Implementasi Secure *Hash* Algorithm 256 (SHA-256) dan Algoritma Rivest Shamir Adleman (RSA) untuk Mendeteksi Perbedaan pada Dokumen PDF

TUGAS AKHIR

Tugas Akhir sebagai salah satu syarat untuk memperoleh gelar sarjana Informatika
Universitas Pembangunan Nasional "Veteran" Yogyakarta

Disusun Oleh:
Zaidan Noor Irfan
123200066

PROGRAM STUDI INFORMATIKA

JURUSAN INFORMATIKA

FAKULTAS TEKNIK INDUSTRI

UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN" YOGYAKARTA

2024

HALAMAN PENGESAHAN PEMBIMBING

Implementasi Secure *Hash* Algorithm 256 (SHA-256) dan Algoritma Rivest Shamir Adleman (RSA) untuk Mendeteksi Perbedaan pada Dokumen PDF

Disusun oleh
Zaidan Noor Irfan
123200066

Telah diuji dan dinyatakan lulus oleh pembimbing

Pada Tanggal :

Menyetujui,
Pembimbing

Rifki Indra Perwira, S.Kom., M.Eng.
NIP. 19830708 202121 1 001

Mengetahui,
Koorinator Program Studi

Dessyanto Boedi Prasetyo, S.T., M.T.
NIDN. **0505127501**

HALAMAN PENGESAHAN PENGUJI

Implementasi Secure *Hash* Algorithm 256 (SHA-256) dan Algoritma Rivest Shamir Adleman (RSA) untuk Mendeteksi Perbedaan pada Dokumen PDF

Disusun oleh
Zaidan Noor Irfan
123200066

Telah diuji dan dinyatakan lulus oleh penguji

Pada Tanggal :

Menyetujui,

Penguji I	Penguji II
Rifki Indra Perwira, S.Kom., M.Eng.	Andiko Putro Suryotomo, S.Kom. M.Cs.
NIDN: 0508078301	NIDN: 0030098504
Penguji III	**Penguji IV**
Rochmat Husaini, S.Kom., M.Kom.	Bambang Yuwono, S.T., M.T.
NIDN: 0026048804	NIDN: 0512027401

SURAT PERNYATAAN KARYA ASLI TUGAS AKHIR

Sebagai mahasiswa Program Studi Informatika Fakultas Teknik Industri Universitas Pembangunan Nasional "Veteran" Yogyakarta, yang bertanda tangan dibawah ini, saya:

Nama : Zaidan Noor Irfan
Nim : 123200066

Menyatakan bahwa karya ilmiah saya yang berjudul:

Implementasi Secure *Hash* Algorithm 256 (SHA-256) dan Algoritma Rivest Shamir Adleman (RSA) untuk Mendeteksi Perbedaan pada Dokumen PDF

merupakan karya asli saya sendiri dan belum pernah dipublikasikan dimanapun. Apabila di kemudian hari, karya saya disinyalir bukan merupakan karya asli saya, maka saya bersedia menerima konsekuensi apapun yang diberikan Program Studi Informatika Fakultas Teknik Industri Universitas Pembangunan "Veteran" Yogyakarta kepada saya.

Demikian surat pernyataan ini saya buat dengan penuh tanggung jawab

Dibuat di : Yogyakarta

Pada tanggal : 21 Agustus 2024

Yang menyatakan

Zaidan Noor Irfan

PERNYATAAN BEBAS PLAGIAT

Saya yang bertanda tangan dibawah ini:

Nama	: Zaidan Noor Irfan
Nim	: 123200066
Fakultas/Prodi	: Fakultas Teknik Industri/Informatika

Dengan ini saya menyatakan bahwa judul tugas akhir

Implementasi Secure *Hash* Algorithm 256 (SHA-256) dan Algoritma Rivest Shamir Adleman (RSA) untuk Mendeteksi Perbedaan pada Dokumen PDF

adalah hasil kerja saya sendiri dan benar bebas dari plagiasi kecuali cuplikan serta ringkasan yang terdapat di dalamnya setelah saya jelaskan sumbernya (sitasi) dengan jelas. Apabila pernyataan ini terbukti tidak benar, maka saya bersedia menerima sanksi sesuai peraturan Mendiknas RI No. 17 Tahun 2000 dan Peraturan Perundang-undangan yang berlaku

Demikian surat pernyataan ini saya buat dengan penuh tanggung jawab

Yogyakarta, 21 Agustus 2024

Zaidan Noor Irfan

ABSTRAK

Dokumen merupakan kumpulan informasi tertulis atau rekaman yang disusun secara sistematis untuk tujuan tertentu. Dokumen digital merupakan dokumen yang dapat dibaca melalui perangkat eletronik tanpa adanya kertas fisik. Dokumen PDF telah menjadi standar dunia sebagai dokumen digital. Maka dari itu, diperlukan suatu metode yang dapat memastikan keamanan pengiriman dan integritas dari sebuah dokumen PDF sehingga tidak menimbulkan masalah yang merugikan bagi pihak pengirim maupun penerima dokumen PDF.

Penelitian ini menggunakan algoritma SHA-256 dan algoritma RSA untuk membuat tanda tangan digital yang dapat mendeteksi perubahan perubahan pada dokumen PDF. Algoritma SHA-256 digunakan untuk mengambil nilai *hash* SHA-256 dari dokumen PDF sementara algoritma RSA digunakan untuk mengenkripsi dan dekripsi nilai *hash* tersebut. Berdasarkan pengujian yang telah dilakukan pada penelitian ini, algoritma *hash* SHA-256 dan algoritma RSA dapat mendeteksi hampir seluruh perbedaan dari dokumen PDF termasuk mendeteksi perbedaan dokumen PDF yang dibuat dari *file* DOCX yang sama tetapi menggunakan *renderer* PDF yang berbeda atau dibuat menggunakan *renderer* PDF yang sama tetapi pada waktu yang berbeda. Meskipun demikian, algoritma *hash* SHA-256 dan algoritma RSA tidak mampu mendeteksi perbedaan dari nama *file* PDF.

Kata kunci: RSA, SHA-256, tanda tangan digital, PDF

ABSTRACT

Document is a collection of written or recorded information that is systematically organized for a specific purpose. Digital documents are documents that can be read through electronic devices without physical paper. PDF documents have become the world standard as digital documents. Therefore, a method is needed that can ensure the security of delivery and integrity of a PDF document so that it does not cause problems that are detrimental to the sender and recipient of the PDF document.

This research uses the SHA-256 algorithm and RSA algorithm to create a digital signature that can detect changes in PDF documents. The SHA-256 algorithm is used to retrieve the SHA-256 hash value of the PDF document while the RSA algorithm is used to encrypt and decrypt the hash value. Based on the tests conducted in this research, the SHA-256 hash algorithm and RSA algorithm can detect almost all differences in PDF documents including detecting differences in PDF documents created from the same DOCX file but using different PDF renderers or created using the same PDF renderer but at different times. However, the SHA-256 hash algorithm and RSA algorithm are not able to detect differences in PDF file names.

Keywords: RSA, SHA-256, digital signature, PDF

KATA PENGANTAR

Dengan memanjatkan puja dan puji Syukur kehadirat Allah SWT yang telah melimpahkan rahmat, taufik, dan hidayah-Nya, sehingga penulis dapat menyelesaikan laporan tugas akhir dengan judul "Implementasi Secure *Hash* Algorithm 256 (SHA-256) dan Algoritma Rivest Shamir Adleman (RSA) untuk Mendeteksi Perubahan Perubahan pada Dokumen PDF" sebagai salah satu syarat untuk menyelesaikan Program Sarjana (S1) di Program Studi Informatika Fakultas Teknik Industri Universitas Pembangunan Nasional "Veteran" Yogyakarta. Penulis menyadari bahwa tugas akhir ini tidak mungkin terselesaikan tanpa dukungan dan bantuan dari berbagai pihak selama penyusunan tugas akhir ini. Pada kesempatan ini penulis mengucapkan terima kasih dengan tulus kepada:

1. Allah SWT, yang senantiasa memberikan kemudahan, kebaikan, dan kasih sayang kepada penulis sehingga dapat mencapai tahap ini.

2. Orang tua dan keluarga, yang selalu memberikan dukungan moril dan materil serta doa dalam menyelesaikan tugas akhir ini.

3. Rifki Indra Perwira, S.Kom., M.Eng. selaku dosen pembimbing tugas akhir yang telah memberikan bimbingan dan arahan untuk penyusunan tugas akhir ini.

4. Bapak Andhiko Putro Suryotomo, S.Kom., M.Cs., Bapak Rochmat Husaini, S.Kom., M.Kom., dan Bapak Bambang Yuwono, S.T., M.T., selaku dosen penguji tugas akhir yang telah memberikan saran dan masukan pada tugas akhir ini.

5. Seluruh rekan yang sudah menemani selama pelaksanaan tugas akhir ini.

Dalam penulisan tugas akhir ini tidak lepas dari kekurangan dan kesalahan. Oleh karena itu penulis mengharapkan kritik dan saran yang dapat menyempurnakan penulisan tugas akhir ini serta dapat bermanfaat bagi semua pihak dan para pembaca.

Yogyakarta, 21 Agustus 2024

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I

PENDAHULUAN

1.1. Latar Belakang

Dokumen merupakan kumpulan informasi tertulis atau rekaman yang disusun secara sistematis untuk tujuan tertentu. Dokumen fisik adalah dokumen yang berbentuk materi fisik atau konkret, seperti kertas. Dokumen digital merupakan dokumen yang dapat dibaca melalui perangkat eletronik tanpa adanya kertas fisik. Dokumen digital memiliki fungsi yang sama dengan dokumen fisik (Aryasanti et al., 2022). Dokumen digital dapat disimpan dalam format yang dapat dibaca oleh komputer, seperti Portable Document Format (PDF) dan Office Open XML Document (DOCX).

Dokumen PDF telah menjadi standar dunia sebagai dokumen digital (Refialy et al., 2015). Maka dari itu, diperlukan suatu metode yang dapat memastikan integritas dari sebuah dokumen PDF sehingga tidak menimbulkan masalah yang merugikan bagi pihak pengirim maupun penerima dokumen PDF.

Sistem kriptografi bisa digunakan untuk memeriksa masalah keaslian suatu dokumen (Aryasanti et al., 2022). Salah satu teknik kriptografi yang dapat dimanfaatkan yaitu tanda tangan digital. Tanda tangan digital berbeda dengan tanda tangan fisik yang di-*scan* atau difoto. Tanda tangan digital sebenarnya adalah suatu nilai kriptografis yang bergantung pada pesan dan pengirim pesan sehingga dengan adanya tanda tangan digital, otentikasi, dan identitas penulis pesan secara konseptual dapat terjamin (Putri & Manullang, 2018).

Tanda tangan digital dapat dibuat dengan menggunakan gabungan algoritma kriptografi asimetris dan algoritma *hash*. Pembuatan tanda tangan digital dimulai dengan proses mendapatkan ringkasan isi dokumen dengan menggunakan algoritma *hash*. Setelah itu, isi ringkasan dokumen akan dienkripsi menggunakan kunci privat dari algoritma kriptografi asimetris (Nuraeni et al., 2018). Hasil enkripsi kemudian akan dikirimkan bersamaan dengan dokumen PDF sebagai sebuah bukti keaslian dokumen PDF.

Algoritma *hash* atau *message digest* ini biasanya digunakan untuk mengambil sidik jari suatu pesan (Saputra & Nasution, 2019). Dalam pembuatan tanda tangan digital, algoritma *hash* digunakan untuk meringkas isi dokumen menjadi sebuah nilai *hash* yang unik.

Algoritma kriptografi asimetri didesain sedemikian sehingga kunci yang digunakan untuk enkripsi berbeda dari kunci yang digunakan untuk dekripsi. Fungsi utama dari algoritma kriptografi asimetri adalah *non-repudiation* atau anti penyangkalan. Dimana apabila dokumen berhasil diverifikasi maka pengirim tidak bisa menyangkal bahwa keberadaan dokumen benar dikirim oleh pengirim yang bersangkutan (Cahyo Prabowo & Afrianto, 2017).

Penelitian tanda tangan digital untuk mendeteksi perubahan suatu dokumen digital sudah pernah dilakukan sebelumnya. (Andika et al., 2021) melakukan penelitian untuk menguji integritas dari suatu dokumen digital. Hasil dari pengujian integritas pada

penelitian ini yaitu jika salah satu baik dokumen maupun tanda tangan digital dipalsukan atau diubah maka hasil pengujiannya adalah tidak valid.

Selain itu, (Fachrul et al., 2022) telah melakukan penelitian tentang memverfikasi keaslian dokumen PDF menggunakan SHA-256 dan algoritma Rivest Shamir Adleman (RSA). Penelitian ini juga menguji keaslian dokumen PDF dengan cara mengecek validitas tanda tangan digital yang telah dibuat. Pengujian dilakukan dengan mengecek dokumen PDF yang telah dimodifikasi dan yang tidak modifikasi. Penelitian tersebut menghasilkan kesimpulan bahwa modifikasi isi dokumen PDF dapat membuat hasil verifikasi tanda tangan menjadi tidak valid.

Meskipun demikian, kedua penelitian tersebut masih dapat dilanjutkan dalam hal mendeteksi perbedaan pada setiap dokumen PDF. Pengujian lanjutan yang dapat dilakukan yaitu penambahan skenario dalam modifikasi dokumen PDF, perbandingan tanda tangan digital dari beberapa *renderer* PDF, dan perbandingan tanda tangan digital dari beberapa *renderer* PDF dalam tiap iterasi.

Maka dari itu, penelitian kali ini ingin menguji seberapa jauh kemampuan SHA-256 dan algoritma RSA dalam mendeteksi perbedaan pada dokumen PDF dengan menambahkan beberapa pengujian lanjutan.

1.2. Rumusan Masalah

Berdasarkan uraian latar belakang, dapat dirumuskan masalah masalah berikut:

1. Bagaimana cara tanda tangan digital dapat mendeteksi perbedaan pada dokumen PDF?
2. Bagaimana hasil pengujian algoritma *hash* SHA-256 dan algoritma RSA dalam mendeteksi perbedaan pada dokumen PDF?

1.3. Batasan Masalah

Masalah yang telah dirumuskan sebelumnya akan dibuat terfokus dengan melakukan pembatasan. Dengan adanya pembatasan masalah ini, dapat mencegah penelitian keluar dari tujuan dan konteks awal. Adapun batasan masalah dalam penelitian ini adalah dalam lingkup sebagai berikut:

1. Jenis *file* yang diuji hanya berupa *file* PDF
2. Kunci algoritma RSA yang digunakan berformat Privacy Enhanced Mail (PEM)
3. Aplikasi hex editor menggunakan HxD
4. Pengujian perubahan dokumen PDF menggunakan aplikasi Microsoft Word 2016, Nitro Pro 11, foxit.com, ilovepdf.com, dan smallpdf.com

1.4. Tujuan Penelitian

Berdasarkan masalah yang diangkat maka penelitian ini bertujuan sebagai berikut:

1. Mengidentifikasi cara tanda tangan digital dalam mendeteksi perbedaan pada dokumen PDF
2. Menguji kemampuan algoritma *hash* SHA-256 dan algoritma RSA dalam mendeteksi perbedaan pada dokumen PDF

1.5. Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini adalah sebagai berikut:

1. Mengetahui hasil pengujian algoritma *hash* SHA-256 dan algoritma RSA dalam mendeteksi perbedaan pada dokumen PDF
2. Menjadi referensi bagi peneliti selanjutnya untuk dapat mengembangkan metode metode verifikasi keaslian dokumen digital lainnnya.

