Vol. 4 No. 1, April 2023, pp. 66-75 **Jurnal Pepadun**


# Penerapan Pengkodean *Data Base*-64 dan Kode QR Citra Foto Wajah untuk Autentikasi Tanda Tangan Dokumen Digital dengan *Library Javascript*

**[1]Hana Afriliza dan [2]Febi Eka Febriansyah**

[1,2] Jurusan Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Lampung
Jalan Soemantri Brojonegoro No.1 Gedung Meneng, Bandar Lampung, Provinsi Lampung, Indonesia
e-mail: [1]hana.afriliza1012@students.unila.ac.id, [2]febieka.febriansyah@fmipa.unila.ac.id

***Abstract*** *— The application of technology has an impact on replacing conventional methods with modern ones, one of which is providing document authentication. Signatures can be used as a solution in authenticating documents because they are considered capable of being used as mathematical proof that the document is not modified illegally. The difference that lies between the signature on digital documents and conventional documents is in terms of security. Signatures on conventional documents are prone to forgery. This is due to the absence of evidence to ensure that the document has been signed by the party concerned. Thus, the use of signatures on digital documents is more secure than on conventional documents. The security that is carried out to ensure its authenticity is by utilizing a QR code which contains an id, photo of the signatory which is encoded first using base-64 coding, and a link which is used to verify the authenticity of the document. The verification process carried out to ensure its authenticity is by scanning the QR code embedded in the document. The library developed using Javascript and Waterfall is used as a development method, and the testing method used is black-box testing.*

***Keywords:*** *Authentication; Base-64 Encoding; QR Code.*

## 1. PENDAHULUAN

Penerapan teknologi berdampak pada tergantikannya cara-cara konvensional menjadi modern, salah satunya dalam memberikan autentikasi dokumen. Tanda tangan dapat digunakan sebagai salah satu solusi dalam melakukan autentikasi dokumen karena dianggap mampu dijadikan bukti secara matematis, bahwa dokumen tidak dimodifikasi secara ilegal [1]. Dengan adanya tanda tangan maka dokumen yang telah ditandatangani berasal dari penandatangan yang telah diketahui [2]. Selain itu, tanda tangan juga dapat digunakan untuk mengidentifikasi keaslian dokumen dalam membuktikan bahwa penandatangan telah menyetujui isi dari dokumen tersebut [3].

Perbedaan yang terletak antara tanda tangan pada dokumen digital dan dokumen konvensional adalah dari segi keamanannya. Tanda tangan pada dokumen konvensional rentan mengalami pemalsuan. Hal ini disebabkan oleh tidak adanya bukti untuk memastikan bahwa dokumen tersebut telah ditandatangani oleh pihak yang bersangkutan dan tidak mengalami perubahan isi dokumen. Penggunaan tanda tangan pada dokumen digital dipercaya dapat mempercepat proses administrasi yang dilakukan, sehingga tanda tangan pada dokumen konvensional menjadi kurang efisien.

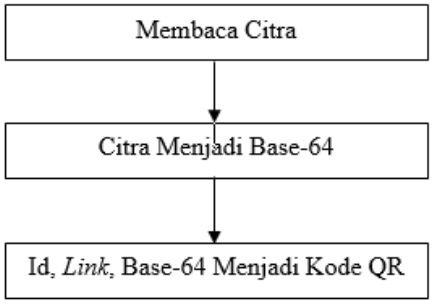
Pada penelitian ini tanda tangan yang digunakan tidak mengalami proses enkripsi, akan tetapi tanda tangan konvensional yang didigitalisasi kemudian ditambahkan pengamanan. Pengamanan yang dilakukan untuk menjamin keasliannya adalah dengan memanfaatkan *Quick Response* (QR) *Code* atau kode QR yang di dalamnya terdapat id dan *link* yang digunakan untuk melakukan verifikasi keaslian dokumen. Foto yang disisipkan di-*encode* terlebih dahulu menggunakan pengkodean base-64 sebelum kode QR dibuat. Penggunaan kode QR dinilai dapat memberikan jaminan keamanan dan privasi di balik kode tersebut terhadap penandatangan yang melakukan pengiriman pesan [4]. Proses verifikasi yang dilakukan untuk

memastikan keasliannya adalah dengan melakukan pemindaian kode QR yang disematkan pada dokumen tersebut [5].

