Implementasi Metode Keccak Untuk Mendeteksi Orisinalitas Citra Digital

Sarma Uli Aprila Batubara¹

¹Fakultas Ilmu Komputer dan Teknologi Informasi, Program Studi Teknik Informatika, Universitas Budi Darma, Medan, Indonesia

Email: ¹sarmauliaprilabatubara75@gmail.com

Abstrak

Perancangan aplikasi untuk otentikasi scan ijazah yang digunakan berjenis citra digital yang akan diberikan nilai biner Keccak. Citra digital merupakan gambar yang harus ditampilkan pada suatu layar komputer sebagai himpunan atau diskrit nilai yang disebut pixel. Scan ijazah pada citra digital sangatlah penting untuk melihat keaslian ijazah tersebut, dengan demikian citra ijazah yang telah diberikan nilai biner Keccak. Pada saat menjalani perubahan oleh orang yang tidak bertanggung jawab dapat diketahui dengan cepat. Jadi dirancanglah aplikasi untuk mengontenkitasi scan ijazah dengan menerapkan metode Keccak. Hasil penelitian menunjukkan bahwa algoritma Keccak dapat di implementasikan dengan baik, sehingga perancangan aplikasi scan ijazah dapat dijamin bagus dengan melakukan encoding yang akan digunakan dalam citra ijazah tersebut.

Kata Kunci : Kriptografi, Keccak, Citra digital, Ijazah, Aplikasi

1. PENDAHULUAN

Pembuktian otentikasi citra digital dapat diterapkan melalui bidang kriptografi. Kriptografi memiliki peningkatan dalam berbagai jenis fungsii pengamanan data. Metode-metode yang digunakan dalam kriptografi juga dikebangkkan dengan pengetahuan matematika dan komputasi, sehingga fungsii pengamanan data atau berkas jadi semakin rumit.

Citra atau gambar dalam bahasa latin imago adalah suatu representasi, kemiripan, atau imitasi dari suatu obyek atau benda. Citragambar dapat didefinisikan sebagai sebuah fungsi dua dimensi f(x,y) dimana x dan y adalah koordinat bidang dataar dan harga f di setiap pasangan koordinat (x,y) disebut intensitas atau level keabuan (gray level) dari gambar dititik tersebut. Jika x,y dan f semuanya berharga (finite) dan nilainya diskrit, maka gambarannya disebut citra digital (gambar digital). Sebuah citra digital terdiri dari sejumlah elemen yang berhinga dimana masing-masing mempunyai lokasi dan nilai tertentu. Elemen-elemen ini disebut sebagai picture element, image element, pels atau pixels. Citra dapat dikelompokkan menjadi citra tampak dan citra tidak tampak. Contoh citra tampak dalam kehidupan sehari-hari: foto, gambar, dan lukisan, sedangkan citra tidak tampak misalnya: data gambar dalam file (citra digital), dan citra yang direpresentasikan menjadi fungsi matematik dan di antara jenis citra tersebut, hanya citra digital yang dapat diolah menggunakan komputer

Keccak merupakan salah satu algoritma fungsi hash yang dirancang oleh Guido Bertoni, Joan Daemen, Michæl Peeters, dan Gilles Van Assche. Keccak merupakan pemenang kompetisi SHA-3 Cryptographic Hash Algorithm Competition yang diselenggarakan oleh NIST dan telah dijaidkan standar untuk algoritma fungsi hash Secure Hash Algorithm (SHA-3) yang baru. Terpilihnya Keccak sebagai pemenang SHA-3 menjadikan Keccak sebagai algoritma fungsi hash standar yang telah terbukti keamannya. Perbedaan utama antara Keccak dengan SHA-1, SHA-2, ataupun MD5 adalah Keccak menggunakan konstruksi spons function pada pembentukan nilai hash-nya, sedangkan tiga algoritma tersebut menggunakan skema Merkle-Damgård

Salah satu metode aplikasi kriptografi berbasis komputer yang sering digunakan adalah Keccak dimana merupakan pengembangan dari pada teknologi terpadu yaitu komunikasi (interaktif) audio, video, penampilan citra (image) yang dikemas dengan sebutan teknologi multimedia. Dalam pembahasan ini Keccak dapat digunakan untuk menganalisis dan mempelajari konversi citra. Untuk mengatasi hal tersebut penulis ingin membuat sebuah aplikasi pembelajaran dan sekaligus membahas tentang konversi dimana nantinya dapat membantu mahasiswa memahami materi tentang bagaimana proses citra secara efektif dan efisien.

