Journal of Informatics Management and Information Technology
ISSN 2774-4744 (Media Online)
Vol 2, No 2, April 2022
Hal 48−53
https://hostjournals.com/jimat

Rancang Bangun Aplikasi Duplicate Docx Scanner Menerapkan Metode SHA-256

Adelina

Ilmu Komputer dan Teknologi Informasi, Teknik Informatika, Universitas Budi Darma, Medan, Indonesia
Email: adelinasiburian6@gmail.com

Abstrak−*Harddisk* merupakan salah satu komponen inti dari sebuah komputer dalam berbagai jenis dan ukuran. *Harddisk* dengan ukuran besar bisa menjadi penuh, meskipun merasa tidak menyimpan *file* yang berukuran besar. Kemungkinan terjadi tersimpannya *file* yang sama pada *directory* sangat sulit dilakukan dan akan memakan waktu yang lama. Aplikasi *Duplicate Docx Scanner* mampu menyelesaikan masalah tersebut. Aplikasi *Duplicate Docx Scanner* mampu mengetahui *file* yang sama terletak di *directory* yang berbeda dengan menentukan *directory* yang terdapat pada *harddisk* dan media penyimpanan lain. Aplikasi ini dapat menghitung nilai *hash* sehingga dapat menemukan *file* yang sama. Aplikasi *Duplicate Docx Scanner* menggunakan metode SHA-256 *hash.* SHA-256 digunakan untuk melakukan pemeriksaan integritas *file* dalam berbagai situasi. Aplikasi *Duplicate Docx Scanner* ini dirancang menggunakan bahasa pemograman *Microsoft Visual Basic.Net* 2008.

Kata Kunci: File; Duplikat File; Pencarian File; SHA-256 Hash

Abstract−Hard drives are one of the core components of a computer in various types and sizes. Large hard disks can become full, even though they don't feel like storing large files. The possibility of saving the same file in the directory is very difficult and will take a long time. The Duplicate Docx Scanner application is able to solve this problem. The Duplicate Docx Scanner application is able to find out the same file is located in a different directory by determining the directory on the hard disk and other storage media. This application can calculate the hash value so that it can find the same file. The Duplicate Docx Scanner application uses the SHA-256 hash method. SHA-256 is used to perform file integrity checks in a variety of situations. This Duplicate Docx Scanner application is designed to use the Microsoft Visual Basic.Net 2008 programming language.

Keywords: File; File Duplicate; File Search; SHA-256 Hash

1. PENDAHULUAN

Teknologi informasi dan ilmu pengetahuan di dunia saat ini berkembang sangat pesat. Perkembangan teknologi tersebut berpengaruh hampir di semua aspek kehidupan manusia, salah satunya adalah penyimpanan data. Penyimpanan pada zaman dulu hanya dilakukan di tempat-tempat yang dirasa aman.

Penyimpanan data pada zaman dahulu disimpan pada batu bertulis, selanjutnya berkembang penyimpanan data dilakukan pada kertas, semakin lama zaman semakin berkembang dengan adanya komputer dan media-media penyimpanan yang sekarang ini lebih efisien, efektif, dan canggih.

Harddisk adalah sebuah media penyimpanan sekunder pada sebuah komputer. *Harddisk* digunakan untuk menyimpan data, dokumen dan file-file lain didalam sebuah personal komputer. Cakram keras (bahasa inggris: *harddisk* atau *harddisk drive* disingkat HDD atau *hard drive* disingkat HD) adalah sebuah komponen perangkat keras yang menyimpan data sekunder dan berisi piringan magnetis. Sebagai penyimpanan data , *harddisk* merupakan salah satu komponen inti dari sebuah komputer dalam berbagai jenis dan ukuran. Karena kebutuhan penyimpanan data yang semakin meningkat dari masa kemasa hingga saat ini para produsen berlomba membuat *harddisk* yang berukuran besar. Saat ini, berbagai macam pabrik *harddisk* sudah mengeluarkan berbagai macam ukuran *harddisk*, dari 40 GB sampai 1 TB (*Tera Byte*) atau setara dengan 1000 Giga. Menggunakan *harddisk* berukuran besar memang menyenangkan. *Harddisk* dengan ukuran besar itu pun masih akan menjadi penuh, meskipun kita merasa tidak menyimpan file yang berukuran besar.