## 2. METODOLOGI PENELITIAN

### 2.1 Skema *Library*

Gambar 1 memperlihatkan skema *library* yang direpresentasikan menggunakan blok diagram. Blok diagram adalah gambaran dasar dalam merancang sebuah *library*, khususnya untuk citra menjadi kode QR. Setiap bagian blok diagram memiliki fungsinya masing-masing, sehingga dengan memahami hal tersebut, maka *library* yang dirancang sudah dapat dibangun dengan baik.




Gambar 1. Blok diagram.

Pada saat membaca citra dari webcam, fungsi yang dibutuhkan adalah fungsi *set(), attach(), takesnapshot().* Kemudian untuk mengubah citra menjadi base-64, dibutuhkan fungsi *webcamsnap()* sehingga citra menjadi data URI (*string base*-64). Id dan *link* verifikasi kemudian di-*generate* menggunakan kode QR *generator* dengan bantuan API. API atau *Application Programming Interface* adalah konsep fungsi antarmuka pemrograman aplikasi, yang digunakan dan dimanfaatkan untuk mengakses suatu aplikasi tanpa mengubah struktur kode utama maupun basis data sistem, serta memudahkan komunikasi antar sistem sekalipun berada pada *platform* yang berbeda [6], sehingga hasil akhir yang didapat berupa kode QR.

Untuk menggambarkan hubungan dan rangkaian proses suatu aktivitas secara detail dengan ringkas, maka diperlukan sebuah *flowchart*. Berikut penjelasan flowchart pada penelitian ini berdasarkan uraian blok diagram yang telah dijelaskan pada Gambar 1.

#### 2.1.1. Proses Membaca Citra

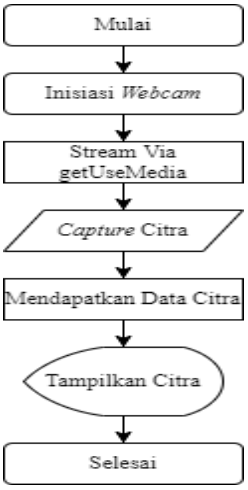
Pada saat menginisiasi *webcam*, API getUserMedia memberikan akses kepada *webcam* pengguna untuk melakukan pengambilan citra. Citra tersebut akan menghasilkan sebuat data citra yang kemudian akan menampilkan sebuah citra. *Flowchart* dari proses membaca citra dapat dilihat pada Gambar 2.

#### 2.1.2. Proses Citra Menjadi Base-64

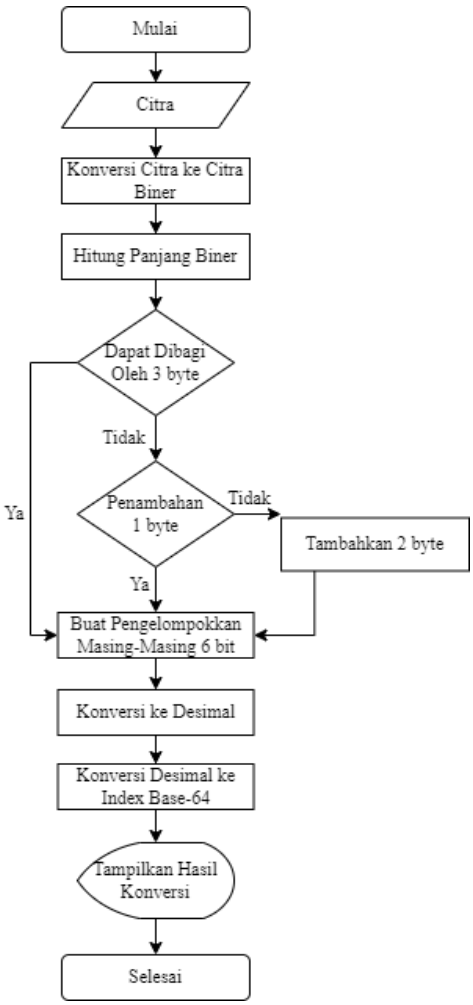
*Flowchart* dari proses *encoding* base-64 dapat dilihat pada Gambar 3. Adapun langkah-langkah *encoding* menggunakan base-64 adalah sebagai berikut.