2. TEORITIS

A. Kriptografi

Cryptography (Kriptografi) berasal dari bahasa Yunani yaitu dari kata crypto yang berarti penulisan secret (rahasia), sedangkan graphein artinya (tulis). Jadi secara sederhana dapat diartikan secreet writing (tulisan rahasia). Definisi lain dari kriptografi adalah sebuah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan, integritas data serta otentikasi [1].

B. Citra Digital

piksel (pixel = picture element) atau elemen terkecil dari sebuah citra.

C. Metode Keccak

Keccak merupakan algoritma fungsi hash satu arah yang berbasis pada konstruksi spon dengan menggunakan fungsi permutasi keccak-f dengan rentang (panjang) permutasi b, ukuran setiap lane dimana:

\[ b = 25 \times x \]

Dengan

\[ 0 \leq 1 \leq 6 \]

Algoritma keccak memiliki prinsip yang sama dengan algoritma cipher block, dimana proses dilakukan terhadap blok-blok, setiap hasil proses bergantung dari masukan dan hasil proses sebelumnya, serta setiap proses dikenakan pada sebuah fungsi utama yang terdiri dari sejumlah round fungsi yang diiterasi beberapa kali. Namun terdapat perbedaan antara algoritma hash satu arah Keccak, dengan algoritma cipher block, yaitu:

1. Keccak tidak memiliki key-schedule.
2. Menggunakan konstanta round yang bersifat tetap.

Keccak menggunakan inner state selama proses hashing berlangsung. Dan fungsi spon yang digunakan terdiri dari padding, absorbing, dan squeezing. Setiap state memiliki panjang sesuai dengan panjang permutasi, yaitu b. Algoritma keccak nantinya akan diubah menjadi tiga parameter masukan, yaitu bitrate (r), capacity (c), dan diversity (d). Secara umum proses dari keccak ini adalah:

1. Preproses pesan masukan (P), yaitu menerapkan padding pada pesan masukan. Panjang pesan masukan hasil padding harus merupakan kelipatan r, dengan r = bitrate.
2. Pemecahan pesan masukan menjadi P0, P1, P2,...,Pi, dimana i = jumlah kelipatan panjang bitrate untuk panjang pesan masukan.
3. Absorbing pada semua pecahan pesan masukan.
4. Squeezing sebanyak j, dimana j = kelipatan panjang keluaran r/w untuk memenuhi panjang output yang diinginkan, r = bitrate dan w = panjang lane dari state. Dengan:

\[ w = 2l \]

5. Keluaran merupakan konkatenasi dari keluaran Squeezing pada rentang bitrate tertentu.

3. ANALISA

A. Analisa Masalah

Analisa masalah dalam mendeteksi citra ijazah penulis akan menguraikan bagaimana proses mendeteksian dengan menggunakan metode Keccak. Ijazah yang digunakan dengan tipe file JPG. File ijazah tersebut akan diberikan hasil enkripsi yang dilakukan dengan Keccak. Karena penelitian ini dilakukan untuk penanda alamat binary file yang digunakan sebagai identitas keaslian yang dijadikan sampel pada data penelitian ini.

Keccak merupakan suatu metode yang digunakan dalam mendeteksi kerusakan file dengan menggunakan nilai sebuah file dan dapat mendeteksi perubahan-perubahan yang terjadi pada file tersebut. Hasil yang diinginkan adalah file citra yang memiliki penanda atau identitas yang dapat diuji oleh aplikasi yang sudah tersedia. Sehingga analisa dilakukan untuk otentikasi hasil citra ijazah dengan menerapkan metode Keccak. Adapun ijazah yang digunakan berjenis citra digital yang akan diberikan nilai hash. Pixel citra ijazah diambil menggunakan aplikasi matlab dan dari pixel tersebut dilakukan encoding Keccak, sehingga file citra. Citra ijazah memiliki identitas yang berfungsi sebagai autentikasi citra ijazah. Dengan demikian citra ijazah yang telah diberikan nilai hash pada saat mengalami perubahan atau pun tidak.