Kemungkinan terjadi tersimpannya file yang sama pada *directory* yang berbeda sangat besar, mencari file yang sama tersebut pada masing-masing *directory* sangat sulit dilakukan dan akan memakan waktu yang lama. Aplikasi ini mampu menyelesaikan masalah tersebut. Aplikasi ini mampu mengetahui file yang sama yang terletak di *directory* yang berbeda dengan menentukan *directory* yang terdapat pada *harddisk.* Aplikasi ini menggunakan metode SHA-256 hash. SHA-256 (*Secure Hash Algorithm256*) digunakan untuk melakukan pemeriksaan integritas file dalam berbagai situasi. SHA-256 merupakan salah satu fungsi *hash* satu arah, karena tidak mungkin menemukan pesan dari *message digest* yang dihasilkan [1]. Algoritma SHA-256 menghitung nilai *message digest* dari sebuah pesan, dimana pesan tersebut memiliki panjang maksimum 2^{64} bit. Algoritma ini menggunakan sebuah *message schedule* yang terdiri dari 64 element 32-bit *word*, delapan buah variabel 32-bit, dan variabel penyimpan nilai *hash* 8 buah *word* 32-bit dan menghasilkan *message digest* yang panjangnya 256-bit.

2. METODOLOGI PENELITIAN

2.1 Rancang Bangun

Perancangan adalah kegiatan yang memiliki tujuan untuk mendesain sistem baru yang dapat menyelesaikan masalah-masalah yang dihadapai perusahaan yang di peroleh dari pemilihan alternatif sistem yang terbaik [1]. Rancang bangun (desain) adalah tahap dari setelah analisis dari siklus pengembangan sistem yang merupakan pendefenisian dari kebutuhan-kebutuhan fungsional, serta menggambarkan bagaimana suatu sistem dibentuk yang dapat berupa

Journal of Informatics Management and Information Technology
ISSN 2774-4744 (Media Online)
Vol 2, No 2, April 2022
Hal 48−53
https://hostjournals.com/jimat

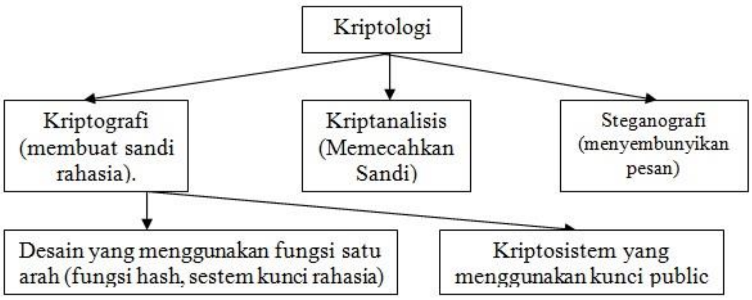
penggambaran, perencanaan dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah ke dalam satu kesatuan yang utuh dan berfungsi, termasuk menyangkut mengkonfigurasikan dari komponen-komponen perangkat keras dan perangkat lunak dari suatu sistem [2].

2.2 Aplikasi

Pengertian aplikasi menurut [3] adalah penggunaan dalam suatu komputer, instruksi (*instruction*) atau pernyataan (*statement*) yang disusun sedemikian rupa sehingga komputer dapat memproses *input* menjadi *output.* Aplikasi merupakan rangkaian kegiatan atau perintah untuk dieksekusi pleh komputer. Program merupakan kumpulan *instruction set* yang akan dijalankan ole pemroses, yaitu berupa *software.* Program berisi konstruksi logika yang dibuat oleh manusia, dan sudah diterjemahkan ke dalam bahasa mesin sesuai format yang ada pada *instruction set.* Program aplikasi merupakan program siap pakai. Program yang direkam untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain. Contoh-contoh aplikasi ialah program pemproses kata dan Web Browser.

2.3 Kriptografi

Kriptografi (*cryptography*) berasal dari bahasa Yunani, yaitu dari kata *crypto* dan *graphia* yang berarti ‘penulisan rahasia’. Kriptografi adalah ilmu ataupun seni yang mempelajari bagaimana membuat suatu pesan yang dikirim oleh pengirim dapat disampaikan kepada penerima dengan aman. Kriptografi merupakan bagian dari suatu cabang ilmu matematika yang disebut kriptologi (*cryptology*). Kriptografi bertujuan menjaga kerahasiaan informasi yang terkandung dalam data sehingga informasi tersebut tidak dapat diketahui oleh pihak yang tidak sah[4]. Kriptografi pada awalnya dijabarkan sebagai ilmu yang mempelajari bagaimana menyembunyikan pesan. Namun pada pengertian modern Kriptografi adalah ilmu yang bersandarkan pada teknik matematika untuk berurusan dengan keamanan informasi seperti kerahasiaan, keutuhan data dan otentikasi entitas[5].