1.6. Tahapan Penelitian

Penelitian ini melibatkan serangkaian tahapan yang mencakup beberapa langkah penting dalam proses penelitian. Tahapan-tahapan tersebut dirinci sebagai berikut:

1. Rencana Penelitian
   a. Identifikasi masalah
      Pada tahap ini, penulis melakukan identifikasi terhadap masalah yang akan diselesaikan dalam penelitian ini. Masalah tersebut meliputi ancaman pemalsuan dokumen digital terutama dokumen PDF. Dokumen digital yang dikirimkan melalui internet berpotensi untuk dipalsukan oleh pihak pihak tak bertanggungjawab. Hal ini mengakibatkan sulitnya memercayai integritas dari dokumen PDF yang diterima.
   b. Studi literatur
      Studi literatur dilakukan untuk mencari teori sebagai landasan memecahkan masalah penelitian dan mendukung penelitian. Sumber yang digunakan berasal dari artikel dan jurnal ilmiah tentang metode metode yang berkaitan dengan tanda tangan digital pada *file* PDF. Melalui tahap ini, penulis berharap dapat menemukan solusi terbaik berdasarkan penelitian yang telah dilakukan sebelumnya.
   c. Pembuatan tanda tangan digital
      Tanda tangan digital dibuat dengan mendapatkan nilai *hash* pada *file* PDF menggunakan algoritma *hash* SHA-256. Kemudian melakukan pembangkitan kunci privat dan publik algoritma RSA. Setelah itu, nilai *hash* yang telah didapatkan sebelumnya akan dienkripsi menggunakan kunci privat dari algoritma RSA dan akan dikirimkan bersama dengan *file* PDF sebagai sebuah tanda tangan digital.

d. Verifikasi tanda tangan digital
   Tanda tangan digital yang telah dikirimkan bersama *file* PDF akan didekripsi menggunakan kunci publik algoritma RSA sehingga didapatkan nilai *hash* dari tanda tangan digital tersebut. Kemudian nilai *hash* dari *file* PDF yang dikirimkan akan dihitung menggunakan algoritma *hash* SHA-256. Terakhir, nilai *hash* hasil dekripsi dari tanda tangan digital akan dicocokkan dengan nilai *hash* yang didapat dari hasil hitung algoritma *hash* SHA-256. Jika nilai *hash* dekripsi tanda tangan digital sama dengan nilai *hash* yang didapat dari algoritma *hash* maka *file* PDF tersebut tidak mengalami perubahan pada isinya. Sebaliknya, jika tidak sama maka *file* PDF tersebut sudah mengalami perubahan pada isinya.
e. Pengujian tanda tangan digital
   Tanda tangan digital yang dibuat menggunakan algoritma SHA-256 dan RSA akan diuji dengan beberapa skenario pengujian. Hal ini dilakukan untuk mengetahui kemampuan SHA-256 dan RSA dalam memverifikasi keaslian *file* PDF
f. Hasil Penelitian
   Hasil penelitian ini merupakan hasil dari skenario skenario pengujian yang telah dilakukan sebelumnya

2. Pengembangan Sistem
   Penelitian ini menggunakan metode pengembangan sistem *waterfall*.
3. Pengujian Sistem
   Pada penelitian ini, sistem yang telah dibuat akan diuji fungsionalitasnya sesuai dengan yang diharapkan pada saat tahap pengembangan sistem.

1.7. Sistematika Penulisan

Rencana penulisan Tugas Akhir ini akan dilakukan secara sistematik, Adapun sistematika penulisan laporan sebagai berikut:

BAB I PENDAHULUAN

Bab ini membahas mengenai latar belakang mengapa penelitian dilakukan, rumusan masalah yang ada, pembatasan masalah agar penelitian terfokus, tahapan penelitian yang akan dilakukan, serta sistematika penulisan

BAB II TINJAUAN LITERATUR

Bab ini membahas mengenai dasar teori yang mendukung penelitian dilakukan, metode yang akan digunakan, dan pustaka ilmiah terkait penelitian-penelitian sebelumnya.

BAB III METODOLOGI PENELITAN

Bab ini membahas mengenai langkah, dan penjelasan detail mengenai cara kerja metode yang digunakan untuk menyelesaikan masalah pada penelitian ini

BAB IV HASIL DAN PEMBAHASAN

Bab ini membahas mengenai langkah, dan penjelasan detail mengenai cara kerja metode yang digunakan untuk menyelesaikan masalah pada penelitian ini

BAB V PENUTUP

Bab ini membahas kesimpulan, saran, dan keberlanjutan terkait dengan tujuan penelitian yang telah dijabarkan sebelumnya. Hasil kesimpulan yang telah dijelaskan, kemudian dievaluasi untuk penjabaran saran sehingga penelitian selanjutnya dapat lebih baik.

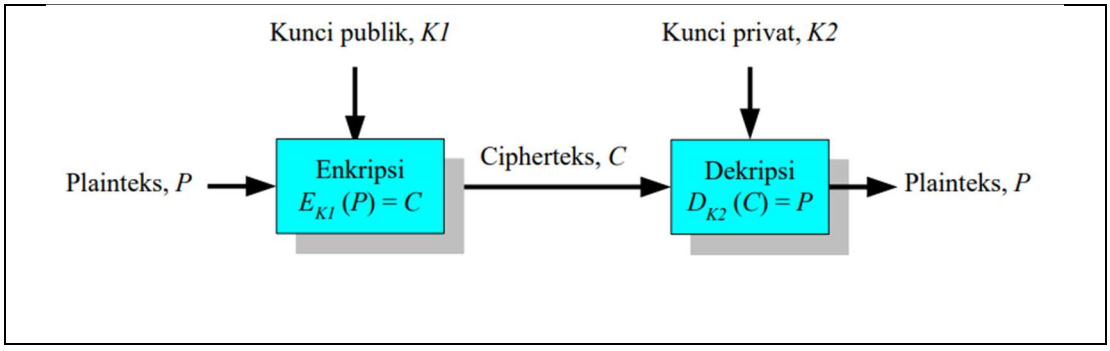
BAB II
TINJAUAN LITERATUR

2.1. Dokumen PDF

PDF (Portable Document Format) adalah format *file* yang dibuat oleh Adobe Systems pada tahun 1993 untuk bertukar dokumen digital. Dokumen PDF telah menjadi standar dunia sebagai dokumen digital (Refialy et al., 2015). Format PDF digunakan untuk menyajikan dokumen dua dimensi yang berisi teks, huruf, gambar, dan grafik. Format PDF adalah format dokumen yang sangat populer karena tidak bergantung pada perangkat lunak.

2.2. Kriptografi Asimetris

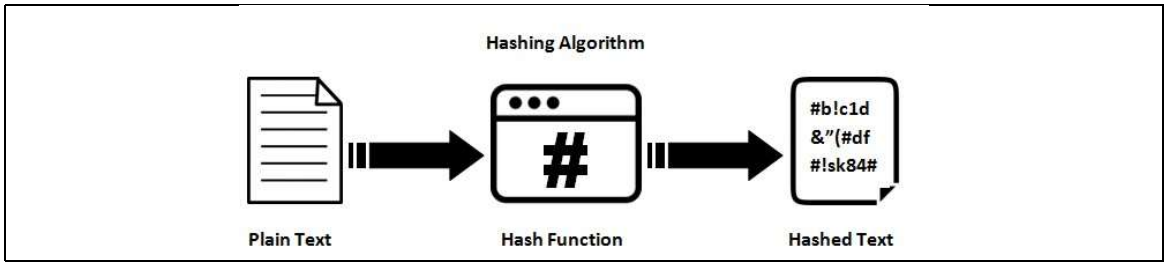
Kriptografi asimetris atau kriptografi kunci publik merupakan algoritma kritografi yang kunci untuk melakukan enkripsi berbeda dengan kunci untuk melakukan dekripsi. Kriptografi asimetris membuat dunia kriptografi menjadi lebih aman dengan memanfaatkan dua pasang kunci (kunci publik dan kunci privat) yang saling terkait secara matematis untuk melakukan enkripsi dan dekripsi (Maqsood et al, 2017). Proses penggunaan kriptografi asimetris dimulai dengan membangkitkan dua pasang kunci. Kunci privat disimpan dengan aman oleh pemilik kunci sementara kunci publik dapat dibagikan ke pihak pihak lain.



Gambar 2.1 Proses enkripsi dan dekripsi kriptografi asimetri (Munir, 2024)

2.3. Algoritma *Hash*

Algoritma *hash* atau fungsi *hash* adalah fungsi yang akan menerima pesan dengan panjang sembarang dan melakukan kompresi menjadi pesan acak dengan panjang yang tetap (Dermawan Ikhsan, 2021). Hasil dari algoritma *hash* disebut dengan *message digest* atau nilai *hash*. Nilai *hash* yang dihasilkan memiliki panjang yang berbeda beda bergantung dengan algoritma *hash* yang digunakan. Skema proses *hash*ing dari suatu algoritma *hash* dijelaskan pada gambar dibawah ini.



Gambar 2.2 Proses mendapatkan *message digest* atau nilai *hash* (Iwan, 2018)

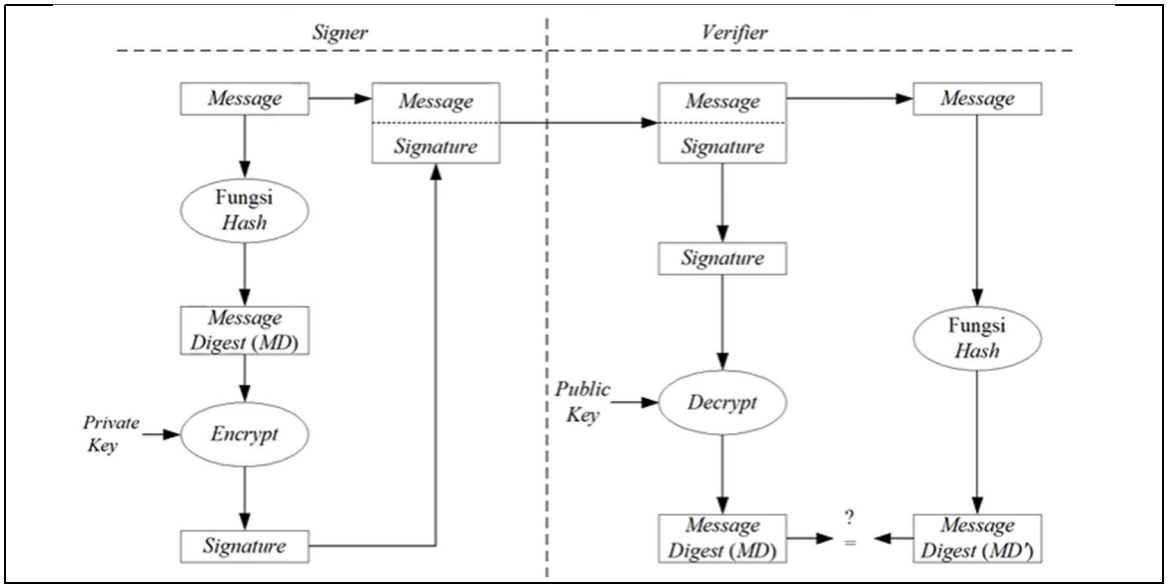
2.4. Tanda Tangan Digital

Tanda tangan digital adalah suatu nilai kriptografis yang bergantung pada pesan dan pengirim pesan sehingga dengan adanya tanda tangan digital, otentikasi dan identitas penulis pesan secara konseptual dapat terjamin (Putri & Manullang, 2018). Sedikit saja perubahan pada dokumen akan menghasilkan perubahan tanda tangan digital secara drastis.

Tanda tangan digital memiliki empat karakteristik utama pada aspek keamanan kriptografi yaitu integritas (integrity), kerahasiaan (confidentiality), otentikasi (autentication), dan anti-penyangkalan (non-repudiation) (Zaatsiyah & Djuniadi, 2021). Pengirim tidak dapat menyangkal telah mengirim dokumen tersebut jika tanda tangan pada dokumen tersebut valid ketika diverifikasi oleh penerima.

Proses tanda tangan digital melibatkan penerapan algoritma kriptografi asimetris dan algoritma *hash*. Dokumen yang hendak dikirim terlebih dahulu akan dilakukan *hash*ing menggunakan algoritma *hash* sehingga menjadi bentuk yang ringkas yang disebut dengan *message digest*. Kemudian, *message digest* dienkripsi menggunakan algoritma kriptografi asimetris. Kunci privat milik pengirim dokumen akan digunakan untuk melakukan enkripsi *message digest* (Anshori & Erwin Dodu, 2019). Hasil dari enkripsi inilah yang disebut sebagai tanda tangan digital.

Proses utama pada tanda tangan digital terdiri atas dua proses, yaitu proses penandatanganan dan verifikasi (Anshori & Erwin Dodu, 2019). Proses penandatangan dilakukan dengan mengubah sebuah isi dokumen menjadi *message digest* dan mengenkripsinya menggunakan algoritma kriptografi asimetris. Sementara, proses verifikasi dilakukan dengan membandingkan hasil dekripsi *message digest* dokumen yang diterima dengan *message digest* dari dokumen sebenarnya. Jika hasilnya cocok maka dokumen tersebut masih asli dan tidak mengalami perubahan. Akan tetapi, jika hasilnya tidak cocok maka dokumen tersebut sudah tidak asli dan telah mengalami perubahan.



Gambar 2.3 Proses pembentukan dan verifikasi tanda tangan digital (Munir, 2022)

2.5. Secure *Hash* Algorithm 256

Secure *Hash* Algorithm 256 (SHA-256) adalah salah satu dari keluarga kriptografi *hash* SHA yang di desain oleh United State NSA pada tahun 2001 dan dipatenkan di bawah US patent 6829355 (Bhargav et al., 2015). SHA-256 merupakan sebuah algoritma *hash* yang menghasilkan nilai *hash* dengan panjang 256 bit (Endelina, 2021). Nilai *hash* sepanjang 256 bit tersebut biasanya akan direpresentasikan dalam hex. Langkah-langkah dalam menerapkan SHA-256 adalah sebagai berikut:

1) Preprocessing
   a) Membagi pesan menjadi blok blok sebesar 512 bit,
   b) Menambahkan bit padding pada blok terakhir pesan hingga mencapai 448 bit. Bit padding diawali dengan angka 1 kemudian angka 0 hingga tercapai 448 bit,
   c) Menambahkan Representasi panjang pesan pada 64 bit terakhir

2) Processing
   a) Parsing penguraian pesan pada setiap 512 bit blok menjadi 16 blok berisi 32 bit,
   b) Ekspansi pesan yang awalnya 16 blok menjadi 64 blok yang nantinya akan dijadikan untuk penjadwalan pesan. Langkah ekspansi dilakukan dari blok ke-17 sampai blok ke-64 dengan melibatkan operasi bitwise, rotasi, dan operasi non-linier. Persamaan matematis dalam proses ekspansi pesan ditunjukkan pada Persamaan berikut

$$W_t = \begin{cases} M_t \ 0 \ \leq t \ \leq 15 \\ \sigma_1(W_{t-2}) + (W_{t-7}) + \sigma_0(W_{t-15}) + (W_{t-16}) \ \ 16 \ \leq t \ \leq 63 \end{cases} \text{....(2.1)}$$

Keterangan :

W : blok pesan

σ_0 x:operasi bitwise ((w[x] →7) ⊕ (w[x] → 18) ⊕ (w[x] >> 3))

σ_1 x:operasi bitwise ((w[x] →17) ⊕ (w[x] → 19) ⊕ (w[x] >> 10))

→ : operasi bitwise rotate

⊕: operasi bitwise xor

≫ : operasi bitwise right shift

+ : Penjumlahan nilai module 2^{32}

c) Inisialisasi variabel awal nilai *hash* yang telah ditetapkan pada SHA-256.

Tabel 2.1 Inisialisasi variabel awal nilai *hash*

H0	6A09E667	H4	510E527F
H1	BB67AE85	H5	9B05688C
H2	3C6EF372	H6	1F83D9AB
H3	A54FF53A	H7	5BE0CD19

d) Inisialisasi konstanta yang telah ditetapkan pada algoritma *hash* SHA-256. Terdapat 64 nilai konstanta berbeda yang digunakan. Setiap iterasi akan menggunakan nilai konstanta yang berbeda beda. Konstanta dari ketetapan SHA-256 adalah sebagai berikut:

Tabel 2.2 Inisialisasi nilai konstanta

K[0]	428a2f98	K[16]	49b69c1	K[32]	27b70a85	K[48]	19a4c116
K[1]	71374491	K[17]	efbe4786	K[33]	2e1b2138	K[49]	1e376c08
K[2]	b5c0fbcf	K[18]	0fc19dc6	K[34]	4d2c6dfc	K[50]	2748774c
K[3]	9b5dba5	K[19]	240ca1cc	K[35]	53380d13	K[51]	34b0bcb5
K[4]	3956c25b	K[20]	2de92c6f	K[36]	650a7354	K[52]	391c0cb3
K[5]	59f111f1	K[21]	4a7484aa	K[37]	766a0abb	K[53]	4ed8aa4a
K[6]	923f82a4	K[22]	5cb0a9dc	K[38]	81c2c92e	K[54]	5b9cca4f
K[7]	ab1c5ed5	K[23]	76f988da	K[39]	92722c85	K[55]	682e6ff3
K[8]	d807aa98	K[24]	983e5152	K[40]	a2bfe8a1	K[56]	748f82ee
K[9]	12835b01	K[25]	a831c66d	K[41]	a81a664b	K[57]	78a5636f
K[10]	243185be	K[26]	b00327c8	K[42]	c24b8b70	K[58]	84c87814
K[11]	550c7dc3	K[27]	bf597fc7	K[43]	c76c51a3	K[59]	8cc70208
K[12]	72be5d74	K[28]	c6e00bf3	K[44]	d192e819	K[60]	90befffa
K[13]	80deb1fe	K[29]	d5a79147	K[45]	d6990624	K[61]	a4506ceb
K[14]	9bdc06a7	K[30]	06ca6351	K[46]	f40e3585	K[62]	bef9a3f7
K[15]	19bf174	K[31]	14292967	K[47]	106aa070	K[63]	c67178f2

e) Pemrosesan pesan

Proses pembuatan *message digest* melakukan 64 putaran atau iterasi sesuai dengan jumlah konstanta. Perhitungan digest dapat dilihat melalui pseudocode berikut:

```
a = H0
b = H1
c = H2
d = H3
e = H4
f = H5
g = H6
h = H7

for (let t = 0; t < 64; t++) {
  T1 = (h + Σ1(e) + Ch(e, f, g) + K[t] + W[t]);
  T2 = (Σ0(a) + Maj(a, b, c));
  h = g;
  g = f;
  f = e;
  e = (d + T1);
  d = c;
  c = b;
  b = a;
  a = (T1 + T2);
}
```

Gambar 2.4 Pseudocode pemrosesan pesan SHA-256

Keterangan:

$Ch(e,f,g) = (e\&f) \oplus (\neg e\&g)$

$Maj(e,f,g) = (e\&f) \oplus (e\&g) \oplus (f\&g)$

$\sum_0 x = (x \rightarrow 2) \oplus (x \rightarrow 13) \oplus (x \rightarrow 22)$

$\sum_1 x = (x \rightarrow 6) \oplus (x \rightarrow 11) \oplus (x \rightarrow 25)$

K_t= konstanta ke t

W_t= blok pesan ke t

3) Value

Setelah melalui pemrosesan pesan, didapatkan 8 blok 32 bit dari iterasi terakhir yaitu a,b,c…h. Kemudian dilakukan penambahan terhadap masing masing blok tersebut dengan variabel awal nilai *hash* yang telah ditetapkan pada SHA-256 sesuai dengan persamaan berikut:

$H0 = H0 + a$

$H1 = H1 + b$

$H2 = H2 + c$

$H3 = H3 + d$

$H4 = H4 + e$
$H5 = H5 + f$
$H6 = H6 + g$
$H7 = H7 + h$

Nilai akhir *hash* SHA-256 merupakan gabungan dari H0,H1,H2,...H7.