a) Konversi citra ke biner.
b) Cari bilangan biner 8-bit dari hasil konversi citra.
c) Gabungkan 8-bit menjadi 24-bit.
d) Kemudian pecahan 24-bit dibagi menjadi 6-bit. Maka akan menghasilkan 4 pecahan.
e) Masing-masing pecahan diubah ke dalam nilai desimal.

f) Terakhir, jadikan nilai-nilai desimal tersebut menjadi indeks untuk memilih karakter penyusun dari *base*-64.



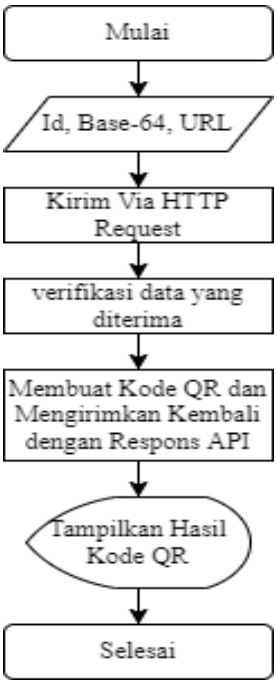

Gambar 2. Proses membaca citra.




Gambar 3. Proses citra menjadi *base*-64.

2.1.3. Proses Id dan *Link* Verifikasi menjadi Kode QR

Proses pertama yaitu menganalisis data berupa id dan *link* verifikasi akan di-*generate* menjadi kode QR. Kemudian data dikirim via HTTP *request* dan diverifikasi. Jika tidak terdapat *error*, maka akan dibuat kode QR dan mengirimkan kembali respon API. Hasil *output*-nya ditampilkan gambar Kode QR. Berikut *flowchart* dari proses id dan *link* verifikasi menjadi kode QR dapat dilihat pada Gambar 4.




Gambar 4. Proses *generate* QR.

**2.2 Pengkodean yang Digunakan Pada *Library***

Pengkodean yang efisien adalah pengkodean yang meminimumkan kebutuhan ruang dan waktu dalam proses *encoding* dan *decoding*. Pengkodean yang digunakan pada *library* ini adalah pengkodean *base*-64. Pengkodean *base*-64 digunakan sebagai metode dalam melakukan *encoding* terhadap data *binary* yang didasarkan pada bilangan dasar 64. Proses *encoding* algoritme *base*-64 lebih cepat jika dibandingkan dengan pengkodean lainnya. Transformasi *base*-64 menghasilkan karakter yang terdiri dari A-Z, a-z dan 0-9, serta ditambah dengan dua karakter terakhir yang bersimbol yaitu + dan / serta satu buah karakter = yang digunakan sebagai pengisi pad dalam menyesuaikan dan menggenapkan data *binary* [7]. Transformasi *base*-64 banyak digunakan sebagai format untuk mengirimkan data karena hasil dari *encode base*-64 berupa *plaintext*, sehingga data yang dihasilkan mudah untuk dikirim [8].

**2.3 Implementasi**

Pengembangan *library* menggunakan metode *Waterfall*. Metode *Waterfall* merupakan model klasik yang bersifat terurut dalam merancang perangkat lunak [9]. Metode ini disebut dengan *Waterfall* karena pada setiap tahap yang dilalui harus menunggu tahapan sebelumnya selesai dan berjalan secara sekuensial yang dimulai dari analisis, desain, pengkodean, dan pengujian [10]. Bahasa pemrograman dalam penelitian ini ditulis menggunakan JavaScript. Pengujian pada penelitian ini berfokus pada perangkat lunak, baik secara fungsional dan nonfungsional, untuk memastikan bahwa telah dilakukan pengujian pada setiap bagian untuk meminimalkan kesalahan dan kesesuaian *output* [10]. Pengujian dilakukan dengan menguji setiap komponen fungsionalitas *library* menggunakan *black-box testing* dengan teknik *equivalence partitioning*.