Gambar 1. Area bidang kriptologi

2.4 Fungsi *Hash*

Fungsi *hash* sering disebut dengan fungsi *hash* satu arah *(one-way function), message digest*, *fingerprint*, fungsi kompresi dan *message authentication code* (MAC), merupakan suatu fungsi matematika yang mengambil masukan panjang variabel dan mengubahnya ke dalam urutan biner dengan panjang yang tetap. Fungsi *hash* biasanya diperlukan bila ingin membuat sidik jari dari suatu pesan. Sidik jari pada pesan merupakan suatu tanda bahwa pesan tersebut benar-benar berasal dari orang yang diinginkan. Tentang hal ini akan dibahas lebih lanjut pada bagian berikutnya[6]. Fungsi *hash (hash function)* merupakan salah satu teknik kriptografi untuk menghitung nilai unik dari sebuah data. Fungsi *hash* dapat diibaratkan sebagai sidik jari elektronik *(digital fingerprint)* dari informasi elektronik. Sidik jari elektronik berguna untuk menentukan orisinalitas sebuah dokumen elektronik. Dua dokumen elektronik yang berbeda akan memiliki nilai *hash* yang berbeda, itulah sebabnya apabila sebuah dokumen telah mengalami perubahan, maka nilai *hash* juga akan berubah[7]. Fungsi *hash* adalah sebuah fungsi yang memasukannya adalah sebuah pesan dan keluaran sebuah sidik pesan *(message fingerprint)*. Sidik pesan sering juga disebut *message digest*, fungsi *hash* dapat digunakan untuk mewujudkan beberapa layanan keamanan jaringan misalnya untuk keutuhan data dan otentikasi pesan[5].

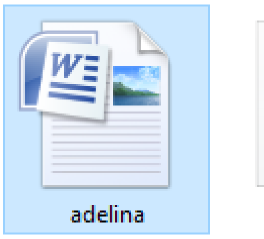
3. HASIL DAN PEMBAHASAN

Hash yang dihasilkan adalah kunci dalam menentukan apakah terdapat file yang sama. Salah satu solusi penyelesaian melakukan proses pencarian file yang sama pada direktori yaitu menggunakan aplikasi *duplicate docx* menerapkan metode SHA-256 yang mana metode digunakan untuk melakukan pencarian nilai hash dari sebuah file tersebut.

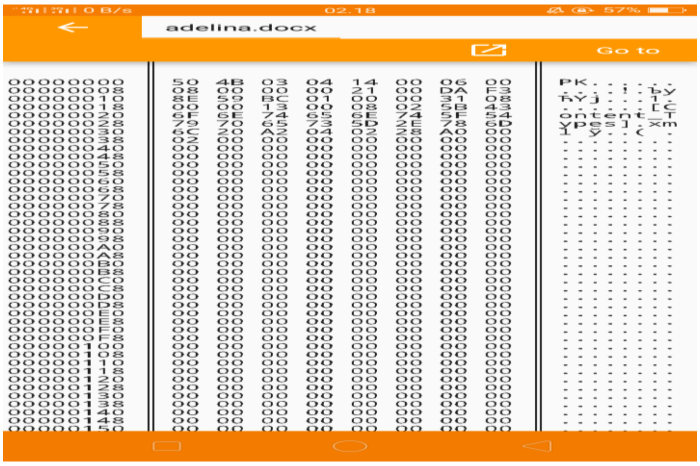
3.1 Analisa Algoritma SHA-256

Contoh kasus Asli dalam proses ini yaitu *file docx*. Data yang diambil sebanyak 32 *byte*, cara pengambilan nilai *hexadecimal file docx* menggunakan aplikasi *Binary Viewer*, seperti dibawah ini.