2.6. Algoritma Rivest Shamir Adleman

Algoritma RSA diciptakan oleh tiga orang peneliti yaitu Ron Rivest, Adi Shamir, dan Leonard Adleman dari Massachussets Institute of Technology pada tahun 1978 (Rangkuti et al., 2019). Algoritma Rivest Shamir Adleman (RSA) merupakan salah satu algoritma kritografi asimetris. Algoritma RSA biasa digunakan untuk pertukaran kunci dan penandatanganan digital (Hutagalung et al., 2023). Algoritma RSA didasarkan penggunaan sifat matematika dari bilangan prima dan eksponensiasi modular untuk menciptakan pasangan kunci kriptografi yang digunakan dalam proses enkripsi dan dekripsi. Algoritma RSA memanfaatkan tingkat kerumitan dalam memfaktorkan dua bilangan prima yang sangat besar untuk menjaga kerahasiaan data (Rivest et al., 1978). Berikut merupakan langkah langkah dalam menerapkan algoritma RSA.

1) Pembangkitan Kunci
   a) Memilih dua bilangan prima (p dan q),
   b) Mendapatkan nilai modulus
      $n = p \times q$ …….....(2.3)
   c) Mendapatkan nilai totien ϕ(n)
      $\phi(n) = (p - 1) \times (q - 1)$ ……...(2.4)
   d) Memilih bilangan e yang relatif prima terhadap ϕ(n) dan 1 < e < ϕ(n). Sebuah bilangan dikatakan relatif prima terhadap bilangan lainnya jika Greatest Common Divisor (GCD) dari kedua bilangan tersebut adalah 1. Algoritma euclidean dapat digunakan untuk mencari GCD dari dua buah bilangan. Algoritma euclidean untuk GCD(A,B) adalah sebagai berikut:
      - Menulis dalam persamaan $A = B \times Q + R$
      - Mencari nilai $GCD(B, R)$, karena $GCD(A, B) = GCD(B, R)$
      - Jika ditemukan R = 0, maka nilai R sebelum nya merupakan merupakan nilai $GCD(A, B)$
   e) Hitung d sebagai invers modular dari e modulo $\phi(n)$
      $e \times d \equiv 1 \, mod \, (\phi(n))$ ……..……....……....(2.5)
      persamaan tersebut ekivalen dengan
      $e \times d = 1 + k \, . \phi(n)$ ……(2.6)

sehingga

$d = \frac{1+k \cdot \phi(n)}{e}$ ……..(2.7)

Dengan k = 1,2,3,4….n, hingga didapat nilai d yang bulat

f) Didapatkan kunci publik (e,n) dan kunci privat (d,n)

2) Enkripsi

Enkripsi pada algoritma RSA menggunakan persamaan sebagai berikut

$c = m^e \ mod \ n$ …….(2.8)

3) Dekripsi

Dekripsi pada algoritma RSA menggunakan persamaan sebagai berikut

$m = c^d \ mod \ n$ …….(2.9)

Keterangan :

p = bilangan prima acak

q = bilangan prima acak

c = cipher teks

m = plain teks

e = kunci enkripsi

d = kunci dekripsi

n = modulo

2.7. Penelitian Sebelumnya

Beberapa penelitian terkait dengan pembuatan tanda tangan digital untuk memverifikasi dokumen elektronik telah dilakukan sebelumnya. Penelitian penelitian tersebut telah dirangkum dalam tabel *state of the art* berikut:

Tabel 2.3 State of the Art

No	Penulis	Judul	Metode	Hasil
1	Fachrul et al., 2022	Penerapan Konsep Digital Signature Terhadap Verifikasi Keaslian Dokumen Transkrip Nilai Mahasiswa Menggunakan Enkripsi Rivest Shamir Adleman	RSA dan SHA-256	Tanda tangan digital yang dibuat menggunakan algoritma RSA dan SHA-256 dapat digunakan untuk memverifikasi keaslian dokumen digital seperti PDF
2	Andika et al., 202	Aplikasi Enterprise Document Digital Signature menggunakan RSA dan SHA256 untuk WFH di Era Pandemi COVID-19	RSA dan SHA-256	Hasil dari penelitian ini menunjukan melalui uji integritas, verifier dapat mendeteksi jika dokumen

No	Penulis	Judul	Metode	Hasil
				PDF atau tanda tangan telah dipalsukan.
3	Arisandi et al., 2020	Pemeriksaan Integritas Dokumen Dengan Digital Signature Algorithm	Digital Signature Algorithm (DSA)	Hasil dari penelitian ini menunjukan algoritma DSA berhasil mengidentifikasi perubahan dari *key* yang digunakan
4	Cahyo Prabowo & Afrianto, 2017	Penerapan Digital Signature Dan Kriptografi Pada Otentikasi Sertifikat Tanah Digital	RSA dan SHA-256	Hasil dari penelitian ini menunjukan bahwa tanda tangan digital dapat melacak perubahan yang terjadi jika dokumen PDF diubah maupun kunci yang digunakan tidak sesuai
5	Hutagalung et al., 2023	Keamanan Data Menggunakan Secure *Hash*ing Algorithm (SHA)-256 dan Rivest Shamir Adleman (RSA) pada Digital Signature	RSA dan SHA-256	Hasil dari penelitian ini menunjukan bahwa tanda tangan digital yang dibuat menggunakan algoritma RSA dan SHA-256 dapat memverifikasi keaslian Surat Keterangan Lulus (SKL)
6	Rangkuti et al., 2019	Implementasi Digital Signature Pada E-Invoice Di Uniqa Digital Invitation Menggunakan Algoritma SHA-256 (Secure *Hash* Algorithm-256) Dan RSA (Rivest Shamir Adleman)	RSA dan SHA-256	Penelitian ini meyinpulkan algoritma SHA-256 dan RSA dapat diterapkan dalam menjaga keaslian data e-invoice dengan hasil digital siganture yang menghasilkan kode acak yang rumit.
7	Endelina, 2021	Implementasi Digital Signature Pada *File* Audio Menerapkan Metode SHA-256	SHA-256	Penelitian menghasilkan kesimpulan bahwa metode SHA 256 dapat mendeteksi keaslian dari *file* audio yang digunakan
8	Az-Zahra et al., 2024	Implementasi QR Code dengan Algoritma Secure *Hash* Algorithm (SHA)-256 dan Rivest Shamir Adleman (RSA) yang Ditingkatkan untuk Autentikasi Dokumen Digital	RSA dan SHA 256	Hasil dari penelitian ini menunjukan bahwa algoritma RSA dan SHA-256 dapat dimanfaatkan untuk proses autentikasi dokumen digital dengan menggunakan QR Code

No	Penulis	Judul	Metode	Hasil
9	Saputra & Nasution, 2022	Perbandingan Performa Algoritma Md5 Dan Sha-256 Dalam Membangkitkan Identitas *File*	MD5 dan SHA-256	algoritma MD5 memiliki performa yang lebih unggul dibandingkan dengan algoritma SHA-256 dalam hal penggunaan prosesor, penggunaan memori dan waktu yang dibutuhkan untuk membangkitkan identitas *file*.
10	Nuraeni et al., 2018	Implementasi Tanda Tangan Digital Menggunakan RSA dan SHA-512 Pada Proses Legalisasi Ijazah	RSA dan SHA-512	Penerapan Algoritma RSA dan SHA-512 pada digital signature telah teruji aman, hal tersebut terbukti melalui proses pengujian dimana dilakukan modifikasi data pada dokumen legalisasi dan program berhasil mengetahuinya.

Berdasarkan hasil penelitian penelitian sebelumnya, penelitian ini menggunakan metode yang dirasa efektif dalam membuat dan memverifikasi tanda tangan digital. Tinjauan penelitian ini menunjukan bahwa penelitian ini memiliki beberapa kesamaan dengan penelitian penelitian sebelumnya.

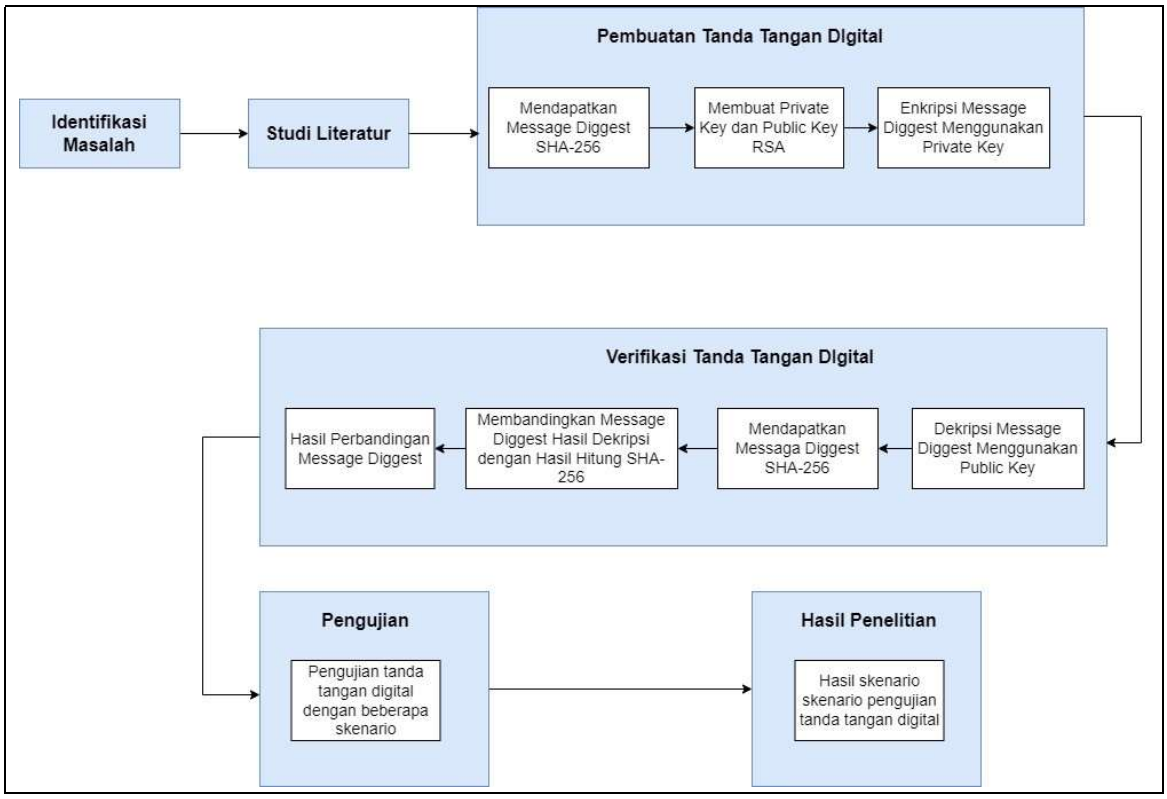
Penelitian ini akan melanjutkan skenario pengujian dari penelitian sebelumnya. Pengujian lanjutan yang akan dilakukan yaitu penambahan skenario dalam modifikasi dokumen PDF, perbandingan tanda tangan digital dari beberapa *renderer* PDF, dan perbandingan tanda tangan digital dari beberapa *renderer* PDF dalam tiap iterasi.

Pengujian lanjutan tersebut dilakukan untuk melihat seberapa mampu tanda tangan yang dihasilkan dari algoritma SHA-256 dan RSA dalam mendeteksi perbedaan yang ada pada dokumen PDF.

BAB III
METODOLOGI PENELITIAN

3.1. Metodologi Penelitian

Penelitian ini menggunakan metode kuantitatif. Metode kuantitatif digunakan dalam penelitian yang hasilnya dapat terukur sehingga dapat menguji sebuah hipotesis. Hasil dari penelitian ini tidak didasarkan pada logika ilmiah melainkan pada hasil uji hipotesis. Alur tahapan penelitian ini dijelaskan pada gambar berikut.



Gambar 3.1 Alur Tahapan Penelitian

3.1.1. Identifikasi Masalah

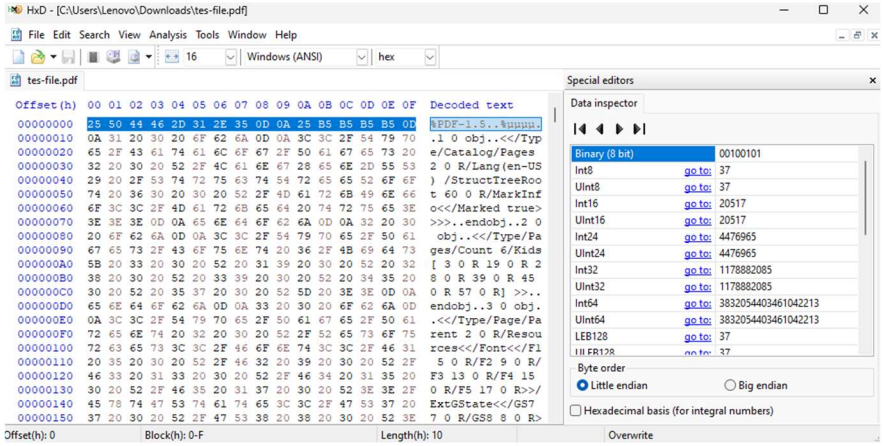
Pada tahap ini, penulis mengidentifikasi masalah yang akan diselesaikan dalam penelitian ini. Masalah tersebut meliputi ancaman pemalsuan dokumen digital terutama dokumen PDF. Dokumen digital yang dikirimkan melalui internet berpotensi untuk dipalsukan oleh pihak pihak tak bertanggungjawab. Hal ini mengakibatkan sulitnya memercayai integritas dari dokumen PDF yang diterima.

3.1.2. Studi Literatur

Studi literatur dilakukan untuk mencari teori sebagai landasan memecahkan masalah penelitian dan mendukung penelitian. Sumber yang digunakan berasal dari artikel dan jurnal ilmiah tentang metode metode yang berkaitan dengan tanda tangan digital pada *file* PDF sehingga dapat memastikan keaslian dari *file* PDF itu sendiri. Berdasarkan penelitian yang telah dilakukan sebelumnya, diharapkan dapat ditemukan solusi terbaik dalam memastikan sebuah keaslian *file* PDF. Penelitian penelitian sebelumnya dapat dilihat pada tabel 2.4 State of the Art.

3.1.3. Pembuatan Tanda Tangan Digital

Pembuatan Tanda tangan digital menggunakan sampel dokumen pdf bernama tes-*file*.pdf dengan hanya mengambil 128 bit awalnya saja.



Gambar 3.2 Sampel *file* pdf

Langkah langkah membuat tanda tangan digital berdasarkan sampel *file* tersebut adalah sebagai berikut

1. Mendapatkan *Message digest* SHA-256
   Terdapat beberapa langkah untuk mendapatkan *message digest* SHA-256 yaitu *preprocessing*, *processing*, dan *value*. Langkah langkah tersebut akan dijelaskan dalam contoh berikut
   a) Preprocessing
      1) Membagi pesan menjadi blok blok sebesar 512 bit

Tabel 3.1 Pembagian Pesan Menjadi Blok 512 Bit

00100101	01010000	01000100	01000110	00101101	00110001
00101110	00110101	00001101	00001010	00100101	10110101
10110101	10110101	10110101	00001101		

2) Menambahkan bit padding pesan hingga mencapai 448 bit

Tabel 3.2 Menambahkan Bit Padding Hingga Mencapai 448 Bit

00100101	01010000	01000100	01000110	00101101	00110001
00101110	00110101	00001101	00001010	00100101	10110101
10110101	10110101	10110101	00001101	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	00000000
00000000	00000000	00000000	00000000	00000000	00000000
00000000	00000000				

3) Menambahkan representasi panjang pesan pada 64 bit terakhir
Panjang bit pesan adalah sebanyak 288. Maka dimasukkan angka 288 dalam biner pada 64 bit terakhir.