B. Penerapan Metode Keccak

Pengerjaan kasus berikut yang pertama kali dilakukan adalah mengubah nilai RGB citra digital ijazah pada menjadi grayscale dan mendapatkan nilai pixelnya. Berikut ini adalah cara mendapatkan nilai pixel citra ijazah. Adapun langkah-langkah untuk mengambil nilai – nilai pixel citra ijazah dengan matlab adalah :

1. Buka aplikasi MATLAB.
2. Siapkan gambar grayscale bertipe JPEG dan simpan direktori.
3. Langkah selanjutnya tuliskan listing program.
4. Selanjutnya akan ditampilkan nilai pixel yang berisikan nilai berikut ini :

Gambar 1. RGB 5x5 pixel yang diambil menggunakan Matlab.

Berdasarkan pixel di atas diketahui nilai pixel yang terdapat pada citra tersebut diambil menggunakan aplikasi matlab. Nilai-nilai pixel grayscale ijazah nilai-nilai pixel yang akan diproses dengan menerapkan metode Keccak untuk mengetahui hasil orisinalitas terhadap citra ijazah itu sendiri. Demikian nilai pixel diatas diproses sesuai dengan ketentuan dari metode Keccak.

Langkah-langkah dalam menghitung nilai Keccak pada file adalah sebagai berikut :

Langkah 1 : Menambahkan Padding Bit

| 255 | 247 | 220 | 151 | 110 |
| 255 | 250 | 193 | 87 | 43 |
| 275 | 255 | 199 | 78 | 49 |
| 243 | 255 | 200 | 88 | 86 |
| 255 | 255 | 191 | 83 | 100 |

Gambar 2. Nilai Padding Bit
Setelah nilai pixel didapat selanjutnya menghitung panjang dan dimoduskan seperti dibawah ini:
M = 5 x 5 pixel
= 25 pixel
= 25x8
= 200 bit
M + P = 288
200 + P = 288
P = 288 – 200
P = 88

Tabel 1. Tabel Nilai Biner

<table>
<thead>
<tr>
<th>11111111</th>
<th>11110111</th>
<th>11011100</th>
<th>1001 0111</th>
<th>0110 1110</th>
</tr>
</thead>
<tbody>
<tr>
<td>11111111</td>
<td>11111010</td>
<td>11000001</td>
<td>1000 0111</td>
<td>0010 1011</td>
</tr>
<tr>
<td>11111111</td>
<td>11111111</td>
<td>1100 0111</td>
<td>0111 1000</td>
<td>0100 1001</td>
</tr>
<tr>
<td>11110011</td>
<td>11111111</td>
<td>1100 1000</td>
<td>1000 1000</td>
<td>1000 0110</td>
</tr>
<tr>
<td>11111111</td>
<td>11111111</td>
<td>1011 1111</td>
<td>1000 0011</td>
<td>0110 0100</td>
</tr>
</tbody>
</table>

Kemudian 0 Ditambahkan sebanyak 88:

<table>
<thead>
<tr>
<th>10000000</th>
<th>00000000</th>
<th>00000000</th>
<th>00000000</th>
<th>00000000</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
</tr>
<tr>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
</tr>
</tbody>
</table>

Tabel 3.2 Tabel Nila yang diX0

<table>
<thead>
<tr>
<th>11111111</th>
<th>11110111</th>
<th>11011100</th>
<th>1001 0111</th>
</tr>
</thead>
<tbody>
<tr>
<td>0110 1110</td>
<td>11111111</td>
<td>11111010</td>
<td>1000 0111</td>
</tr>
<tr>
<td>0100 0111</td>
<td>11111111</td>
<td>1100 0111</td>
<td>0111 1000</td>
</tr>
<tr>
<td>0111 1000</td>
<td>0100 1001</td>
<td>1100 1000</td>
<td>1000 1000</td>
</tr>
<tr>
<td>1110 1000</td>
<td>1100 1000</td>
<td>1000 0110</td>
<td>1011 1111</td>
</tr>
<tr>
<td>11111111</td>
<td>11111111</td>
<td>1011 1111</td>
<td>1000 0111</td>
</tr>
</tbody>
</table>

Kemudian X0 digabungkan dengan C0:

<table>
<thead>
<tr>
<th>11111111</th>
<th>11110111</th>
<th>11011100</th>
<th>1001 0111</th>
<th>0110 1110</th>
</tr>
</thead>
<tbody>
<tr>
<td>0110 1110</td>
<td>11111111</td>
<td>11111010</td>
<td>1000 0111</td>
<td>0010 1011</td>
</tr>
<tr>
<td>0100 0111</td>
<td>11111111</td>
<td>1100 0111</td>
<td>0111 1000</td>
<td>0100 1001</td>
</tr>
<tr>
<td>0111 1000</td>
<td>0100 1001</td>
<td>1100 1000</td>
<td>1000 1000</td>
<td>1000 0110</td>
</tr>
<tr>
<td>1110 1000</td>
<td>1100 1000</td>
<td>1000 0110</td>
<td>1111 1111</td>
<td></td>
</tr>
<tr>
<td>11111111</td>
<td>11111111</td>
<td>1011 1111</td>
<td>1000 0111</td>
<td>0110 0100</td>
</tr>
</tbody>
</table>

Mengubah Array menjadi State sebanyak 800:

A = [0,0,0] = S = [0]
A = [0,0,1] = S = [1]
A = [0,0,2] = S = [2]
A = [0,0,3] = S = [3]
A = [0,1,4] = S = [4]
A = [0,6,6] = S = [6]
A = [0,8,8] = S = [8]
A = [0,9,9] = S = [9]

<table>
<thead>
<tr>
<th>00000000</th>
<th>00000000</th>
<th>00000000</th>
<th>00000000</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
</tr>
<tr>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
</tr>
<tr>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
</tr>
<tr>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
</tr>
<tr>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
</tr>
<tr>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
</tr>
<tr>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
</tr>
<tr>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
</tr>
</tbody>
</table>

Tabel 3. Tabel Nilai R0 dan X0

<table>
<thead>
<tr>
<th>11111111</th>
<th>11110111</th>
<th>11011100</th>
<th>1001 0111</th>
<th>0110 1110</th>
</tr>
</thead>
<tbody>
<tr>
<td>11111111</td>
<td>11111010</td>
<td>11000001</td>
<td>1000 0111</td>
<td>0010 1011</td>
</tr>
<tr>
<td>11111111</td>
<td>11111111</td>
<td>1100 0111</td>
<td>0111 1000</td>
<td>0100 1001</td>
</tr>
<tr>
<td>11110011</td>
<td>11111111</td>
<td>1100 1000</td>
<td>1000 1000</td>
<td>1000 0110</td>
</tr>
<tr>
<td>11111111</td>
<td>11111111</td>
<td>1011 1111</td>
<td>1000 0011</td>
<td>0110 0100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>10000000</th>
<th>00000000</th>
<th>00000000</th>
<th>00000000</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
</tr>
</tbody>
</table>

ISSN 2301-9425 (Media Cetak)
Pelita Informatika : Informasi dan Informatika, Volume 9, Nomor 4, April 2021
Hal: 266-275
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[253]</td>
<td>[285]</td>
<td>[284]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[254]</td>
<td>[286]</td>
<td>[287]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A = [0,3,0] = S =</th>
<th>A = [1,3,0] = S =</th>
<th>A = [0,4,0] = S =</th>
</tr>
</thead>
<tbody>
<tr>
<td>[384]</td>
<td>[417]</td>
<td>[448]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[385]</td>
<td>[417]</td>
<td>[449]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[386]</td>
<td>[418]</td>
<td>[450]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[387]</td>
<td>[419]</td>
<td>[451]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[388]</td>
<td>[420]</td>
<td>[452]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[389]</td>
<td>[421]</td>
<td>[453]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[390]</td>
<td>[422]</td>
<td>[454]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[391]</td>
<td>[423]</td>
<td>[455]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[392]</td>
<td>[424]</td>
<td>[456]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[393]</td>
<td>[425]</td>
<td>[457]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[394]</td>
<td>[426]</td>
<td>[458]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[395]</td>
<td>[427]</td>
<td>[459]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[396]</td>
<td>[428]</td>
<td>[460]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[397]</td>
<td>[429]</td>
<td>[461]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[398]</td>
<td>[430]</td>
<td>[462]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[399]</td>
<td>[431]</td>
<td>[463]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[400]</td>
<td>[432]</td>
<td>[464]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[401]</td>
<td>[433]</td>
<td>[465]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[402]</td>
<td>[434]</td>
<td>[466]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[403]</td>
<td>[435]</td>
<td>[467]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[404]</td>
<td>[436]</td>
<td>[468]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[405]</td>
<td>[437]</td>
<td>[469]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[406]</td>
<td>[438]</td>
<td>[470]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[407]</td>
<td>[439]</td>
<td>[471]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[408]</td>
<td>[440]</td>
<td>[472]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[409]</td>
<td>[441]</td>
<td>[473]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[410]</td>
<td>[442]</td>
<td>[474]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[411]</td>
<td>[443]</td>
<td>[475]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[412]</td>
<td>[444]</td>
<td>[476]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[413]</td>
<td>[445]</td>
<td>[477]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[414]</td>
<td>[446]</td>
<td>[478]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[415]</td>
<td>[447]</td>
<td>[479]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[481]</td>
<td>[515]</td>
<td>[547]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[498]</td>
<td>[516]</td>
<td>[548]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[483]</td>
<td>[517]</td>
<td>[549]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[482]</td>
<td>[518]</td>
<td>[550]</td>
</tr>
</tbody>
</table>
\[ A = [1,0,7] = S = [712] \]
\[ A = [2,0,7] = S = [716] \]
\[ A = [0,4,17] = S = [788] \]
\[ A = [0,4,18] = S = [789] \]
\[ A = [0,4,19] = S = [790] \]
\[ A = [0,4,20] = S = [791] \]
\[ A = [0,4,21] = S = [792] \]
\[ A = [0,4,22] = S = [793] \]
\[ A = [0,4,23] = S = [794] \]
\[ A = [0,4,24] = S = [795] \]
\[ A = [0,4,25] = S = [796] \]
\[ A = [0,4,26] = S = [797] \]
\[ A = [0,4,27] = S = [798] \]
\[ A = [0,4,28] = S = [799] \]
\[ A = [0,4,29] = S = [800] \]