ISSN 2774-4744 (Media Online)
Vol 2, No 2, April 2022
Hal 48-53
https://hostjournals.com/jimat



Gambar 2. *File* docx



Gambar 3. *Binary Viewer File Docx*

Sebelum menerapkan metode SHA-256 terlebih dahulu nilai *hexa* dari *file docx* diubah menjadi bilangan biner. Untuk bilangan binernya bisa dilihat pada tabel berikut:

Tabel 1. Nilai Biner

01010000	01001011	00000011	00000100	00010100	00000000	00000110	00000000
00001000	00000000	00000000	00000000	00100001	00000000	11011010	11110011
10001110	01011001	10111100	00000001	00000000	00000000	00110001	00001000
00000000	00000000	00010011	00000000	00001000	00000010	01011011	01000011

1. *Padding* Pesan
   Diketahui panjang M = 256. *Padding* dilakukan dengan cara menambahkan bit ‘1’ dan sisanya adalah bit ‘0’ sejumlah k. Untuk nilai k bisa didapatkan melalui persamaan berikut :
   $k = l + 1 \equiv 448 \bmod 512$
   $k = 256 + 1 \equiv 448 \bmod 512$
   $k = 257 \equiv 448 \bmod 512$
   $k = 448 - 257$
   $k = 191$
   Maka banyaknya bit 0 yang ditambahkan sejumlah 191 bit.

Tabel 2. Hasil *Padding* Pesan

01010000	01001011	00000011	00000100	00010100	00000000	00000110	00000000
00001000	00000000	00000000	00000000	00100001	00000000	11011010	11110011
10001110	01011001	10111100	00000001	00000000	00000000	00110001	00001000
00000000	00000000	00010011	00000000	00001000	00000010	01011011	01000011
10000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000

2. Panjang *Append*
   Panjang *append* dilakukan dengan penambahan panjang pesan 64 bit di akhir. Panjang pesan sebanyak 256 bit sehingga penambahan panjang *append* adalah sebagai berikut :

Journal of Informatics Management and Information Technology
ISSN 2774-4744 (Media Online)
Vol 2, No 2, April 2022
Hal 48−53
https://hostjournals.com/jimat

Tabel 3. Penambahan Panjang Append

01010000	01001011	00000011	00000100	00010100	00000000	00000110	00000000
00001000	00000000	00000000	00000000	00100001	00000000	11011010	11110011
10001110	01011001	10111100	00000001	00000000	00000000	00110001	00001000
00000000	00000000	00010011	00000000	00001000	00000010	01011011	01000011
10000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000	00000000	00000000	00000000	10000000

3. *Parsing* Pesan
   Pada kasus ini panjang pesan tidak lebih dari 512 sehingga hanya menghasilkan 1 blok 512 bit yaitu $M^{(0)}$. Tahap selanjutnya adalah melakukan *parsing* pesan dengan membagi setiap blok 512 bit menjadi 16 blok berukuran 32 bit.

Tabel 4. *Parsing* pesan

$M_0^{(0)}$	:	01010000010010110000001100000100	$M_1^{(0)}$	:	00010100000000000000011000000000
$M_2^{(0)}$	:	00001000000000000000000000000000	$M_3^{(0)}$	:	00100001000000001101101011110011
$M_4^{(0)}$	:	10001110010110011011110000000001	$M_5^{(0)}$	:	00000000000000000011000000001000
$M_6^{(0)}$	:	00000000000000000001001100000000	$M_7^{(0)}$	:	00001000000000100101101101000011
$M_8^{(0)}$	:	10000000000000000000000000000000	$M_9^{(0)}$	:	00000000000000000000000000000000
$M_{10}^{(0)}$	:	00000000000000000000000000000000	$M_{11}^{(0)}$	:	00000000000000000000000000000000
$M_{12}^{(0)}$	:	00000000000000000000000000000000	$M_{13}^{(0)}$	:	00000000000000000000000000000000
$M_{14}^{(0)}$	:	00000000000000000000000000000000	$M_{15}^{(0)}$	:	00000000000000000000000010000000

4. Inisialisasi Nilai *Hash*
   Initial hash value untuk SHA-256 adalah sebagai berikut :
   H_0 = 6a09e667
   H_1 = bb67ae85
   H_2 = 3c6ef372
   H_3 = a54ff53a
   H_4 = 510e527f
   H_5 = 9b05688c
   H_6 = 1f83d9ab
   H_7 = 5be0cd19

5. Penjadwalan Pesan
   Penjadwalan pesan diawali dengan mengubah setiap blok pesan menjadi bilangan heksadesimal dengan ketentuan sebagai berikut :

$$W_t = \begin{cases} M_t^{(t)} & 0 \le t \le 15 \\ \sigma_1^{(256)}(W_{i-2}) + W_{i-7} + \sigma_0^{(256)}(W_{i-15}) + W_{i-16}, & 16 \le t \le 63 \end{cases}$$