Tabel 3.3 Menambahkan Representasi Panjang Pesan pada 64 Bit Terakhir

00100101	01010000	01000100	01000110	00101101	00110001
00101110	00110101	00001101	00001010	00100101	10110101
10110101	10110101	10110101	00001101	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	00000000
00000000	00000000	00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000	00000000	00000000
00000000	00000000	00000000	10000000		

b) Processing

1) Parsing pesan 512 bit menjadi 16 blok berisi 32 bit
Blok pesan 512 bit tersebut akan dipecah menjadi 16 blok yang masing masing berisi 32 bit

Tabel 3.4 Parsing Pesan 512 bit menjadi 16 blok berisi 32 bit

M_0	00100101010100000100010001000110
M_1	00101110011000100101111000110101
M_2	00001101000010100010010110110101
M_3	10110101101101011011010100001101

M_4	10000000000000000000000000000000
M_5	00000000000000000000000000000000
M_6	00000000000000000000000000000000
M_7	00000000000000000000000000000000
M_8	00000000000000000000000000000000
M_9	00000000000000000000000000000000
M_{10}	00000000000000000000000000000000
M_{11}	00000000000000000000000000000000
M_{12}	00000000000000000000000000000000
M_{13}	00000000000000000000000000000000
M_{14}	00000000000000000000000000000000
M_{15}	00000000000000000000000010000000

2) Ekspansi pesan yang awalnya 16 blok menjadi 64 blok
Setiap blok akan ditampilkan dalam format heksadesimal untuk memersingkat perhitungan. Nilai nilai blok baru yang akan diisi akan mengikuti rumus berikut

$$W_t = \begin{cases} M_t \; 0 \; \leq t \; \leq 15 \\ \sum_1 (W_{t-2}) + (W_{t-7}) + \sum_0 (W_{t-15}) + (W_{t-16}) \; 16 \; \leq t \; \leq 63 \end{cases}$$

W_0 sampai W_{15} diambil dari blok pesan sebelumnya. Sedangkan W_{16} sampai W_{63} dihitung menggunakan rumus diatas.

$W_{16} = \sum_1 (W_{16-2}) + (W_{16-7}) + \sum_0 (W_{16-15}) + (W_{16-1}\ )$

$\sum_1 (W_{14}) = W_{14} \rightarrow 17 \oplus W_{14} \rightarrow 19 \oplus W_{14} >> 10$

$W_{14} \rightarrow 17 = 00000000\ 00000000\ 00000000\ 00000000 \rightarrow 17$

$= 00000000\ 00000000\ 00000000\ 00000000$

$W_{14} \rightarrow 19 = 00000000\ 00000000\ 00000000\ 00000000 \rightarrow 19$

$= 00000000\ 00000000\ 00000000\ 00000000$

$W_{14} >> 10 = 00000000\ 00000000\ 00000000\ 00000000 >> 10$

$= 00000000\ 00000000\ 00000000\ 00000000$

$\sum_1 (W_{14}) = 00000000\ 00000000\ 00000000\ 00000000$

$\sum_1 (W_{14}) = 00000000$ (dalam hexadesimal)

$W_{16-7} = W_9 = 00000000$

$\sum_0 (W_1) = W_1 \rightarrow 7 \oplus W_1 \rightarrow 18 \oplus W_1 >> 3$

$W_1 \rightarrow 7 = 00101101\ 00110001\ 00101110\ 00110101 \rightarrow 7$

$W_1 \rightarrow 7 = 01101010\ 01011010\ 01100010\ 01011100$

$W_1 \rightarrow 18 = 00101101\ 00110001\ 00101110\ 00110101 \rightarrow 18$

$W_1 \rightarrow 18 = 01001011100011010100101101001100$

$W_1 >> 3 = 00101101\ 00110001\ 00101110\ 00110101 >> 3$

$W_1 >> 3$ = 00000101 10100110 00100101 11000110
$\sum_0(W_1)$ = 01101010010110100110001001011100⊕
01001011100011010100101101001100⊕
00000101101001100010010111000110
$\sum_0(W_1)$ = 00100100011100010000110011010110
$\sum_0(W_1)$ = 24710CD6

W_{16-16}= W_0 = 25504446

$W_{16} = \sum_1(W_{16-2}) + (W_{16-7}) + \sum_0(W_{16-15}) + (W_{16-16})$
W_{16} = 00000000 + 00000000 + 24710CD6 + 25504446
W_{16} = 49C1511C
Demikian seterusnya hingga mendapatkan nilai W_{63}. Nilai W_1 sampai W_{63} disajikan dalam tabel berikut

Tabel 3.5 Ekspansi Pesan Menjadi 64 blok

25504446	2D312E35	0D0A25B5	B5B5B50D	80000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000080
49C1511C	105741F4	F068B3E7	0F7674FE	CFB31C15	F4E31BC8	EDBBB368	3861AD47
C039477A	D36E173E	9AD8AB48	98FF8474	35CF03E5	204E93E9	9A04FCF6	C1A7AA75
3949CA7F	C888594A	C6B60405	62700E7C	FC3D3E80	B6877498	CA40059D	82F84E83
6D16173F	7191ACFC	E1DF0229	8565C88D	C0733ADA	094DF5D8	302E3AE5	FED77D2F
1FD056D0	8DCFEE06	64E7BDE8	7FA4109D	29770933	94B49AC7	33C74916	779DF0C9

3) Inisialisasi variabel awal nilai *hash* SHA-256
Variabel awal nilai *hash* akan diinisialisasi menggunakan nilai default dari SHA-256

Tabel 3.6 Inisialisasi variabel awal nilai *hash*

H0	6A09E667	H4	510E527F
H1	BB67AE85	H5	9B05688C
H2	3C6EF372	H6	1F83D9AB
H3	A54FF53A	H7	5BE0CD19

4) Inisialisasi konstanta SHA-256
Konstanta SHA-256 akan diinisialisasi dengan menggunakan nilai default

Tabel 3.7 Inisialisasi Nilai Konstanta

K[0]	428A2F98	K[16]	49B69C1	K[32]	27B70A85	K[48]	19A4C116
K[1]	71374491	K[17]	EFBE4786	K[33]	2E1B2138	K[49]	1E376C08
K[2]	B5C0FBCF	K[18]	0FC19DC6	K[34]	4D2C6DFC	K[50]	2748774C
K[3]	9B5DBA5	K[19]	240CA1CC	K[35]	53380D13	K[51]	34B0BCB5
K[4]	3956C25B	K[20]	2DE92C6F	K[36]	650A7354	K[52]	391C0CB3

K[5]	59F111F1	K[21]	4A7484AA	K[37]	766A0ABB	K[53]	4ED8AA4A
K[6]	923F82A4	K[22]	5CB0A9DC	K[38]	81C2C92E	K[54]	5B9CCA4F
K[7]	AB1C5ED5	K[23]	76F988DA	K[39]	92722C85	K[55]	682E6FF3
K[8]	D807AA98	K[24]	983E5152	K[40]	A2BFE8A1	K[56]	748F82EE
K[9]	12835B01	K[25]	A831C66D	K[41]	A81A664B	K[57]	78A5636F
K[10]	243185BE	K[26]	B00327C8	K[42]	C24B8B70	K[58]	84C87814
K[11]	550C7DC3	K[27]	BF597FC7	K[43]	C76C51A3	K[59]	8CC70208
K[12]	72BE5D74	K[28]	C6E00BF3	K[44]	D192E819	K[60]	90BEFFFA
K[13]	80DEB1FE	K[29]	D5A79147	K[45]	D6990624	K[61]	A4506CEB
K[14]	9BDC06A7	K[30]	06CA6351	K[46]	F40E3585	K[62]	BEF9A3F7
K[15]	19BF174	K[31]	14292967	K[47]	106AA070	K[63]	C67178F2

5) Pemrosesan pesan
Selanjutnya akan dilakukan pemrosesan pesan mengikuti arahan pseudocode berikut

```
a = H0
b = H1
c = H2
d = H3
e = H4
f = H5
g = H6
h = H7

for (let t = 0; t < 64; t++) {
  T1 = (h + Σ1(e) + Ch(e, f, g) + K[t] + W[t]);
  T2 = (Σ0(a) + Maj(a, b, c));
  h = g;
  g = f;
  f = e;
  e = (d + T1);
  d = c;
  c = b;
  b = a;
  a = (T1 + T2);
}
```

Gambar 3.3 Pseudocode Pemrosesan Pesan

Setiap variabel diambil dari initial *hash value* a=H0, b=H1,…h=H7. Kemudian dilakukan pemrosesan pesan sebanyak 64 perulangan. Berikut merupakan perhitungan pada perulangan pertama dimana t=0.

$T1 = (h + \Sigma1(e) + Ch(e, f, g) + K[0] + W[0])$

$T1 = 5BE0CD19 + 3587272B + 1F85C98C + 428A2F98 + 25504446$

$T1 = 18C831AE$

$T2 = \Sigma0(a) + Maj(a, b, c)$

$T2 = CE20B47E + 3A6FE667$

$T2 = 08909AE5$

$h = 1F83D9AB$
$g = 9B05688C$
$f = 510E527F$
$e = A54FF53A + 18C831AE = BE1826E8$
$d = 3C6EF372$
$c = BB67AE85$
$b = 6A09E667$
$a = 18C831AE + 08909AE5 = 2158CC93$

Demikian seterusnya hingga mendapatkan nilai perulangan ke 64, dimana t=63

Tabel 3.8 Hasil Pemrosesan Pesan

	a	b	c	d	e	f	g	h
init	6A09E667	BB67AE85	3C6EF372	A54FF53A	510E527F	9B05688C	1F83D9AB	5BE0CD19
t=0	2158CC93	6A09E667	BB67AE85	3C6EF372	BE1826E8	510E527F	9B05688C	1F83D9AB
t=1	3E3DD28B	2158CC93	6A09E667	BB67AE85	7F65620F	BE1826E8	510E527F	9B05688C
t=2	4134AB5A	3E3DD28B	2158CC93	6A09E667	25E5E9A8	7F65620F	BE1826E8	510E527F
t=3	50CE5D94	4134AB5A	3E3DD28B	3E3DD28B	81692EE9	25E5E9A8	7F65620F	BE1826E8
t=4	D89B6B3D	50CE5D94	4134AB5A	3E3DD28B	E7D01BE2	81692EE9	25E5E9A8	7F65620F
t=5	0D8DFE99	D89B6B3D	50CE5D94	4134AB5A	B8C87C92	E7D01BE2	81692EE9	25E5E9A8
t=6	EF71F074	0D8DFE99	D89B6B3D	50CE5D94	57C5A432	B8C87C92	E7D01BE2	81692EE9
t=7	D872930B	EF71F074	0D8DFE99	0D8DFE99	DBEC2233	57C5A432	B8C87C92	E7D01BE2
t=8	2422525D	D872930B	EF71F074	EF71F074	8B3D834A	DBEC2233	57C5A432	B8C87C92
t=9	C80FC62F	2422525D	D872930B	2422525D	96823156	8B3D834A	DBEC2233	57C5A432
t=10	3D7B81FF	C80FC62F	2422525D	D872930B	6865918F	96823156	8B3D834A	DBEC2233
t=11	4D61CA51	3D7B81FF	C80FC62F	2422525D	CB07A387	6865918F	96823156	8B3D834A
t=12	1AED4F62	4D61CA51	3D7B81FF	C80FC62F	6AA88291	CB07A387	6865918F	96823156
t=13	0B6D0EDE	1AED4F62	4D61CA51	3D7B81FF	6D7CFCC1	6AA88291	CB07A387	6865918F
t=14	5DD107DB	0B6D0EDE	1AED4F62	4D61CA51	4FCEDA96	6D7CFCC1	6AA88291	CB07A387
t=15	B04AC975	5DD107DB	0B6D0EDE	1AED4F62	340DC1E3	4FCEDA96	6D7CFCC1	6AA88291
t=16	C2BC77BE	B04AC975	5DD107DB	0B6D0EDE	B7C5D077	340DC1E3	4FCEDA96	6D7CFCC1
t=17	E9C46043	C2BC77BE	B04AC975	5DD107DB	27D1341C	B7C5D077	340DC1E3	4FCEDA96
t=18	6E046A2A	E9C46043	C2BC77BE	B04AC975	0119B6FA	27D1341C	B7C5D077	340DC1E3
t=19	A104AAF2	6E046A2A	E9C46043	C2BC77BE	8B504F2C	0119B6FA	27D1341C	B7C5D077
t=20	B1135F22	A104AAF2	6E046A2A	E9C46043	9D4B8461	8B504F2C	0119B6FA	27D1341C
t=21	593002AF	B1135F22	A104AAF2	6E046A2A	89C2233A	9D4B8461	8B504F2C	0119B6FA
t=22	F78C8CDF	593002AF	B1135F22	A104AAF2	B1447818	89C2233A	9D4B8461	8B504F2C
t=23	26264e39	F78C8CDF	593002AF	B1135F22	2CEB09EF	B1447818	89C2233A	9D4B8461
t=24	A47262F8	26264e39	F78C8CDF	593002AF	3CE96D37	2CEB09EF	B1447818	89C2233A
t=25	5F6A65D0	A47262F8	26264e39	F78C8CDF	1B21BFCA	3CE96D37	2CEB09EF	B1447818

	a	b	c	d	e	f	g	h
t=26	322D0A66	5F6A65D0	A47262F8	26264e39	72880CF3	1B21BFCA	3CE96D37	2CEB09EF
t=27	5D4532A4	322D0A66	5F6A65D0	A47262F8	E16DD2B4	72880CF3	1B21BFCA	3CE96D37
t=28	DDB7C662	5D4532A4	322D0A66	5F6A65D0	8C03CE81	E16DD2B4	72880CF3	1B21BFCA
t=29	94101127	DDB7C662	5D4532A4	322D0A66	3AF2DD41	8C03CE81	E16DD2B4	72880CF3
t=30	BB9ABE61	94101127	DDB7C662	5D4532A4	E4367C08	3AF2DD41	8C03CE81	E16DD2B4
t=31	2DCA6C64	BB9ABE61	94101127	DDB7C662	F66A1302	E4367C08	3AF2DD41	8C03CE81
t=32	37C3069D	2DCA6C64	BB9ABE61	94101127	15FE2A9D	F66A1302	E4367C08	3AF2DD41
t=33	B3ADF16D	37C3069D	2DCA6C64	BB9ABE61	92F0C8CF	15FE2A9D	F66A1302	E4367C08
t=34	54FFC0F0	B3ADF16D	37C3069D	2DCA6C64	888BC2EA	92F0C8CF	15FE2A9D	F66A1302
t=35	1A47BA35	54FFC0F0	B3ADF16D	37C3069D	2263CF0B	888BC2EA	92F0C8CF	15FE2A9D
t=36	A0A5BF61	1A47BA35	54FFC0F0	B3ADF16D	3BA6ECB0	2263CF0B	888BC2EA	92F0C8CF
t=37	FE6F74C6	A0A5BF61	1A47BA35	54FFC0F0	9B5BBF4B	3BA6ECB0	2263CF0B	888BC2EA
t=38	D95A1214	FE6F74C6	A0A5BF61	1A47BA35	CF72DEEC	9B5BBF4B	3BA6ECB0	2263CF0B
t=39	9297AF26	D95A1214	FE6F74C6	A0A5BF61	E5B815E7	CF72DEEC	9B5BBF4B	3BA6ECB0
t=40	87DE6B67	9297AF26	D95A1214	FE6F74C6	CAF70F84	E5B815E7	CF72DEEC	9B5BBF4B
t=41	6A324E93	87DE6B67	9297AF26	D95A1214	115D5DFE	CAF70F84	E5B815E7	CF72DEEC
t=42	2B801AA3	6A324E93	87DE6B67	9297AF26	1B162DF2	115D5DFE	CAF70F84	E5B815E7
t=43	4CDF2435	2B801AA3	6A324E93	87DE6B67	9450B2B2	1B162DF2	115D5DFE	CAF70F84
t=44	9B14BB85	4CDF2435	2B801AA3	6A324E93	AA539D72	9450B2B2	1B162DF2	115D5DFE
t=45	C367E2CC	9B14BB85	4CDF2435	2B801AA3	39FE1692	AA539D72	9450B2B2	1B162DF2
t=46	D5DD5418	C367E2CC	9B14BB85	4CDF2435	7CD0FBB5	39FE1692	AA539D72	9450B2B2
t=47	DDBE75B9	D5DD5418	C367E2CC	9B14BB85	76660EA7	7CD0FBB5	39FE1692	AA539D72
t=48	D547E3D0	DDBE75B9	D5DD5418	C367E2CC	76E592D2	76660EA7	7CD0FBB5	39FE1692
t=49	D00D873F	D547E3D0	DDBE75B9	D5DD5418	893DE2D5	76E592D2	76660EA7	7CD0FBB5
t=50	E2BE816A	D00D873F	D547E3D0	DDBE75B9	E7AACEE6	893DE2D5	76E592D2	76660EA7
t=51	685A7D7C	E2BE816A	D00D873F	D547E3D0	2C0C5197	E7AACEE6	893DE2D5	76E592D2
t=52	5F4EB883	685A7D7C	E2BE816A	D00D873F	BC77EAA8	2C0C5197	E7AACEE6	893DE2D5
t=53	3BF48447	5F4EB883	685A7D7C	E2BE816A	7877F8F4	BC77EAA8	2C0C5197	E7AACEE6
t=54	625615AF	3BF48447	5F4EB883	685A7D7C	8ADFF878	7877F8F4	BC77EAA8	2C0C5197
t=55	74258477	625615AF	3BF48447	5F4EB883	3371431A	8ADFF878	7877F8F4	BC77EAA8
t=56	F01A6D47	74258477	625615AF	3BF48447	74518382	3371431A	8ADFF878	7877F8F4
t=57	1948F1E5	F01A6D47	74258477	625615AF	E59B6C83	74518382	3371431A	8ADFF878
t=58	C72C7370	1948F1E5	F01A6D47	74258477	CEBC415D	E59B6C83	74518382	3371431A
t=59	38419BB8	C72C7370	1948F1E5	F01A6D47	BFDDC827	CEBC415D	E59B6C83	74518382
t=60	681B035E	38419BB8	C72C7370	1948F1E5	692D38B2	BFDDC827	CEBC415D	E59B6C83
t=61	677506B2	681B035E	38419BB8	C72C7370	2BBBEB70	692D38B2	BFDDC827	CEBC415D
t=62	30FA541F	677506B2	681B035E	38419BB8	1781103C	2BBBEB70	692D38B2	BFDDC827
t=63	3220DCA1	30FA541F	677506B2	681B035E	84500425	1781103C	2BBBEB70	692D38B2

c) Value

Nilai akhir *hash* SHA-256 merupakan gabungan dari hasil penjumlahan dari iterasi terakhir dengan nilai *hash* awal