*Black-box testing* adalah metode pengujian yang berfokus pada spesifikasi fungsional dari perangkat lunak [11]. *Equivalence partitioning* merupakan pengujian yang membagi masukan dari program ke dalam kelas-kelas data untuk memperoleh sebuah *test case*. Perancangan *test case* berdasarkan evaluasi kelas *equivalence* untuk kondisi masukan yang menggambarkan sekumpulan kondisi valid atau tidak valid [12].

## 3. HASIL DAN PEMBAHASAN

Penelitian ini menghasilkan autentikasi tanda tangan pada dokumen digital dalam bentuk *library* JavaScript. *Library* digunakan untuk memfungsikan sistem berbasis web yang membutuhkan autentikasi dokumen. Metode *Waterfall* digunakan untuk mengimplementasikan kode dengan bahasa pemrograman JavaScript. *Library* yang dihasilkan memiliki tiga fungsi, yaitu tanda tangan, kamera, dan kode QR yang di dalamnya terdapat id dan *link* verifikasi yang digunakan untuk melakukan verifikasi dokumen. Pembahasan pada penelitian ini dibagi menjadi dua, yaitu pembahasan mengenai kode program *library* dan implementasi *library*.

### 3. 1. Kode Program *Library*

Pembahasan pada penelitian ini dibagi menjadi tiga, yaitu pembahasan mengenai kode program *library*, implementasi *library* pada sistem, dan pengujian.

a. Tanda Tangan

Tanda tangan yang terdapat pada *library* ini memiliki ukuran lebar 400 px dan tinggi 200 px, dengan format *Portable Network Graphics* (PNG). Fungsi yang terdapat pada tanda tangan yaitu fungsi *signature pad*, *save*, dan *clear*. Berikut untuk potongan kode programnya dapat dilihat pada Gambar 5.

```
var signaturePad = new SignaturePad(document.getElementById(this.pad), {
    backgroundColor: 'rgba(255, 255, 255, 0)',
    penColor: 'rgb(0, 0, 0)'
    });

function getSignaturePad() {
  var imageData = signaturePad.toDataURL('image/png');
  $(result).val(imageData)
  $(img_result).attr('src',"data:"+imageData);
}
  $('#save').click(function() {
    getSignaturePad();
    return false;
  });
  $('#clear').click(function(e) {
    e.preventDefault();
    signaturePad.clear();
  })
```

Gambar 5. Kode program tanda tangan.

b. Kamera

Kamera yang terdapat pada *library* ini dikonfigurasi terlebih dahulu menggunakan fungsi *set()*. Di dalam fungsi *set()* terdapat empat *properties* yaitu *width*, *height*, *format*, dan *quality*. Untuk ukuran lebar dan tinggi sebesar 120 px, dengan format *Joint Photographic Experts Group* (JPEG), serta

resolusi 50 KB. Resolusi pada kamera dapat disesuaikan dengan kebutuhan, akan tetapi semakin besar resolusi yang digunakan, maka *string base*-64 yang dihasilkan semakin banyak.

Berikutnya yaitu fungsi *attach()*, digunakan untuk menampilkan kamera. Untuk menangkap gambar yang telah ditampilkan oleh kamera yaitu dengan memanggil fungsi *takesnapshot()*. Selanjutnya, fungsi *webcamsnap()* yang memiliki data URI sebagai parameter yang digunakan untuk sumber gambar. *Data Uniform Resource Identifier* (URI) adalah skema yang memungkinkan data dikodekan menjadi *string* (*base*-64). Berikut untuk potongan kode programnya dapat dilihat pada Gambar 6.

```
Webcam.set({
  width: 120,
  height: 120,
  image_format: 'jpeg',
  jpeg_quality: 1
});
Webcam.attach(this.camera);
takeSnapShot = function () {
  Webcam.snap(function (data_uri) {
    console.log(data_uri);
    generateQR(data_uri, img);
  });
}
}
```

Gambar 6. Kode program kamera.

c. Kode QR

Kode QR pada *library* ini di-*generate* menggunakan API dari https://api.qrserver.com/v1/create-qr-code/, dengan ukuran sebesar 250 x 250. Data yang terdapat di dalam kode QR yaitu berupa id dan *link* yang digunakan untuk melakukan verifikasi dokumen. Berikut untuk potongan kode programnya dapat dilihat pada Gambar 7.