Tabel 5. Penjadwalan Pesan

4F606F69	6962646E	6D79736B	5E586A61	707A7260	69666169	4C65606F	4C656F
60567374	80000000	00000000	00000000	00000000	00000000	00000000	00000120
18458186	F1C56C43	B25F3BC3	CC660898	32D405C0	CE182B88	79CDBB87	ED495557
CDC92068	B299951B	807015AD	2AD60F3A	D65B97BB	403EA139	E6B05C80	C1CE665A
B394B55B	D1EB82CF	5391D3AD	DB1B2C81	49879567	17DA8AA1	5ACF9149	3517EFA5
1EC0C640	5EBC651C	CA3CE603	BB20E07F	B0F32F26	116867D3	4250DF45	ABEA303E
EBD90AF2	1E99300C	86C5B810	6D08B661	A89881C5	25D17EA6	0ECBA874	63F53B98
08BF0E68	477CF579	6E4E79F6	D871E636	889E2FAB	A53FA658	5C6DE651	96877234

6. penjadwalan pesan dari 16 - 63 dilakukan perhitungan sebagai berikut :

$\sigma_1^{(256)}(W_{i-2})$ = $((W_{i-2})\text{ROTR } 17) \oplus ((W_{i-2})\text{ROTR } 19) \oplus ((W_{i-2})\text{SHR}10)$
$((W_{16-2})\text{ROTR } 17)$ = $((W_{14})\text{ROTR } 17)$ = ((00000000) ROTR 17)
= ((00000000 00000000 00000000 00000000) ROTR 17)
= (00000000 00000000 00000000 00000000)
$((W_{16-2})\text{ROTR } 19)$ = $((W_{14})\text{ROTR } 19)$ = ((00000000) ROTR 19)

= ((00000000 00000000 00000000 00000000) ROTR 19)
= (00000000 00000000 00000000 00000000)

$((W_{16-2})SHR\ 10)$ = $((W_{14})SHR\ 10)$ = ((00000000) SHR 10)
= ((00000000 00000000 00000000 00000000) SHR 10)
= (00000000 00000000 00000000 00000000)

$\sigma_1^{(256)}(W_{i-2})$ = (00000000 00000000 00000000 00000000)
= (00000000 00000000 00000000 00000000)
= (00000000 00000000 00000000 00000000) $\oplus$
(00000000 00000000 00000000 00000000) = 00000000

W_{16-7} = W_9 = 80000000

$\sigma_0^{(256)}(W_{i-15})$ = $((W_{i-15})ROTR\ 7) \oplus ((W_{i-15})ROTR\ 18) \oplus ((W_{i-2})SHR3)$

$((W_{16-15})ROTR\ 7)$ = $((W_1)ROTR\ 7)$ = ((6962646E) ROTR 7)
= (01101001 01100010 01100100 01101110) ROTR 7)
= (11011100 11010010 11000100 11001000)

$((W_{16-15})ROTR\ 18\ )$ = $((W_1)ROTR\ 18)$ = ((6962646E) ROTR 18)
= (01101001 01100010 01100100 01101110) ROTR 18)
= (10011001 00011011 10011010 01011000)

$((W_{16-15})SHR\ 3\ )$ = $((W_1)SHR\ 3)$ = ((6962646E) SHR 3)
= (01101001 01100010 01100100 01101110) ROTR 18)
= (00001101 00101100 01001100 10001101)

$\sigma_0^{(256)}(W_{i-15})$ = (00001101 00101100 01001100 10001101)
= (11011100 11010010 11000100 11001000)
= (10011001 00011011 10011010 01011000) $\oplus$
(01001000 11100101 00010010 00011101)
= 48E5121D

W_{16-16} = W_0 = 4F606769

W_t = $\sigma_1^{(256)}(W_{i-2})+W_{i-7}+\sigma_0^{(256)}(W_{i-15})+W_{i-16}$

W_t = 00000000+80000000+48E5121D+4F606769 = 1 18458186

Demikian seterusnya hingga $W_{63}^{(i-1)}$, di mana i adalah jumlah blok 512 bit.