$H0 = 6A09E667 + 3220DCA1 = 9C2AC308$
$H1 = BB67AE85 + 30FA541F = EC6202A4$
$H2 = 3C6EF372 + 677506B2 = A3E3FA24$
$H3 = A54FF53A + 681B035E = 0D6AF898$
$H4 = 510E527F + 84500425 = D55E56A4$
$H5 = 9B05688C + 1781103C = B28678C8$
$H6 = 1F83D9AB + 2BBBEB70 = 4B3FC51B$
$H7 = 5BE0CD19 + 692D38B2 = C50E05CB$

Nilai akhir =
9C2AC308EC6202A4A3E3FA240D6AF898D55E56A4B28678C84B3FC51BC50E05CB

2. Membuat Kunci Privat dan Kunci Publik RSA

Terdapat beberapa langkah dalam membuat kunci privat dan kunci publik RSA. Langkah langkah tersebut adalah sebagai berikut

a) Memilih dua bilangan prima p dan q

Misal

p =
167468147802936909617126830143599990347062254582521780365
865184756973183442867121868917591517311008840819134643435
317696131256765914720860019261746738674757290227582619345
227749773138263920424746278906137323352678790298446740312
095971077711167340479171855926737232867555290231122216255
439689017227872348882859

q =
161915741654526302662903796136830116728704569367801056376
033275893293128135715283636333804981166813641014835338181
772270544322736774034148425472257393664135738907275571045
958174539326276050064173250152197044823310757759634718327
252203084095993287184981883760526390066646732082261247036
502133504834457569422681

b) Mendapatkan nilai modulus

$n = p \times q$

$n =$
271157293550223592921681080952590123894281520997530515621
490898440172655774633677734915590819589198964872923901396

5777427373504136503112515055848880622477732847803946616181214898487753544232424518687918869172690124320948984758771218372783403218418454354950136701613737066825841283370119056953599733343912970411257551372209901814107889850776311964265862112985254698764828700132001058133051688761724619701018869961768177180664901091072800051822974734838825710633472312895052812675180030495988318255297769457781345355836871501300463566392122502692773535380317058417349164604699081839284614646265714798994069920792672497 9

c) Mendapatkan nilai totien ϕ(n)
$\phi(n) = (p - 1) \times (q - 1)$
$\phi(n) =$
2711572935502235929216810809525901238942815209975305156214908984401726557746336777349155908195891989648729239013965777427373504136503112515055848880622477732847803946616181214898487753544232424518687918869172690124320948984758771218372783403218418454354950136701613737066825841283370119056953599733343912970378319162426463580586104827222733301256689179717952971024574982635105369900274811138236584969853236621778371179018955904423514849782947473890365425297399583009981566993636061438064741864258248877504875511919019272546494655420528187685276592819317550643043380438242405661637053276299936520616741863687800841944 0

d) Memilih bilangan e yang relatif prima terhadap ϕ(n) dan 1 < e < ϕ(n).
Misal e = 65537

e) Mencari nilai d
Memilih nilai d sedemikian sehingga memenuhi persamaan berikut
$e \times d = 1 \, mod \, (\phi(n))$
$65537 \times d = 1 \, mod$
(2711572935502235929216810809525901238942815209975305156214908984401726557746336777349155908195891989648729239013965777427373504136503112515055848880622477732847803946616181214898487753544232424518687918869172690124320948984758771218372783403218418454354950136701613737066825841283370119056953599733343912970378319162426463580586104827222733301256689179717952971024574982635105369900274811138236584969853236621778371179018955904423514849782947473890365425297399583009981566993636061438064741864258248877504875511919019272546494655420528187685276592819317550643043380438242405661637053276299936520616741863687800841944 0)

Berdasarkan persamaan tersebut dipilih nilai
d =
22281924541638679483656885322216135209427708060791626696796879388127406204261901034127506947217841404175147540756841628022914184623531535069627490671993364942671412139870275286806408514178417228440334342856771908075618411984863339059500416161393519300674654723580375390389678938076905319391058357880205546348895127969439365498708224264957669592120136576976462044579010500149681033707584979330545042795134993214405969372245731613718047625037895121670467005503174869679653830488926324470991746396350723263064767355674941468745734954760993792758729516104113305816631553186918307902681213229451574130841206696529109745953

f) Didapatkan kunci publik (e,n) dan kunci privat (d,n)
Kunci publik (65537,
27115729355022359292168108095259012389428152099753051562149089844017265577463367773491559081958919896487292390139657774273735041365031125150558488806224777328478039466161812148984877535442324245186879188691726901243209489847587712183727834032184184543549501367016137370668258412833701190569535997333439129704112575513722099018141078898507763119642658621129852546987648287001320010581330516887617246197010188699617681771806649010910728000518229747348388257106334723128950528126751800304959883182552977694577813453558368715013004635663921225026927735353803170584173491646046990818392846146462657147989940699207926724979)
kunci privat
(22281924541638679483656885322216135209427708060791626696796879388127406204261901034127506947217841404175147540756841628022914184623531535069627490671993364942671412139870275286806408514178417228440334342856771908075618411984863339059500416161393519300674654723580375390389678938076905319391058357880205546348895127969439365498708224264957669592120136576976462044579010500149681033707584979330545042795134993214405969372245731613718047625037895121670467005503174869679653830488926324470991746396350723263064767355674941468745734954760993792758729516104113305816631553186918307902681213229451574130841206696529109745953,
27115729355022359292168108095259012389428152099753051562149089844017265577463367773491559081958919896487292390139657774273735041365031125150558488806224777328478039466161825

121489848775354423242451868791886917269012432094898475877
121837278340321841845435495013670161373706682584128337011
905695359973334391297041125755137220990181410788985077631
196426586211298525469876482870013200105813305168876172461
970101886996176817718066490109107280005182297473483882571
063347231289505281267518003049598831825529776945778134535
583687150130046356639212250269277353538031705841734916460
469908183928461464626571479899406992079267249 79)

3. Enkripsi *Message digest* Menggunakan Kunci Privat

   Message digest yang telah didapatkan menggunakan SHA-256 akan dienkripsi menggunakan kunci privat yang telah dibuat sebelumnya. Setelah *message digest* berhasil terenkripsi maka terciptalah sebuah tanda tangan digital yang dibuat menggunakan SHA-256 dan RSA.

   Berikut merupakan contoh sederhana dalam mengenkripsi sebuah pesan menggunakan kunci privat RSA.

   a) Mengubah menjadi bentuk bilangan menjadi bilangan desimal

      m =
      9C2AC308EC6202A4A3E3FA240D6AF898D55E56A4B28678C84B3FC51B
      C50E05CB

      m =
      706363580368214661679979585894270322505249503788063226092
      41680693213963093451

   b) Mengenkripsi dengan kunci privat

      $c = m^d \ mod \ n$

      c =
      706363580368214661679979585894270322505249503788063226092
      41680693213963093451$^{222819245416386794836568853222161352094277080607916266967968793881274062042619010341275069472178414041751475407568416280229141846235315350696274906719336494267141213987027528680640851417841722844033434285677190807561841198486333905950041616139351930067465472358037539038967893807690531939105835788020554634889512796943936549870822426495766959212013657697646204457901050014968103370758497933054504279513499321440596937224573161371804762503789512167046700550317486967965383048892632447099174639635072326306476735567494146874573495476099379275872951610411330581663155318691830790268121322945157413084120669652910974595 3 \ mod$
      271157293550223592921681080952590123894281520997530515621490898440172655774633677734915590819589198964872923901396577742737350413650311251505584888062247773284780394661618121489848775354423242451868791886917269012432094898475877121837278340321841845435495013670161373706682584128337011905695359973334391297041125755137220990181410788985077631196426586211298525469876482870013200105813305168876172461970101886996176817718066490109107280005182297473483882571

06334723128950528126751800304959883182552977694577813453558368715013004635663921225026927735353803170584173491646046990818392846146462657147989940699207926724979

$c =$

52358151434052031300167557379701189534504445047823637655857619613970669712664462608601176236470233286312843939330621100498391575716662352188165599587022269075838120791581365580565400614641781152889391805848074744054447356031124722879915449615102757308043249611588725456858344523916621846476083506305017412907276297627034034229068745772002978350131700280072731924560516920684257908371144608054415450622490167656704360596033424362312556984770313773395195647035644249218064422578891549193953391124244646957793857431171726213004348915720125796896314303198562496487653776185557989938354027209974772524454488337221064889 88

c) Mengubah menjadi bentuk bilangan heksadesimal

$c =$

2979C3E1B9D9646A4734B5B800BDF7C45CA5368E5232DF57CBFF90D3B3F66DA9D18138BFA90358DFE66D9D55B68A136F883586C39C8E76A609B5183A941DC46EAAE68DCD3A931BFC3FA8439A8F995976B330A35CA6D9FF95A39A3CF2766A803B43C08B6A4525146D63C2A0FA87904BF38288695B857B2EEE89FD12C37391AE75D83D442C4BDD6751D1BA0687C36EC9B4503548AFF4016826051A6C50CF0DE2EB2B7BFE1D8332B5A9C5F07CDE5AD98751123C72B9D31A581A712ED3A09E0BC32C72DBA7655FD1843FC3B63DA42574DBA3B38BFADBFC5825A59C0C83ECB2C643CA671210E8ABD206C7E51A1C0F3A007A80B6EB0C96897BFD0BB40885260CB3A89C

3.1.4. Verifikasi Tanda Tangan Digital

1. Dekripsi Tanda Tangan Menggunakan Kunci Publik

   Tanda tangan yang dikirimkan akan didekripsi menggunakan kunci publik untuk mendapatkan *message digest* SHA-256 dari *file* PDF yang dikirim. Berikut merupakan langkah mendekripsi pesan menggunakan kunci privat yang telah dibuat sebelumnya

   a) Mengubah bentuk bilangan menjadi bilangan heksadesimal

   $c =$

   2979C3E1B9D9646A4734B5B800BDF7C45CA5368E5232DF57CBFF90D3B3F66DA9D18138BFA90358DFE66D9D55B68A136F883586C39C8E76A609B5183A941DC46EAAE68DCD3A931BFC3FA8439A8F995976B330A35CA6D9FF95A39A3CF2766A803B43C08B6A4525146D63C2A0FA87904BF38288695B857B2EEE89FD12C37391AE75D83D442C4BDD6751D1BA0687C36EC9B4503548AFF4016826051A6C50CF0DE2EB2B7BFE1D8332B5A9C5F07CDE5AD98751123C72B9D31A581A712ED3A09E0BC32C72DBA7655FD1843FC3B63DA42574DBA3B38BFADBFC5825A59C0C83ECB2C643

CA671210E8ABD206C7E51A1C0F3A007A80B6EB0C96897BFD0BB40885
260CB3A89C

$c =$

5235815143405203130016755737970118953450444504782363765585761961397066971266446260860117623647023328631284393933062110049839157571666235218816559958702226907583812079158136558056540061464178115288939180584807474405444735603112472287991544961510275730804324961158872545685834452391662184647608350630501741290727629762703403422906874577200297835013170028007273192456051692068425790837114460805441545062249016765670436059603342436231255698477031377339519564703564424921806442257889154919395339112424464695779385743117172621300434891572012579689631430319856249648765377618555798993835402720997477252445448833722106488988

b) Mendekripsi dengan kunci publik

$m = c^e \ mod \ n$

$m =$

5235815143405203130016755737970118953450444504782363765585761961397066971266446260860117623647023328631284393933062110049839157571666235218816559958702226907583812079158136558056540061464178115288939180584807474405444735603112472287991544961510275730804324961158872545685834452391662184647608350630501741290727629762703403422906874577200297835013170028007273192456051692068425790837114460805441545062249016765670436059603342436231255698477031377339519564703564424921806442257889154919395339112424464695779385743117172621300434891572012579689631430319856249648765377618555798993835402720997477252445448833722106488988$^{65537} \ mod$

2711572935502235929216810809525901238942815209975305156214908984401726557746336777349155908195891989648729239013965777427373504136503112515055848880622477732847803946616181214898487753544232424518687918869172690124320948984758771218372783403218418454354950136701613737066825841283370119056953599733343912970411257551372209901814107889850776311964265862112985254698764828700132001058133051688761724619701018869961768177180664901091072800051822974734838825710633472312895052812675180030495988318255297769457781345355836871501300463566392122502692773535380317058417349164604699081839284614646265714798994069920792672497958363506821466167997958589427032250524950378806322609241680693213963093451

c) Mengubah menjadi bentuk bilangan heksadesimal
$m =$
706363580368214661679979585894270322505249503788063226092416806932139630934 51
$m =$
9C2AC308EC6202A4A3E3FA240D6AF898D55E56A4B28678C84B3FC51BC50E05CB

2. Mendapatkan *Message digest* SHA-256
Setelah melakukan dekripsi terhadap tanda tangan, kemudian dilakukan perhitungan *message digest* dengan metode SHA-256 pada *file* PDF yang dikirimkan. Tahap tahap perhitungan SHA-256 ini sama persis dengan perhitungan SHA-256 pada tahap pembuatan tanda tangan digital.

3. Membandingkan *Message digest* Hasil Dekripsi dengan Hasil Hitung SHA-256
Setelah didapatkan *message digest* hasil perhitungan SHA-256, kemudian dilakukan perbandingan antara *message digest* hasil perhitungan SHA-256 dengan *message digest* hasil dekripsi tanda tangan.

4. Hasil Perbandingan *Message digest*
Sebuah tanda tangan dinyatakan valid jika *message digest* hasil perhitungan SHA-256 sama dengan *message digest* hasil dekripsi tanda tangan. Akan tetapi, jika kedua *message digest* tersebut tidak sama maka tanda tangan tersebut dinyatakan tidak valid.

3.1.5. Pengujian

Tahapan Pengujian pada penelitian ini dilakukan dengan menguji tanda tangan digital yang telah dibuat sebelumnya. Pengujian dilakukan dengan berbagai skenario yang dapat dilihat pada tabel 3.9 berikut.

Tabel 3.9 Pengujian Modifikasi Dokumen PDF

No	Skenario Pengujian	Hasil validasi
1	Dokumen PDF diubah isi kontennya	
2	Dokumen PDF diberikan komentar	
3	Dokumen PDF diberikan *highlight*	
4	Dokumen PDF dikompres ukurannya	
5	Dokumen PDF dikunci dengan *password*	
6	Dokumen PDF diubah nama *file*-nya	
7	Dokumen PDF diubah metadatanya	

Tabel 3.10 Pengujian tanda tangan digital dokumen PDF dari *File* DOCX yang sama dari beberapa *renderer* PDF

Tanda tangan digital / *Renderer* PDF	Microsoft Print to PDF	Nitro PDF	Smallpdf.com	Ilovepdf.com
Microsoft Print to PDF				
Nitro PDF				
Smallpdf.com				
Ilovepdf.com				

Tabel 3.11 Pengujian tanda tangan digital dokumen PDF dari *file* DOCX yang sama dengan 3 kali iterasi dari masing masing *renderer* PDF

Iterasi	Microsoft Print to PDF	Nitro PDF	Smallpdf.com	Ilovepdf.com	Foxit.com
1					
2					
3					

3.1.6. Hasil Penelitian

Hasil dari penelitian ini adalah analisa kemampuan SHA-256 dan RSA berdasarkan hasil pengujian skenario skenario yang telah dilakukan sebelumnya.