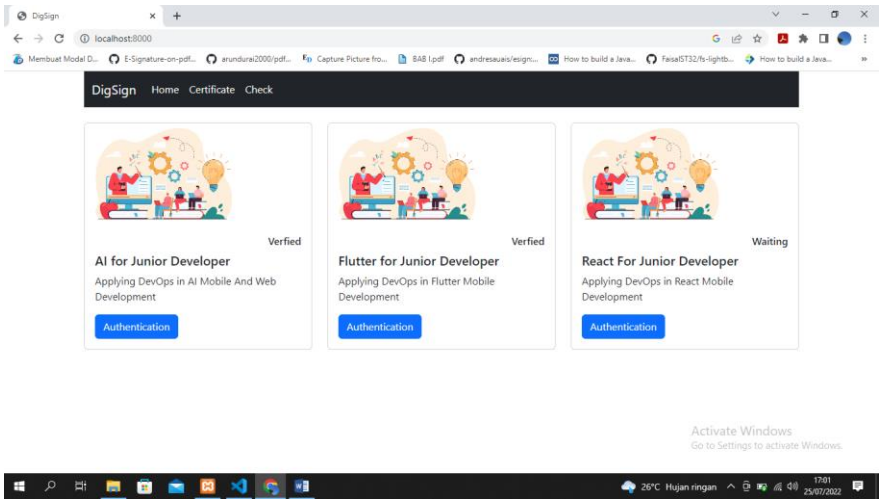
```
const generateQR = (data_uri, img) => {
  const payload = `{
  id: ${$("#id_dokumen").val()},
  foto: ${data_uri},
  link: "http://localhost:8000/verify?id=${$("#id_dokumen").val()}"
  }`;
  $.ajax({
    type: "POST",
    url: `https://api.qrserver.com/v1/create-qr-code/
size=250x250&data=${payload}`,
    xhr:function(){
      var xhr = new XMLHttpRequest();
      xhr.responseType= 'blob'
      return xhr;
    },
    success: async function(response) {
      const img_qr = document.getElementById(img);
      var url = window.URL || window.webkitURL;
      img_qr.src = url.createObjectURL(response);
      const gambar = await blobToBase64(response);
      $("#gambar").val(gambar)
    },
  });
}
```

Gambar 7. Kode program kode QR.

3. 2. Implementasi *Library* Pada Sistem

a. Tampilan Halaman Utama

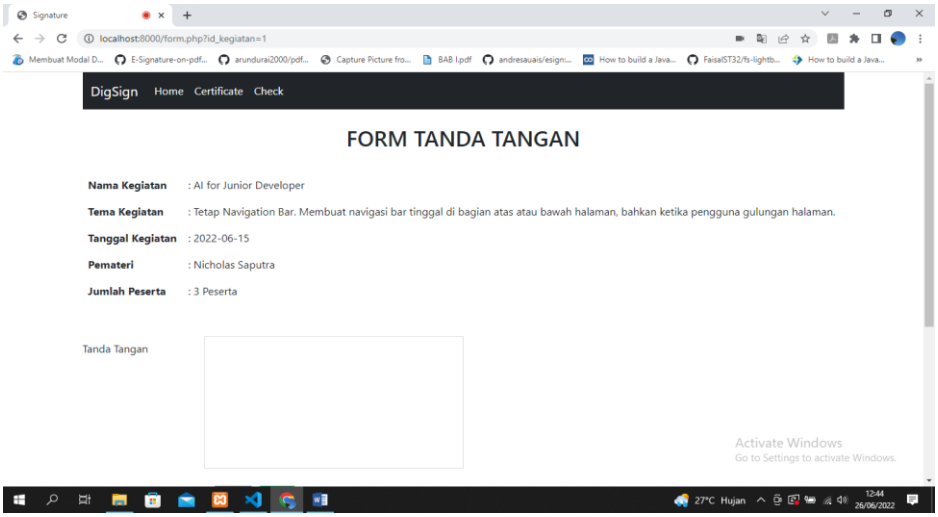
Dokumen yang akan dan telah diberikan autentikasi, akan muncul pada halaman utama. Jika dokumen tersebut belum ditandatangani, maka statusnya adalah *waiting*, dan apabila telah ditandatangani, statusnya akan berubah menjadi *verified*. Tampilannya terdapat pada Gambar 8.