7. Inisialisasi variable kerja
   Seterusnya lakukan inisialisasi variable kerja a,b,c,d,e,f,g dan h di mana setiap variable diambil dari initial hash value $a = H_0^{(0)}$, $b = H_1^{(0)}$, $c = H_2^{(0)}$, $d = H_3^{(0)}$, $e = H_4^{(0)}$, $f = H_5^{(0)}$, $g = H_6^{(0)}$, $h = H_7^{(0)}$. Seterusnya dilakukan proses komputasi fungsi hash SHA-256 dari t = 0 sampai t = 63.

8. Menjumlahkan variable kerja dengan inisial nilai hash

Tabel 6. Penjumlahan dengan *initial hash value*

Variabel	Initial Hash Value		Variabel Kerja	Hasil
$H_0^{(0)}$	6A09E667	+	7D36E2C0	E740C927
$H_1^{(0)}$	BB67EA85	+	8FC52F27	1 4B2D19AC
$H_2^{(0)}$	3C6EF372	+	EE342801	1 2AA31B73
$H_3^{(0)}$	A54FF53A	+	AE321AE6	1 53821020
$H_4^{(0)}$	510E527F	+	2BA3B2A4	7CB20523
$H_5^{(0)}$	9B05688C	+	F005EA8C	1 8B0B5318
$H_6^{(0)}$	1F83D9AB	+	5A93AEA0	7A17884B
$H_7^{(0)}$	5BE0CD19	+	F38852FA	1 4F692013

9. Output
   Output dari SHA-256 sebagai berikut :
   E740C927 || 4B2D19AC || 2AA31B73 || 53821020 || 7CB20523 || 8B0B5318 || 7A17884B || 4F692013
   Sehingga didapat nilai *hash* dari pesan M adalah sebagai berikut :
   E740C9274B2D19AC2AA31B73538210207CB205238B0B53187A17884B4F692013. *Hash* inilah yang dijadikan acuan dalam mencari file duplikat. *Hash* dari file yang didapat dari perhitungan sebelumnya, kemudian dilakukan pencocokan *hash* tersebut dengan *hash-hash* lainnya. Jika terdapat *hash* yang sama persis, maka dapat dinyatakan bahwa file-file yang memiliki *hash* tersebut adalah sama. File-file yang sama kemudian dimasukkan kedalam list file *docx* duplikat.

4. KESIMPULAN

Adapun kesimpulan yang diperoleh dalam penelitian perancangan *Duplicate Docx Scanner* bertujuan untuk memudahkan orang-orang mencari *file docx* yang sama pada *directory* berbeda dengan menggunakan metode SHA-256. Aplikasi yang dirancang mampu melakukan pencarian *file docx* yang sama dan *file* yang ditemukan menunjukkan bahwa *file* tersebut sama. Aplikasi yang dirancang mampu menentukan *file* yang sama pada salah satu direktori yang ada pada harddisk. kriteria *file docx* yang sama ditentukan oleh nilai SHA-256 pada sebuah *file.* Meskipun nama *file* sama tidak mempengaruhi kesamaan sebuah *file.*

REFERENCES

[1] H. Sembiring, S. Utara, F. Y. Manik, S. Utara, and S. Utara, "*Penerapan Algoritma Secure Hash Algorithm (SHA) Keamanan Pada Citra*," vol. 4, no. 1, pp. 33–36, 2019.
[1] Al-Bahra Bin Ladjamudin, *Analisis dan desain sistem informasi*. Yogyakarta, 2005.
[2] jogiyanto, *Analisis & Desain Sistem Informasi*. Yogyakarta, 2013.
[3] H. M. jogiyanto, *Analisis dan Rancangan Sistem Informasi*. Yogyakarta, 2010.
[4] M. K. Emy Setyaningsih, S.Si., *Kriptografi & Implementasinya menggunakan MATLAB*. Yogyakarta: ANDI, 2015.
[5] Rifki Sadikin, *Kriptografi untuk keamanan jaringan dan implementasi dalam Bahasa Java*. Yogyakarta: ANDI, 2012.
[6] Dony Ariyus, *PENGANTAR ILMU KRIPTOGRAFI Teori Analisis & Implementasi*. Yogyakarta, 2008.
[7] Dimaz A Wijaya, *MENGENAL BITCOIN & CRYPTOCURRENCY*. Medan, 2016.
[8] https://ejurnal.stmik-budidarma.ac.id/index.php/ijics/article/view/1404/1186
[9] https://journal.ugm.ac.id/ijccs/article/view/46401/26040.