3.2. Metodologi Pengembangan Sistem

Penelitian ini menggunakan metode pengembangan sistem waterfall. Waterfall merupakan salah satu model *System Development Life Cycle* (SDLC). Tahapan model waterfall antara lain analisa kebutuhan, desain sistem, pengembangan sistem, dan pengujian sistem.

3.2.1. Analisis Kebutuhan

Pada tahap ini akan dilaksanakan analisis terhadap hal-hal yang berkaitan dengan kebutuhan untuk mengembangkan sistem pada penelitian ini. Analisis kebutuhan sistem ini terdapat dua bagian yaitu kebutuhan fungsional dan kebutuhan non-fungsional.

1) Kebutuhan Fungsional

Kebutuhan fungsional merupakan kebutuhan yang berkaitan dengan sistem yang akan dibuat. Sehingga mencakup kebutuhan proses proses yang terdapat dalam sistem. Kebutuhan fungsional dalam sistem ini adalah sebagai berikut :

a. Sistem dapat menerima *input file* pengguna,
b. Sistem dapat menghasilkan *file* tanda tangan pada pengguna,
c. Sistem dapat memverifikasi *file* tanda tangan dengan *file* PDF pengguna,
d. Sistem dapat menampilkan hasil verifikasi *file* tanda tangan dengan *file* PDF pengguna

2) Kebutuhan Non-Fungsional

Kebutuhan non-fungsional merupakan kebutuhan yang tidak berkaitan langsung dengan sistem yang akan dibuat. Kebutuhan non-fungsional mencakup kebutuhan perangkat keras (hadrware) dan kebutuhan perangkat lunak (software). Kebutuhan perangkat lunak dan perangkat keras untuk penelitian ini akan dijelaskan pada tabel berikut

Tabel 3.12 Kebutuhan Perangkat Keras

Perangkat Lunak	Spesifikasi
Sistem Operasi	Windows 11
Processor	i5-8265U CPU @ 1.60GHz 1.80 GHz
Memory / RAM	12 GB

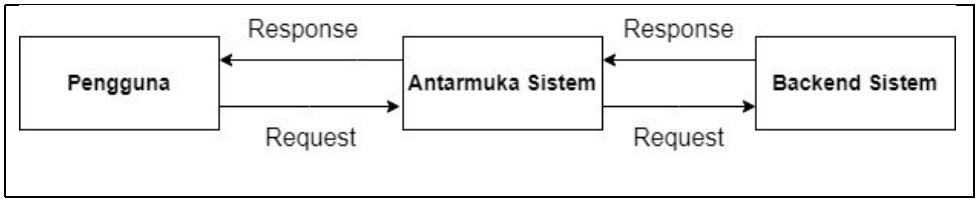
Tabel 3.13 Kebutuhan Perangkat Lunak

Perangkat Keras	Spesifikasi
Kode Editor	Visual Studio Code
Bahasa Pemrograman	Javascript
Web Browser	Chrome

3.2.2. Desain Sistem

Pada tahap ini akan dibuat desain sistem yang akan digunakan dalam penelitian. Pada penelitian ini, desain sistem mencakup desain arsitektur, flowchart, dan antarmuka.

1) Desain Arsitektur Sistem

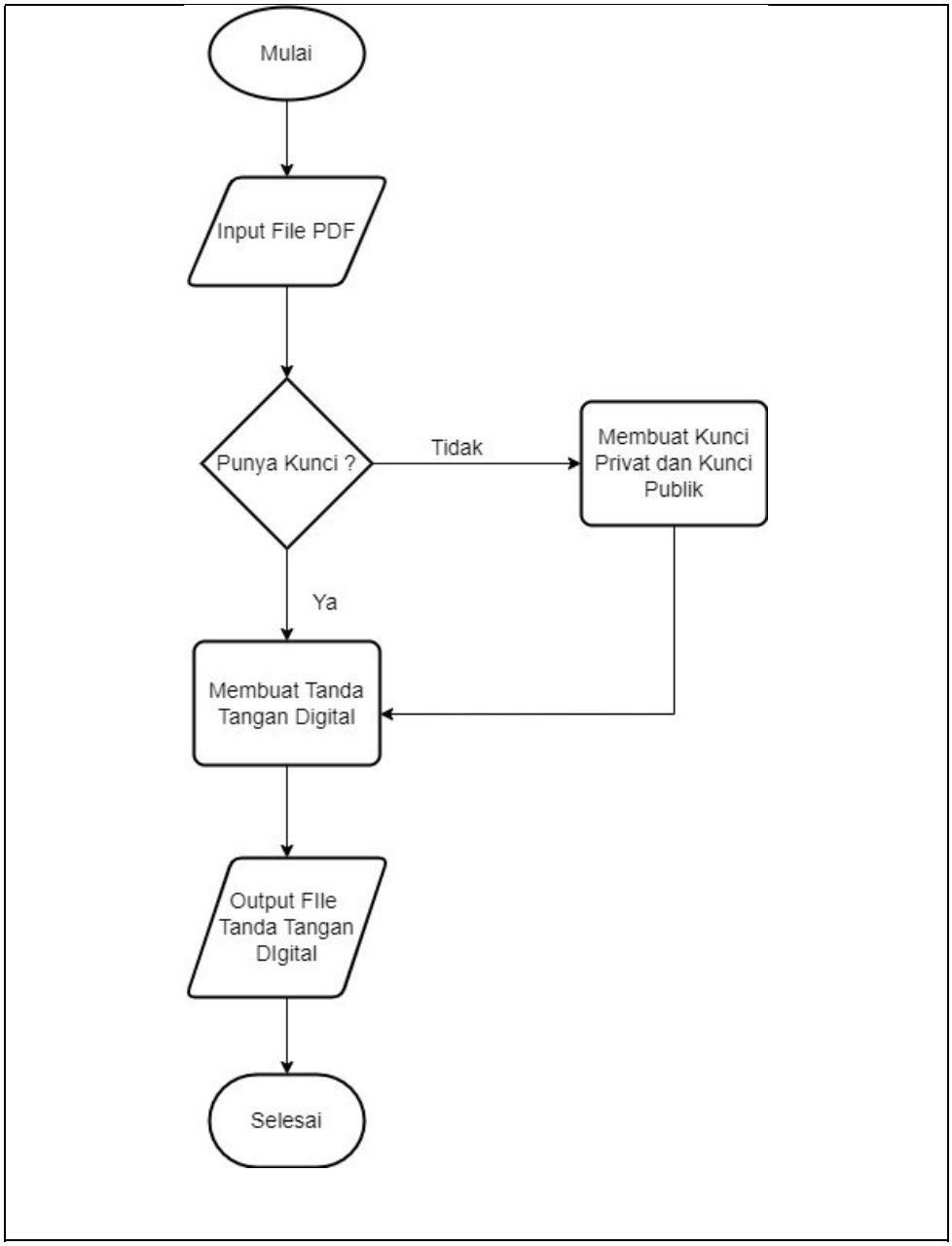


Gambar 3.4 Desain Arsitektur Sistem

2) Desain Flowchart Sistem

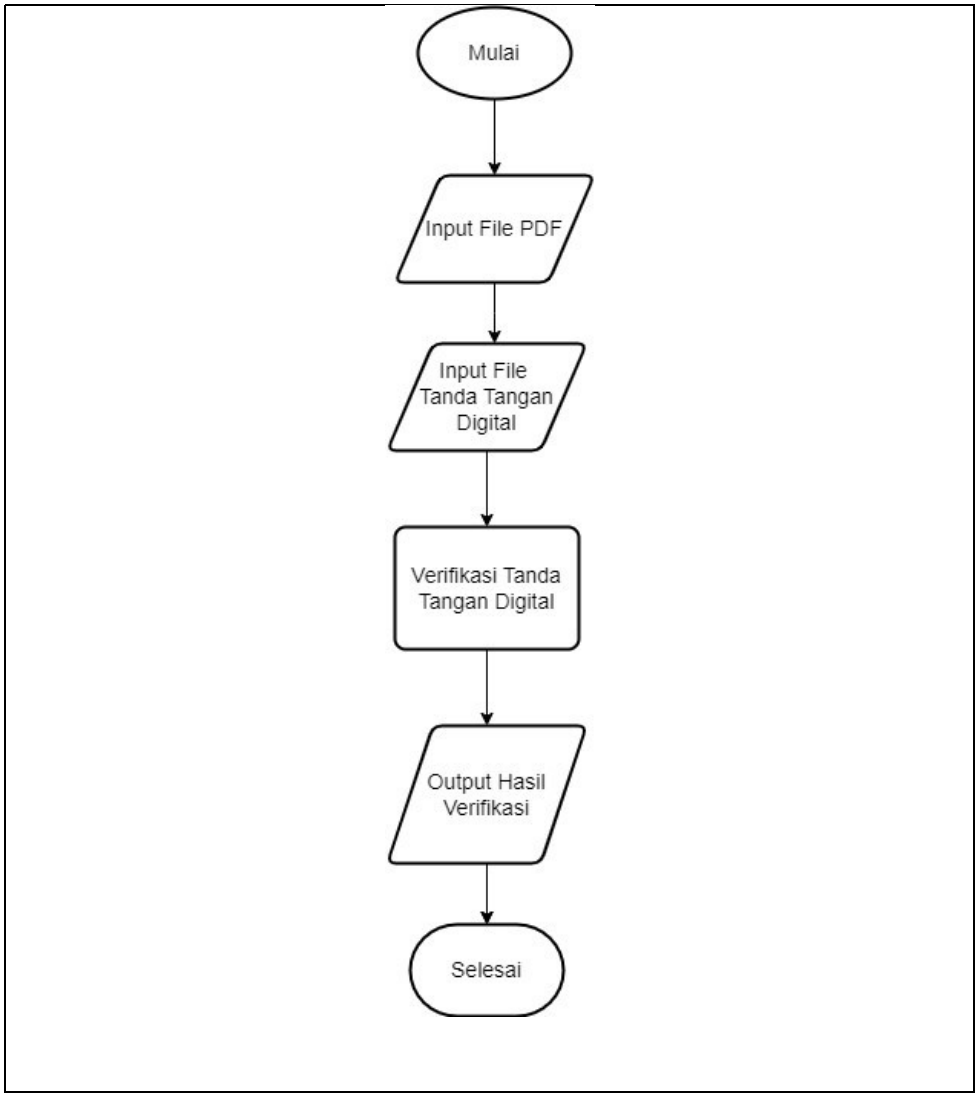
Flowchart atau diagram alir merupakan diagram yang menggambarkan langkah langkah atau alur proses. Berikut adalah flowchart yang digunakan

untuk menggambarkan proses pembuatan dan verifikasi tanda tangan digital pada penelitian ini.



Gambar 3.5 Flowchart Pembuatan Tanda Tangan Digital

Proses pembuatan tanda tangan digital dijelaskan telah dijelaskan oleh gambar 3.5. Proses diawali dengan pengguna melakukan *input file* PDF dan diakhiri dengan pengguna mendapatkan *file* tanda tangan digital.



Gambar 3.6 Flowchart Verifikasi Tanda Tangan Digital

Proses verifikasi tanda tangan digital dijelaskan telah dijelaskan oleh gambar 3.6. Proses diawali dengan pengguna melakukan *input file* PDF dan *file* tanda tangan digital kemudian diakhiri dengan pengguna menerima hasil verifikasi tanda tangan digital.

3) Desain Antarmuka Sistem

Desain antarmuka merupakan tahapan yang sangat penting dalam menentukan bagaimana sebuah sistem akan dilihat dan digunakan oleh pengguna. Tahapan ini memberikan panduan yang jelas untuk pembuatan antarmuka pengguna pada saat proses pengembangan dimulai. Desain antarmuka sistem pada penelitian ini adalah sebagai berikut:

a. Desain Antarmuka Halaman Utama

Halaman ini merupakan halaman yang berfungsi sebagai halaman utama. Halaman ini akan mengarahkan pengguna ke halaman halaman lainnya

Gambar 3.7 Desain antarmuka halaman utama

b. Desain Antarmuka Halaman *Sign* PDF

Halaman ini merupakan halaman yang berfungsi sebagai pembuatan tanda tangan digital dari *file* PDF. Halaman ini akan menerima *input file* PDF kemudian akan menghasilkan *file* tanda tangan yang dapat di unduh oleh pengguna.

Gambar 3.8 Desain antarmuka halaman *Sign* PDF

c. Desain Antarmuka Halaman *Verify* PDF

Halaman ini merupakan halaman yang berfungsi sebagai verifikasi *file* PDF. Halaman ini akan menerima *input file* PDF dan *file* tanda tangan digital kemudian akan menampilkan hasil verifikasi dari *file* PDF tersebut.

Gambar 3.9 Desain antarmuka halaman *Verify* PDF

3.2.3. Pengembangan Sistem

Pada tahap ini akan dilakukan pengembangan sistem yang akan digunakan dalam penelitian. Pengembangan sistem ini meliputi penulisan kode dalam bahasa pemrograman tertentu yang mengimplementasikan desain desain yang telah dibuat pada tahapan sebelumnya. Tujuan dari pengembangan sistem dengan kode pemprograman adalah untuk mewujudkan konsep aplikasi dalam bentuk yang dapat dijalankan dan digunakan oleh pengguna.

3.2.4. Pengujian Sistem

Pada tahap ini akan dilakukan pengujian terhadap sistem yang telah dikembangkan. Pengujian sistem pada penelitian ini akan berfokus pada pengujian fungsionalitas sistem. Tujuan dari pengujian sistem ini adalah untuk memverifikasi agar sistem yang telah dikembangkan telah memenuhi persyaratan dan kelayakan penggunaan sesuai dengan spesifikasi awal yang telah ditentukan. Skenario pengujian atau test case yang akan dilakukan akan dijelaskan pada tabel berikut

Tabel 3.14 Pengujian Sistem

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil yang Didapatkan
1	*Input file* bertipe .pdf pada kolom *input* .pdf	Berhasil	
2	*Input file* selain bertipe .pdf pada kolom *input* .pdf	Gagal	
3	*Input file* bertipe .sig pada kolom *input* .sig	Berhasil	
4	*Input file* selain bertipe .sig pada kolom *input* .sig	Gagal	
5	Menekan tombol *Sign* tanpa melakukan *input file*	Gagal	
6	Menekan tombol *Verify* tanpa melakukan *input file*	Gagal	

BAB IV

HASIL DAN PEMBAHASAN

4.1 Hasil

Pada subbab ini akan dipaparkan hasil dari implementasi sistem yang telah didesain berserta dengan pengujiannya seperti yang dijelaskan pada bab sebelumnya. Hasil dari penelitian ini adalah sebagai berikut

4.1.1. Pembuatan Tanda Tangan Digital

Tanda tangan digital dibuat menggunakan algoritma RSA dan SHA-256 sesuai dengan tahapan tahapan yang dijelaskan dalam bab sebelumnya. Pembuatan tanda tangan digital ini dilakukan dengan menggunakan sistem perangkat lunak yang dibuat berdasarkan rancangan bab sebelumnya.

```
// Mengambil data PDF
const pdfBuffer = fs.readFileSync(filepath);

// Mengambil privat key
privateKey = fs.readFileSync(privateKeyPath, "utf8");

// Membuat tanda tangan digital
const sign = createSign("SHA256");
sign.update(pdfBuffer);
sign.end();
const signature = sign.sign(privateKey);
```

Gambar 4.1 Proses pembuatan tanda tangan digital

4.1.2. Verifikasi Tanda Tangan Digital

Tanda tangan digital yang telah dibuat sebelumnya akan diverifikasi untuk mengecek keaslian dari tanda tangan digital tersebut. Proses verifikasi tanda tangan digital ini dilakukan dengan menggunakan sistem perangkat lunak yang dibuat berdasarkan rancangan bab sebelumnya.

```
// Mengambil data PDF, tanda tangan digital, dan public key
const pdfData = fs.readFileSync(pdfFile.path);
const signatureData = fs.readFileSync(sigFile.path);
const publicKey = fs.readFileSync("public-key.pem", "utf8");

// Memverifikasi tanda tangan digital dengan File PDF
const verifier = createVerify("SHA256");
verifier.update(pdfData);
const isValid = verifier.verify(publikKey, signatureData,
'base64');
```