Gambar 8. Halaman utama.

b. Tampilan *Form* Autentikasi

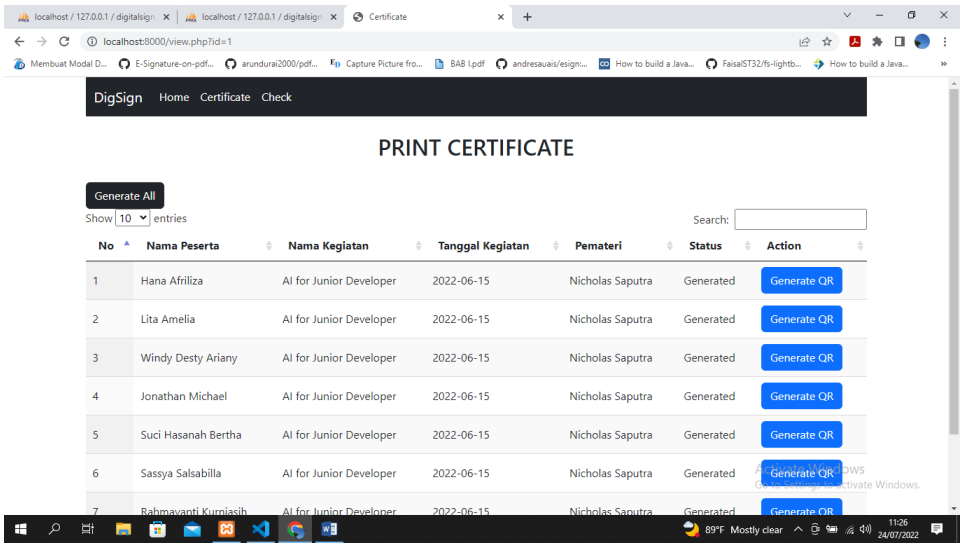
*Form* autentikasi akan meminta untuk mengisikan tanda tangan dan *capture* gambar. Berikut untuk tampilannya dapat dilihat pada Gambar 9.




Gambar 9. Form autentikasi.

c. Tampilan Halaman *Generate* Kode QR

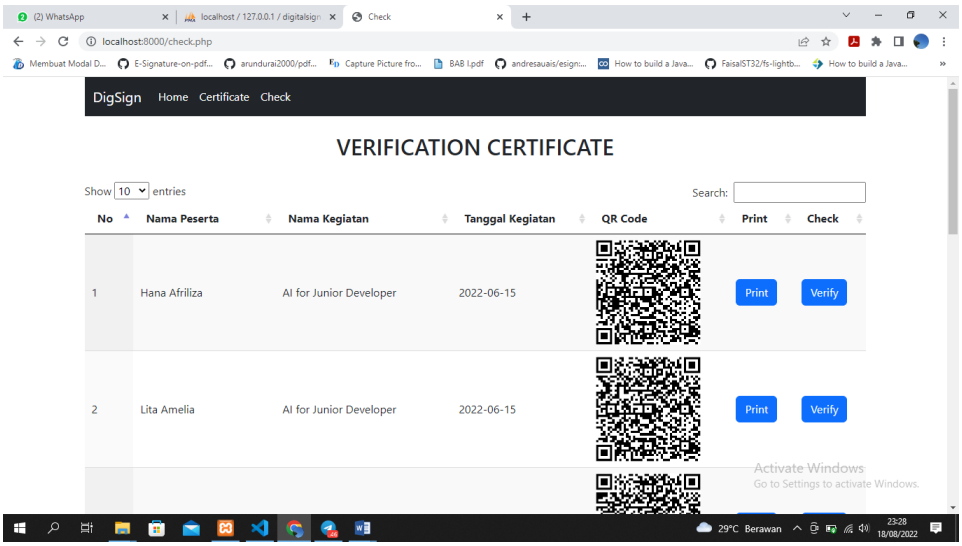
Dokumen yang telah diberi autentikasi akan ditampilkan seluruhnya pada halaman *generate* kode QR. Pada bagian ini, dapat melakukan pemberian kode QR pada dokumen. Berikut untuk tampilannya dapat dilihat pada Gambar 10.