\[ A = [0,4,0] = S = [771] \]
\[ A = [0,4,1] = S = [772] \]
\[ A = [0,4,2] = S = [773] \]
\[ A = [0,4,3] = S = [774] \]
\[ A = [0,4,4] = S = [775] \]
\[ A = [0,4,5] = S = [776] \]
\[ A = [0,4,6] = S = [777] \]
\[ A = [0,4,7] = S = [778] \]
\[ A = [0,4,8] = S = [779] \]
\[ A = [0,4,9] = S = [780] \]
\[ A = [0,4,10] = S = [781] \]
\[ A = [0,4,11] = S = [782] \]
\[ A = [0,4,12] = S = [783] \]
\[ A = [0,4,13] = S = [784] \]
\[ A = [0,4,14] = S = [785] \]
\[ A = [0,4,15] = S = [786] \]
\[ A = [0,4,16] = S = [787] \]
\[
C[x, 8] = [x, 2, 4] \oplus A [x, 1, 25] \oplus A [x, 3, 3] \oplus \\
A [x, 3, 0] \oplus A [x, 1, 15] \\
C[0, 0] = [0, 2, 4] \oplus A [0, 1, 25] \oplus A [0, 3, 38] \oplus A [0, 1, 30] \\
A [0, 1, 15] \\
C[0, 0] = 0 \oplus A 1 \oplus A 0 \oplus A 0 \oplus A 0 \\
C[x, 9] = [x, 1, 20] \oplus A [2, 0, 11] \oplus A [1, 26] \oplus \\
A [0, 1, 12] \oplus A [x, 4, 2] \\
C[0, 0] = [0, 1, 20] \oplus A [2, 0, 11] \oplus A [1, 0, 6] \oplus A [0, 1, 12] \\
A [4, 2, 0] \\
C[0, 0] = 0 \oplus A 0 \oplus A 0 \oplus A 0 \oplus A 0 \\
C[x, 10] = [x, 2, 3] \oplus A [2, 2, 7] \oplus A [2, x, 13] \oplus \\
A [3, x, 23] \oplus A [4, x, 26] \\
C[0, 0] = [0, 2, 3] \oplus A [2, 2, 7] \oplus A [2, 0, 13] \oplus A [3, 0, 23] \\
A [4, 0, 26] \\
C[0, 0] = 0 \oplus A 0 \oplus A 0 \oplus A 0 \oplus A 0 \\
C[x, 11] = [x, 2, 23] \oplus A [x, 3, 30] \oplus A [x, 4, 28] \oplus \\
A [4, 1, x] \oplus A [4, 2, x] \\
C[0, 0] = [0, 2, 23] \oplus A [0, 3, 30] \oplus A [0, 4, 28] \oplus A [4, 1, x] \\
A [4, 2, 0] \\
C[0, 0] = 0 \oplus A 0 \oplus A 0 \oplus A 0 \oplus A 0 \\
C[x, 12] = [x, 2, 23] \oplus A [x, 3, 30] \oplus A [x, 4, 28] \oplus \\
A [4, 1, x] \oplus A [4, 2, x] \\
C[0, 0] = [0, 2, 23] \oplus A [0, 3, 30] \oplus A [0, 4, 28] \oplus A [4, 1, x] \\
A [4, 2, 0] \\
C[0, 0] = 0 \oplus A 0 \oplus A 0 \oplus A 0 \oplus A 0 \\
C[x, 13] = [x, 2, 23] \oplus A [x, 3, 30] \oplus A [x, 4, 28] \oplus \\
A [4, 1, x] \oplus A [4, 2, x] \\
C[0, 0] = [0, 2, 23] \oplus A [0, 3, 30] \oplus A [0, 4, 28] \oplus A [4, 1, x] \\
A [4, 2, 0] \\
C[0, 0] = 0 \oplus A 0 \oplus A 0 \oplus A 0 \oplus A 0 \\
C[x, 14] = [x, 2, 23] \oplus A [x, 3, 30] \oplus A [x, 4, 28] \oplus \\
A [4, 1, x] \oplus A [4, 2, x] \\
C[0, 0] = [0, 2, 23] \oplus A [0, 3, 30] \oplus A [0, 4, 28] \oplus A [4, 1, x] \\
A [4, 2, 0] \\
C[0, 0] = 0 \oplus A 0 \oplus A 0 \oplus A 0 \oplus A 0 \\
\]