Gambar 4.2 Proses verifikasi tanda tangan digital

4.1.3. Pengujian Tanda Tangan Digital

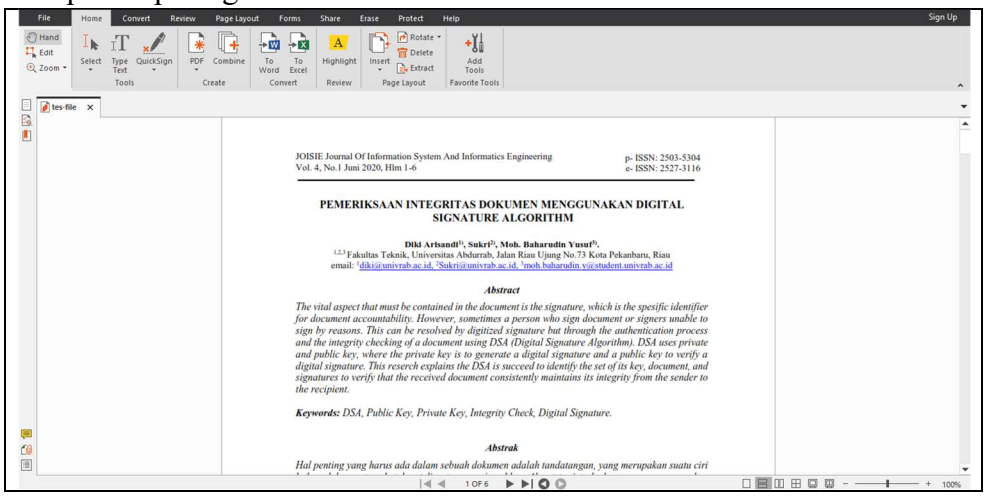
Setelah berhasil melakukan pembuatan dan verifikasi tanda tangan digital. Tahapan berikutnya yaitu melakukan pengujian tanda tangan digital. Tanda tangan digital akan diuji dengan berbagai skenario pengujian guna mengetahui sejauh mana kemampuan tanda tangan digital yang dibuat menggunakan SHA-256 dan RSA dapat mendeteksi perubahan pada dokumen PDF. Hasil pengujian yang dari masing masing skenario adalah sebagai berikut

1. Pengujian Modifikasi Dokumen PDF

   Pengujian modifikasi dokumen PDF ini digunakan menggunakan *tools* Nitro Pro 11

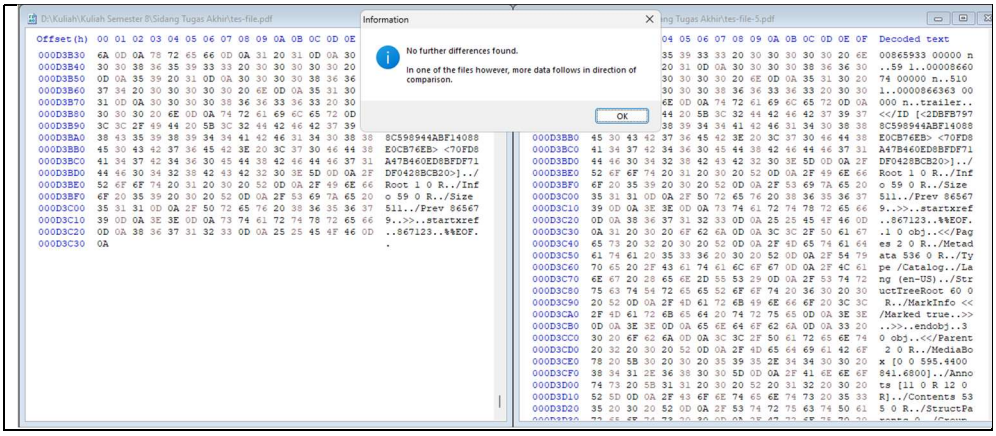
   1) Dokumen PDF diubah isi kontennya

      Skenario berikutnya yaitu menguji validitas tanda tangan digital dari dokumen PDF yang telah diubah isi kontennya. Hasil perubahan konten PDF ditampilkan pada gambar 4.3 berikut



Gambar 4.3 Dokumen PDF yang diubah isi kontennya

Gambar 4.4 Hasil verifikasi dokumen PDF yang diubah isi kontennya

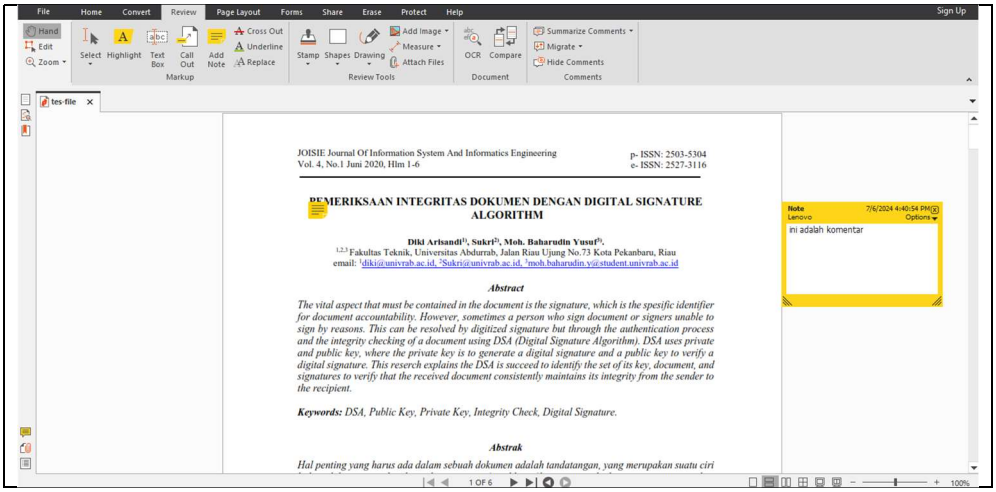


Gambar 4.5 Perbandingan bit sebelum dan sesudah pengubahan konten PDF

Hasil verifikasi dokumen PDF yang diubah isi kontennya dapat dilihat pada gambar 4.4. Hasil verifikasi dari tanda tangan digital tersebut adalah tidak valid yang berarti dideteksi adanya perubahan bit dari dokumen PDF tersebut. Hal tersebut juga dibuktikan dengan gambar 4.5 yang menunjukan perbedaan bit dari kedua *file* PDF. Meskipun berbeda, kedua *file* tersebut memiliki bit awal yang sama persis.

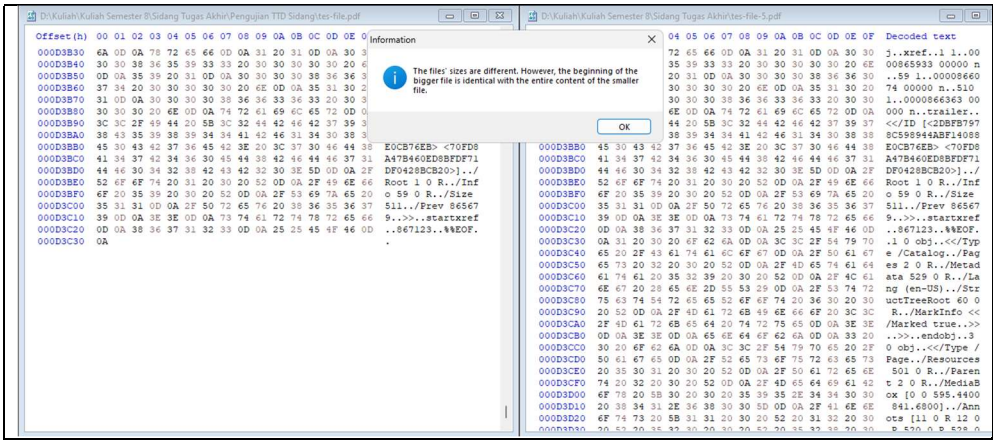
2) Dokumen PDF diberikan komentar

Skenario berikutnya yaitu menguji validitas tanda tangan digital dari dokumen PDF yang telah diberikan komentar. Dokumen PDF yang diberikan komentar ditampilkan pada gambar 4.5 berikut



Gambar 4.6 Dokumen PDF yang diberikan komentar

Gambar 4.7 Hasil verifikasi dokumen PDF yang diberikan komentar

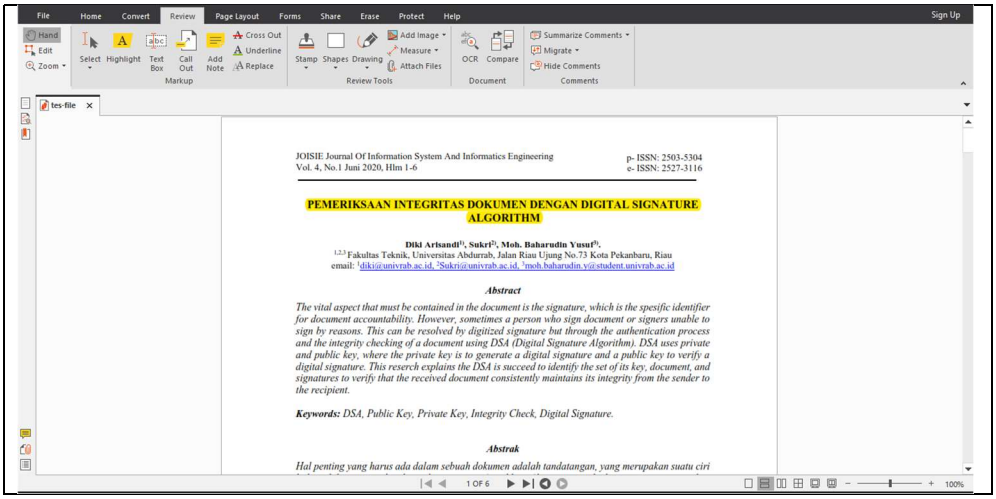


Gambar 4.8 Perbandingan bit sebelum dan sesudah penambahan komentar PDF

Hasil verifikasi dokumen PDF yang diberikan komentar dapat dilihat pada gambar 4.7. Hasil verifikasi dari tanda tangan digital tersebut adalah tidak valid yang berarti dideteksi adanya perubahan bit dari dokumen PDF tersebut. Hal tersebut juga dibuktikan dengan gambar 4.8 yang menunjukan perbedaan bit dari kedua *file* PDF. Meskipun berbeda, kedua *file* tersebut memiliki bit awal yang sama persis.

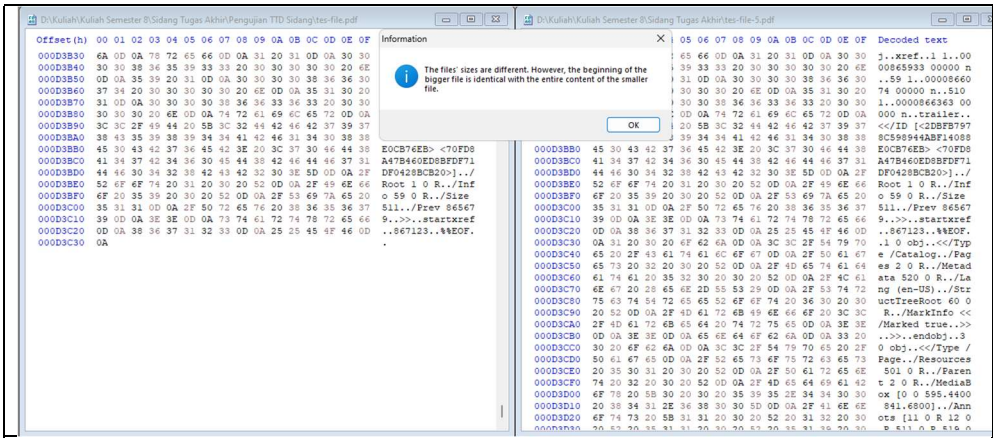
3) Dokumen PDF diberikan *highlight*

Skenario berikutnya yaitu menguji validitas tanda tangan digital dari dokumen PDF yang telah diberikan *highlight*. Dokumen PDF yang diberikan *highlight* ditampilkan pada gambar 4.9 berikut



Gambar 4.9 Dokumen PDF yang diberikan *highlight*

Gambar 4.10 Hasil verifikasi dokumen PDF yang diberikan *highlight*

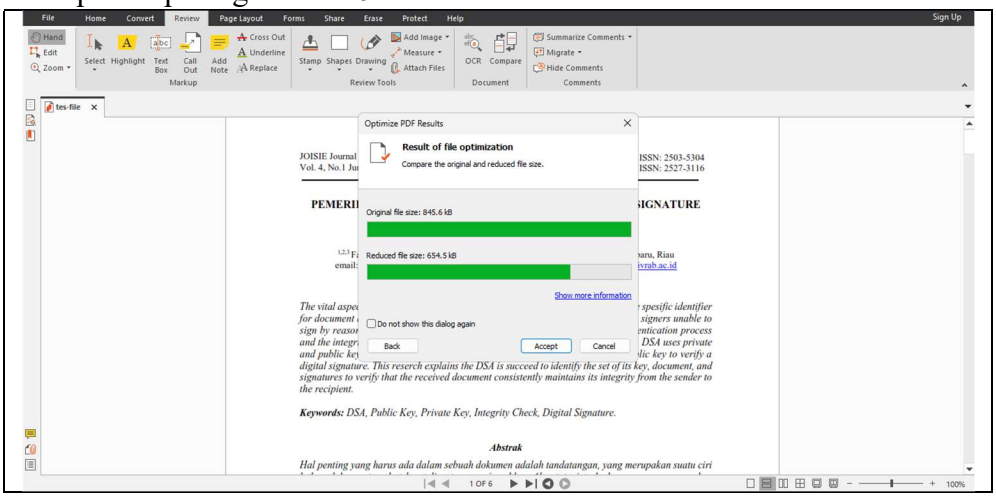


Gambar 4.11 Perbandingan bit sebelum dan sesudah penambahan *highlight* PDF

Hasil verifikasi dokumen PDF yang diberikan *highlight* dapat dilihat pada gambar 4.10. Hasil verifikasi dari tanda tangan digital tersebut adalah tidak valid yang berarti dideteksi adanya perubahan bit dari dokumen PDF tersebut. Hal tersebut juga dibuktikan dengan gambar 4.11 yang menunjukan perbedaan bit dari kedua *file* PDF. Meskipun berbeda, kedua *file* tersebut memiliki bit awal yang sama persis.

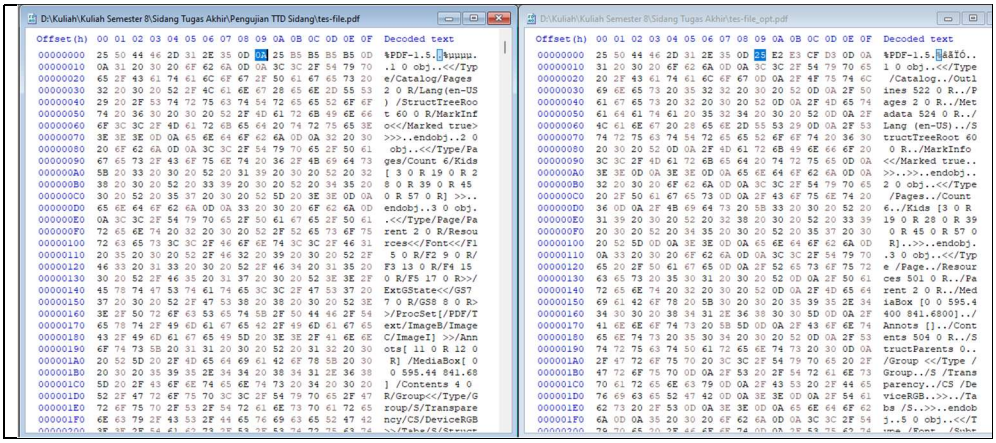
4) Dokumen PDF dikompres ukurannya

Skenario berikutnya yaitu menguji validitas tanda tangan digital dari dokumen PDF yang dikompres ukurannya. Dokumen PDF yang dikompres ditampilkan pada gambar 4.9 berikut



Gambar 4.12 Dokumen PDF yang dikompres ukurannya

Gambar 4.13 Hasil verifikasi dokumen PDF yang dikompres ukurannya

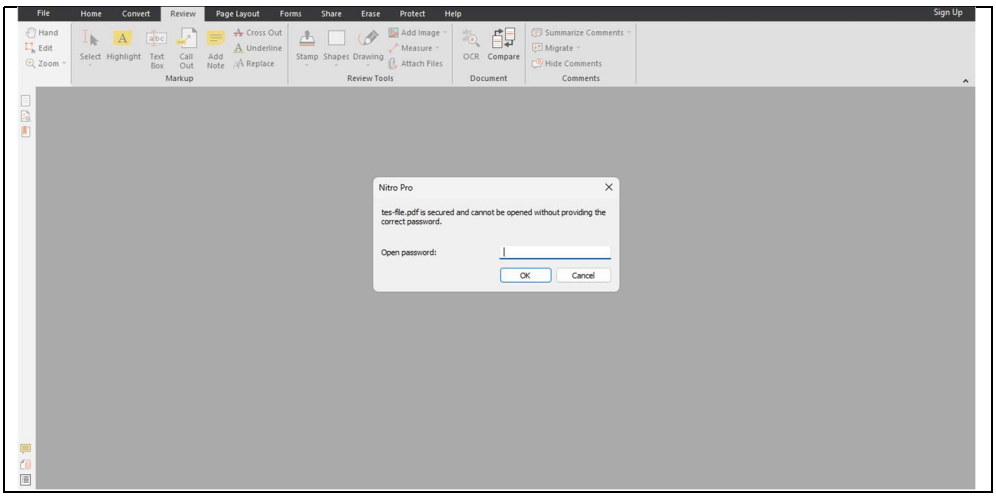


Gambar 4.14 Perbandingan bit sebelum dan sesudah kompres PDF

Hasil verifikasi dokumen PDF yang dikompres ukurannya dapat dilihat pada gambar 4.13. Hasil verifikasi dari tanda tangan digital tersebut adalah tidak valid yang berarti dideteksi adanya perubahan bit dari dokumen PDF tersebut. Hal tersebut juga dibuktikan dengan gambar 4.14 yang menunjukan perbedaan bit dari kedua *file* PDF. Kedua *file* PDF tersebut memiliki perbedaan bit yang signifikan.