Gambar 10. *Generate* kode QR.

d. Tampilan Halaman Cetak, Verifikasi, dan Base-64

Dokumen yang telah diberi autentikasi berupa kode QR akan ditampilkan seluruhnya pada halaman ini. Pada bagian ini, dapat dilakukan proses cetak, verifikasi, dan *base*-64 dokumen. Berikut untuk tampilannya dapat dilihat pada Gambar 11.




Gambar 11. Halaman cetak, verifikasi, dan *base*-64.

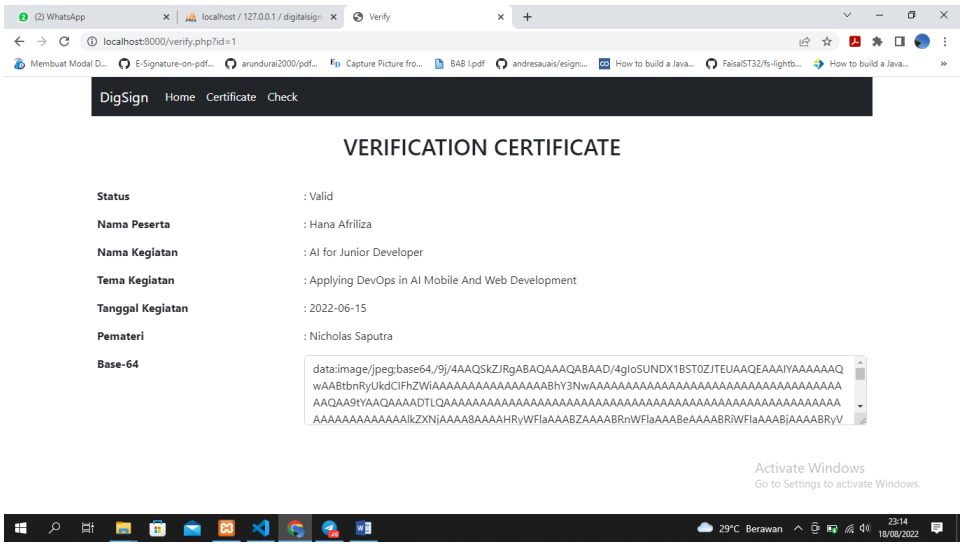
e. Tampilan Hasil Autentikasi

Dokumen yang diberikan autentikasi menggunakan format Portable Document Format atau PDF. Pada dokumen tersebut terdapat kode QR yang apabila di pindai, akan menghasilkan id, dan *link* untuk verifikasi dokumen. Berikut untuk tampilannya dapat dilihat pada Gambar 12.


Gambar 12. Hasil autentikasi.

f. Tampilan Halaman Detail Verifikasi

*Link* yang terdapat di dalam kode QR, apabila di-klik akan mengarah ke halaman detail verifikasi. Halaman detail verifikasi digunakan untuk mengetahui apakah dokumen tersebut *valid* atau *invalid*. Berikut untuk tampilannya dapat dilihat pada Gambar 13.




Gambar 13. Detail verifikasi.

## 4. KESIMPULAN

Dari penelitian yang telah dilakukan, dapat disimpulkan bahwa telah berhasil dikembangkan *library* untuk mempermudah dalam memberikan autentikasi dokumen. *Library* yang digunakan memiliki keamanan yang baik. *Library* yang dikembangkan juga dapat digunakan pada sistem yang membutuhkan autentikasi dokumen.