For J = “0 To 15”

1. J = 0, i = 0
   Set x(16+j) To M(i*16+j)
   Set x[16+0] To M[0*16+0]
   Set x[16] To M[0]
   Set x[16] To 100

   Set x[32+j] To x[16+j] xor x[j]
   Set x[32+0] To x[16+0] xor x[0]
   Set x[32] To x[16] xor x[0]
   Set x[32] To 16 xor 0
   Set x[32] To 16

   X = 00000000 00000000 1000000000 0000000000

2. J = 1, i = 0
   Set x[16+1] To M[i*17+j]
   Set x[16+1] To M[0*17+1]
   Set x[17] To M[1]
   Set x[17] To 106

   Set x[32+j] To x[17+j] xor x[j]
   Set x[32+1] To x[17+1] xor x[1]
   Set x[33] To x[18] xor x[1]
   Set x[33] To 19 xor 0
   Set x[33] To 19

   X = 0000000000 00000000 10010610200000 00000000

   X16 X17 X18 X19 X20 X21 X22 X23

3. J = 2, i = 0
   Set x[16+j] To M(i*16+j)
   Set x[16+2] To M[0*16+2]
   Set x[18] To M[2]
   Set x[18] To 102

   Set x[32+j] To x[18+j] xor x[j]
   Set x[32+2] To x[18+2] xor x[2]
   Set x[34] To x[20] xor x[2]
   Set x[34] To 22 xor 0
   Set x[34] To 22

   X = 0000000000 00000000 10010610200000 00000000

   X16 X17 X18 X19 X20 X21 X22 X23

4. J = 3, i = 0
   Set x[16+j] To M[i*16+j]
   Set x[16+3] To M[0*16+3]
   Set x[19] To M[3]
   Set x[19] To 96

   Set x[32+j] To x[19+j] xor x[j]
   Set x[32+3] To x[19+3] xor x[3]
   Set x[35] To x[23] xor x[3]
   Set x[35] To 20 xor 0
   Set x[35] To 20

   X = 0000000000 00000000 100106102960000 00000000

   X16 X17 X18 X19 X20 X21 X22 X23

5. J = 4, i = 0
   Set x[16+j] To M[i*16+j]
   Set x[16+4] To M[0*16+4]
   Set x[20] To 103

   Set x[32+j] To x[20+j] xor x[j]
   Set x[36] To 19 xor 0
   Set x[36] To 19

   X = 00000000 00000000 10010610296103000 00000000

   X16 X17 X18 X19 X20 X21 X22 X23

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### 1. Perancangan Aplikasi

Perancangan aplikasi ini adalah sebagai berikut:

1. Pengolahan nantinya. Adapun kebutuhan sistem terbagi dua, yang digunakan, agar program bisa berjalan dengan baik dalam sebuah bahasa pemrograman. Pengimplementasi suatu sistem yang merupakan proses menerjemahkan hasil analisa dan perancangan sistem ke dalam sebuah bahasa pemrograman. Pengimplementasi suatu sistem akan berpengaruh pada spesifikasi komputer yang digunakan, agar program bisa berjalan dengan baik berdasarkan spesifikasi perangkat keras (hardware) dan perangkat lunak (software) harus sesuai. Sistem pengolahan suatu program terdiri dari prosedur dan pelaksanaan data, komputer sebagai sarana pengolahan program harus menyediakan fasilitas-fasilitas yang mendukung dalam pengolahan nantinya. Adapt kebutuhan sistem terbagi dua yaitu perangkat keras dan perangkat lunak.

Untuk membuat aplikasi ini dibutuhkan spesifikasi perangkat keras sebagai berikut:

1. Prosesor : Intel® Celeron® CPU N3060 @1,60GHz
2. RAM : 2 GB
3. Input : Mouse
4. Output : Monitor

Perangkat lunak (software) yang digunakan dalam perancangan aplikasi ini adalah sebagai berikut:

1. Sistem Operasi : Microsoft Windows 10
2. Program Aplikasi : Cryptool 2.0

### 4. IMPLEMENTASI

Kebutuhan sistem merupakan spesifikasi tentang perangkat yang akan digunakan yaitu software (perangkat lunak) dan hardware (perangkat keras) yang dibutuhkan oleh aplikasi. Berdasarkan kebutuhan sistem terdapat pengimplementasi suatu sistem yang merupakan proses menerjemahkan hasil analisa dan perancangan sistem ke dalam sebuah bahasa pemrograman. Pengimplementasi suatu sistem akan berpengaruh pada spesifikasi komputer yang digunakan, agar program bisa berjalan dengan baik berdasarkan spesifikasi perangkat keras (hardware) dan perangkat lunak (software) harus sesuai. Sistem pengolahan suatu program terdiri dari prosedur dan pelaksanaan data, komputer sebagai sarana pengolahan program harus menyediakan fasilitas-fasilitas yang mendukung dalam pengolahan nantinya. Adapt kebutuhan sistem terbagi dua yaitu perangkat keras dan perangkat lunak.
A. Tampilan Pengujian

Aplikasi pengujian implementasi metode keccak untuk otentikasi hasil citra ijazah yang akan dilakukan pengujian hasil dari pembentukan kode fungsi hash dengan menerapkan metode keccak yang digunakan untuk hasil dari implementasi metode keccak yang mendeteksi keaslian dengan menggunakan aplikasi Cryptool 2.0 sebagai berikut :

Gambar 1. Aplikasi Cryptool 2.0

Pada form yang digunakan pada Aplikasi Cryptool 2.0 tersebut terdapat beberapa langkah yang dapat dilakukan oleh user untuk menjalankan pengujian implementasi metode keccak untuk mendeteksi file hasil citra ijazah diantaranya Menginputkan file citra ijazah adalah proses dimana memanggil file dokumen yang akan di cari nilai keccak seperti tampilan gambar 2.

Gambar 2. Menginputan File awal

Memilih metode keccak adalah proses dimana memilih metode yang digunakan untuk mendapatkan hasil dari metode keccak. Berikut hasil dari pengujian aplikasiCRYptool 2.0 yang asli maupun yang telah di edit.

Gambar 3. Memilih Metode Keccak

5. KESIMPULAN

Dari hasil semua pembahasan yang penulis lakukan terhadap penelitian ini penulis dapat menarik beberapa kesimpulan yang terkait dengan proses penelitian. Metode keccak dapat mengetahui dan mendeteksi keaslian pada sebuah file citra ijazah dengan cara membandingkan nilai hash dari citra asli yaitu citra asli ijazah dengan citra hasil yang diduga manipulasi. Dengan menggunakan cryptool 2.0 dapat dilakukan pembuktian orisinalitas sebuah file citra ijazah dengan membandingkan nilai hash citra asli dan citra yang dilakukan modifikasi yaitu dengan membandingkan hasil berdasarkan perubahan resolusi citra, perubahan nilai pixel, perubahan karakter dan perubahan kontras citra.

REFERENCES