, diakses tanggal 27 Juni 2019
[5] https://brainly.co.id/tugas/844932, diakses tanggal 27 Juni 2019.
[6] Basuki, Sulistyo, 1996, "*Kerjasama dan Jaringan Perpustakaan",* Penerbit Universitas Terbuka, Jakarta.
[7] Elva Rahmah, 2013, *"Dokumentasi dan kearsipan",* Artikel: Program Studi Ilmu Informasi Perpustakaan dan Kearsipan. FBS Universitas Negeri Padang.
[8] H.M. Nawawi, Sibali, 2010, "*Penerapan Sistem Kearsipan Pada Kantor Arsip Daerah Kabupaten Kutai Barat*", Jurnal Eksis Vol.6 No.2, Agustus 2010: 1440 – 1605.
[9] Roland, L, 2008, Makalah Algoritma Riped-MD, Program Studi Teknik Informatika, Institut Teknologi Bandung, Jalan Ganesha 10, Bandung.
[10] A.S. Rosa dan Shalahuddin. M, 2013, *"Rekayasa Perangkat Lunak Terstruktur",* Andi, Yogyakarta.
[11] Aditya, Arif Primananda, 2013, *"Dasar-Dasar Pemrograman Database Dekstop Dengan Visual Basic.Net 2008",* PT. Elex Media Komputindo, Jakarta.