## DAFTAR PUSTAKA


[1] Y. Anshori, A. Y. Erwin Dodu & D. M. P. Wedananta, "Implementasi Algoritma Kriptografi Rivest Shamir Adleman (RSA) pada Tanda Tangan Digital," *Techno.Com*, vol. 18, no. 2, pp. 110–121, 2019, doi: 10.33633/tc.v18i2.2166.

[2] A. G. P. Suratma & A. Azis, "Tanda Tangan Digital Menggunakan QR Code Dengan Metode Advanced Encryption Standard," *Techno*, vol. 18, no. 1, pp. 59–68, 2017.

[3] R. A. Azdy, "Tanda tangan Digital Menggunakan Algoritme Keccak dan RSA," *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 5, no. 3, pp. 184–191, 2016, doi: 10.22146/jnteti.v5i3.255.

[4] F. Nuraeni, Y. H. Agustin, D. Kurniadi & I. D. Ariyanti, "Implementasi Skema QR-Code dan Digital Signature menggunakan Kombinasi Algoritma RSA dan AES untuk Pengamanan Data Sertifikat Elektronik," *Semin. Nas. Teknol. Informasi, Komun. dan Ind. 12*, no. December, pp. 43–52, 2020.

[5] A. Lorien & T. Wellem, "Implementasi Sistem Otentikasi Dokumen Berbasis Quick Response (QR) Code dan Digital Signature," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 5, no. 4, pp. 663–671, 2021, doi: 10.29207/resti.v5i4.3316.

[6] D. Wijonarko & B. W. R. Mulya, "Pengembangan Antarmuka Pemrograman Aplikasi Menggunakan Metode RESTful pada Sistem Informasi Akademik Politeknik Kota Malang," *Smatika J.*, vol. 8, no. 02, pp. 63–66, 2018, doi: 10.32664/smatika.v8i02.202.

[7] F. Wahyu. C, A. P. Rahangiar & F. de Fretes, "Penerapan Algoritma Gabungan RC4 dan Base64 pada Sistem Keamanan E- Commerce (Application of Joint RC4 and BASE64 Algorithm for E-Commerce Security System)," no. June 2012, 2016.

[8] E. Gunadhi & A. P. Nugraha, "Penerapan Kriptografi Base64 Untuk Keamanan URL (Uniform Resource Locator) Website Dari Serangan SQL Injection," *J. Algoritm.*, vol. 13, no. 2, pp. 391–398, 2017, doi: 10.33364/algoritma/v.13-2.391.

[9] I. Sholikhah, M. Sairan & N. O. Syamsiah, "Aplikasi Pembelian Dan Penjualan Barang Dagang Pada CV Gemilang Muliatama Cikarang," *Tek. Komput. AMIK BSI*, vol. III, no. 1, pp. 16–23, 2017.

[10] H. Nur, "Penggunaan Metode Waterfall Dalam Rancang Bangun Sistem Informasi Penjualan," *Gener. J.*, vol. 3, no. 1, p. 1, 2019, doi: 10.29407/gj.v3i1.12642.

[11] T. Hidayat & M. Muttaqin, "Pengujian Sistem Informasi Pendaftaran dan Pembayaran Wisuda Online menggunakan Black Box Testing dengan Metode Equivalence Partitioning dan Boundary Value Analysis," *J. Tek. Inform. UNIS JUTIS*, vol. 6, no. 1, pp. 2252–5351, 2018, [Online]. Available: www.ccssenet.org/cis.

[12] A. Krismadi, A. F. Lestari, A. Pitriyah, I. W. P. A. Mardangga, M. Astuti & A. Saifudin, "Pengujian Black Box berbasis Equivalence Partitions pada Aplikasi Seleksi Promosi Kenaikan Jabatan," *J. Teknol. Sist. Inf. dan Apl.*, vol. 2, no. 4, p. 155, 2019, doi: 10.32493/jtsi.v2i4.3771.