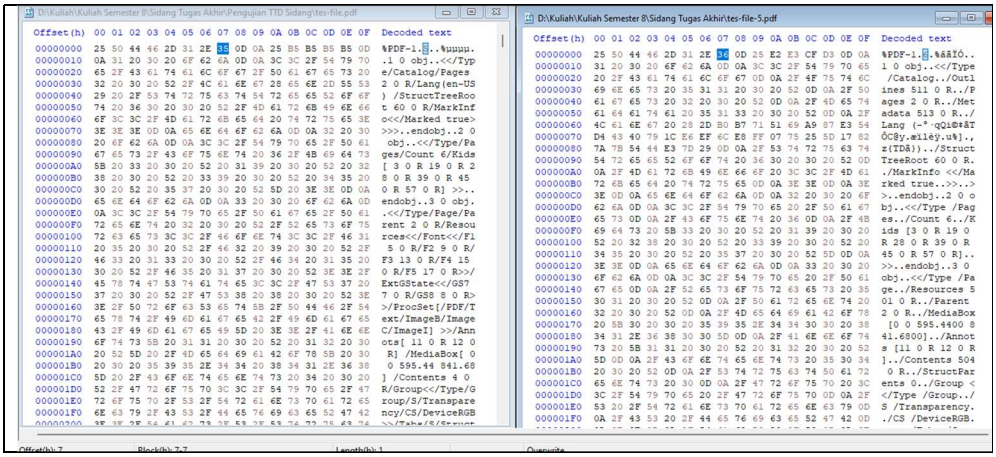
5) Dokumen PDF dikunci dengan *password*

Skenario berikutnya yaitu menguji validitas tanda tangan digital dari dokumen PDF yang dikunci dengan *password*. Dokumen PDF yang dikunci dengan *password* ditampilkan pada gambar 4.11 berikut



Gambar 4.15 Dokumen PDF yang dikunci dengan *password*

Gambar 4.16 Hasil verifikasi dokumen PDF yang dikunci dengan *password*



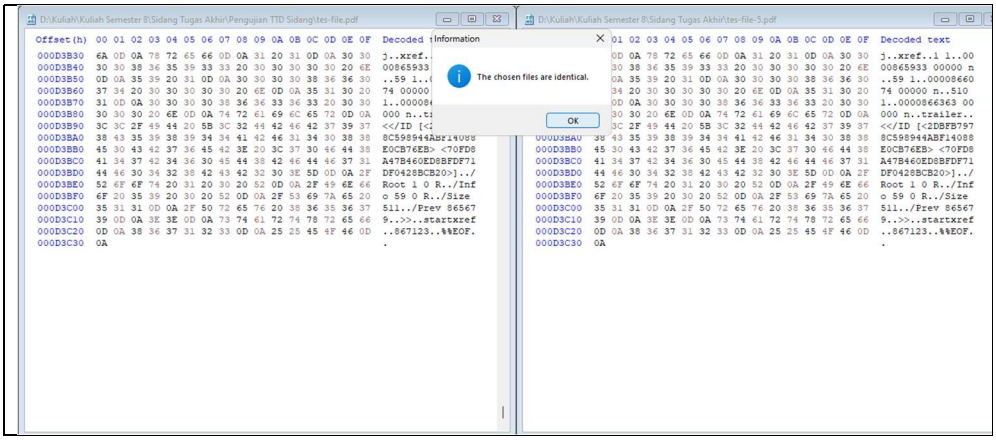
Gambar 4.17 Perbandingan bit sebelum dan sesudah dikunci dengan *password*

Hasil verifikasi dokumen PDF yang dikunci dengan *password* dapat dilihat pada gambar 4.16. Hasil verifikasi dari tanda tangan digital tersebut adalah tidak valid yang berarti dideteksi adanya perubahan bit dari dokumen PDF tersebut. Hal tersebut juga dibuktikan dengan gambar 4.17 yang menunjukan perbedaan bit dari kedua *file* PDF. Kedua *file* PDF tersebut memiliki perbedaan bit yang signifikan.

6) Dokumen PDF diubah nama *file*-nya

Skenario berikutnya yaitu menguji validitas tanda tangan digital dari dokumen PDF yang diubah nama *file*-nya. Dokumen PDF diubah nama *file* nya menjadi tes-*file*2.pdf

Gambar 4.18 Hasil verifikasi dokumen PDF yang diubah nama *file*-nya

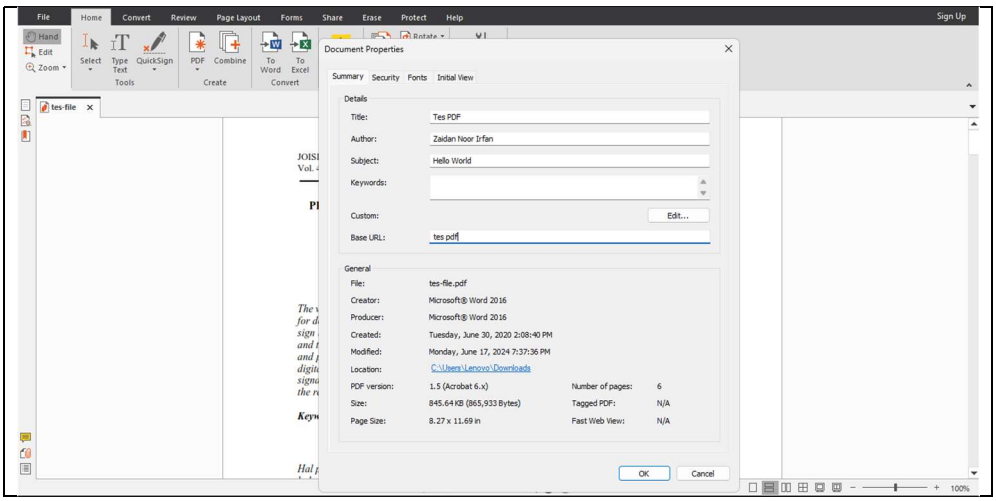


Gambar 4.19 Perbandingan bit sebelum dan sesudah diubah nama *file*-nya

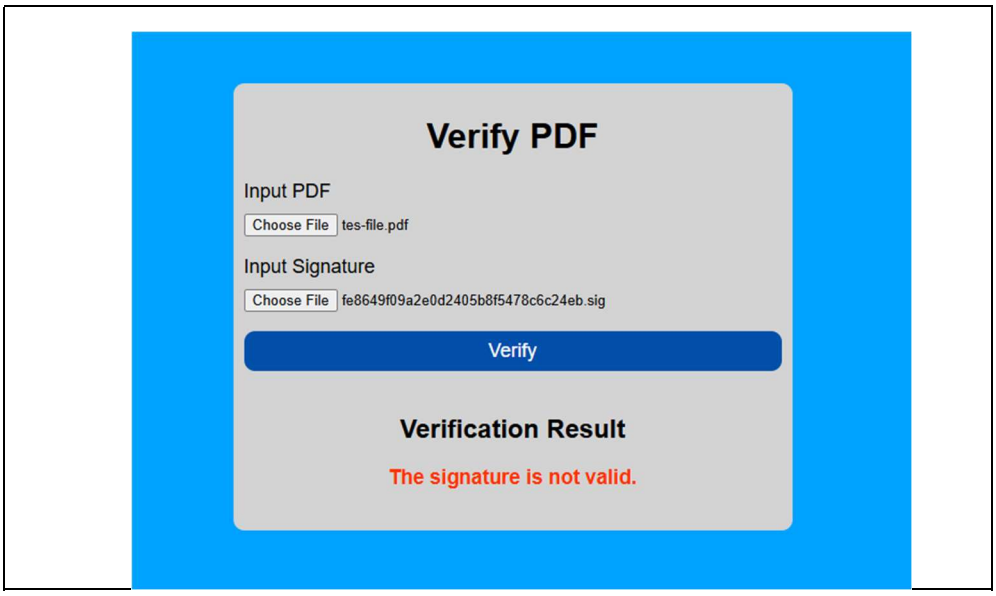
Hasil verifikasi dokumen PDF yang diubah nama *file*-nya dapat dilihat pada gambar 4.18. Hasil verifikasi dari tanda tangan digital tersebut adalah valid yang berarti tidak terdeteksi adanya perubahan bit dari dokumen PDF tersebut. Hal ini juga dibuktikan dengan gambar 4.19 yang menyatakan bahwa bit dari kedua *file* PDF tersebut identik.

7) Dokumen PDF diubah metadatanya

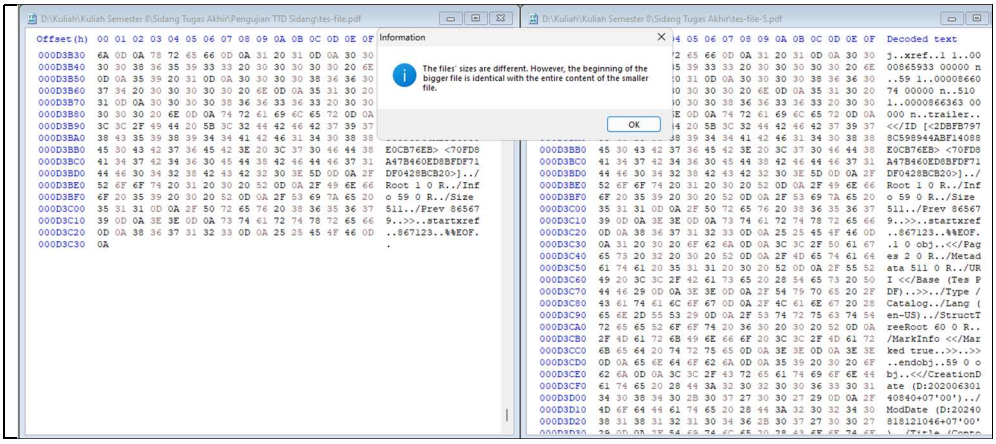
Skenario berikutnya yaitu menguji validitas tanda tangan digital dari dokumen PDF yang telah diubah meta datanya. Dokumen PDF yang diubah metadatanya ditampilkan pada gambar 4.14 berikut



Gambar 4.20 Dokumen PDF yang diubah metadatanya



Gambar 4.21 Hasil verifikasi dokumen PDF yang diubah metadanya



Gambar 4.22 Perbandingan bit sebelum dan sesudah diubah metadatanya

Hasil verifikasi dokumen PDF yang diubah metadatanya dapat dilihat pada gambar 4.21. Hasil verifikasi dari tanda tangan digital tersebut adalah tidak valid yang berarti terdeteksi adanya perubahan dari dokumen PDF tersebut. Hal tersebut juga dibuktikan dengan gambar 4.122 yang menunjukan perbedaan bit dari kedua *file* PDF. Meskipun berbeda, kedua *file* tersebut memiliki bit awal yang sama persis.

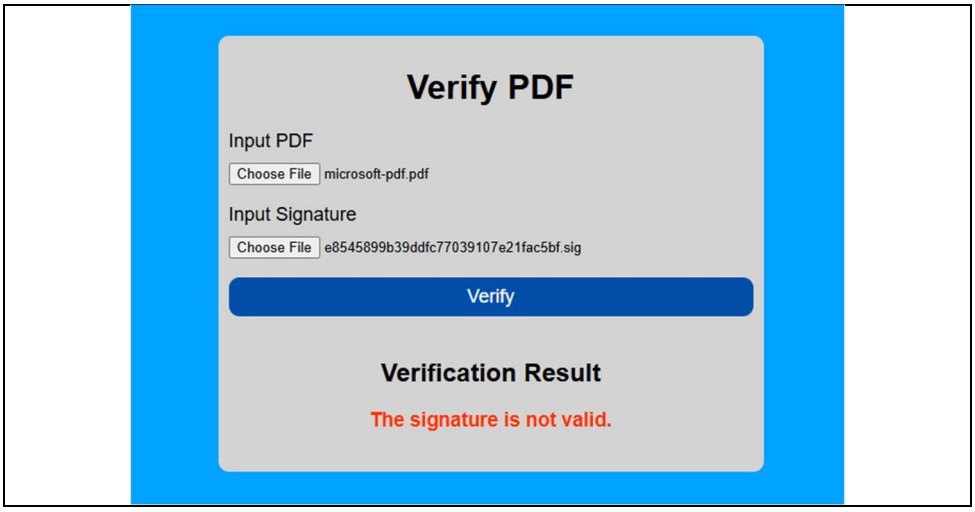
Setelah melakukan berbagai skenario pengujian modifikasi dokumen PDF diatas, dibuatlah tabel ringkasan hasil pengujian sebagai berikut

Tabel 4.1 Hasil pengujian modifikasi dokumen PDF

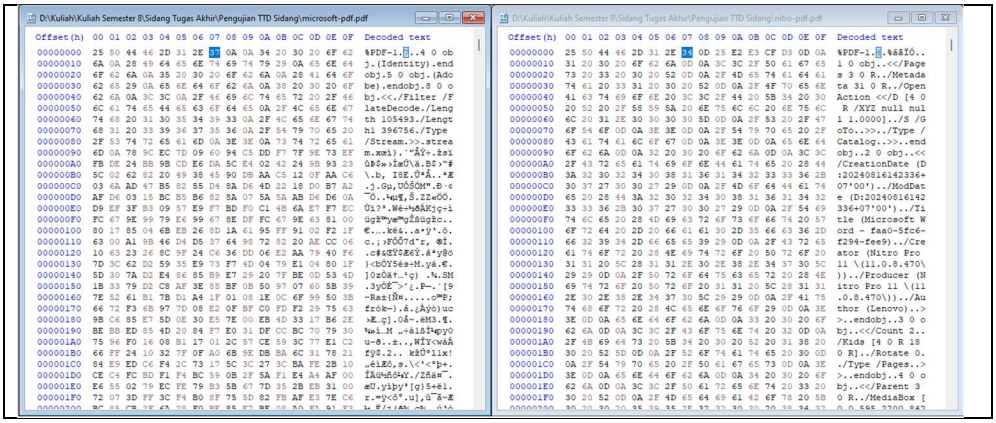
No	Skenario Pengujian	Hasil validasi
1	Dokumen PDF diubah isi kontennya	Tidak Valid
2	Dokumen PDF diberikan komentar	Tidak Valid
3	Dokumen PDF diberikan *highlight*	Tidak Valid
4	Dokumen PDF dikompres ukurannya	Tidak Valid
5	Dokumen PDF dikunci dengan *password*	Tidak Valid
6	Dokumen PDF diubah nama *file*-nya	Valid
7	Dokumen PDF diubah metadatanya	Tidak Valid

2. Pengujian Pembuatan dokumen PDF dari *File* DOCX yang Sama Menggunakan Beberapa *Renderer* PDF
   1) Pengujian dokumen PDF Microsoft dengan tanda tangan digital beberapa *renderer* PDF

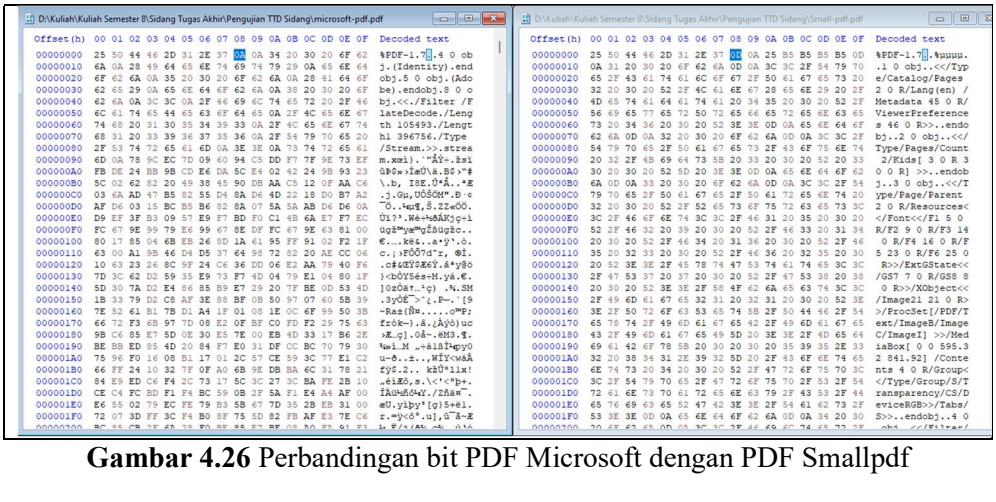
Gambar 4.23 Hasil verifikasi dokumen PDF Microsoft dengan tanda tangan digital PDF Microsoft



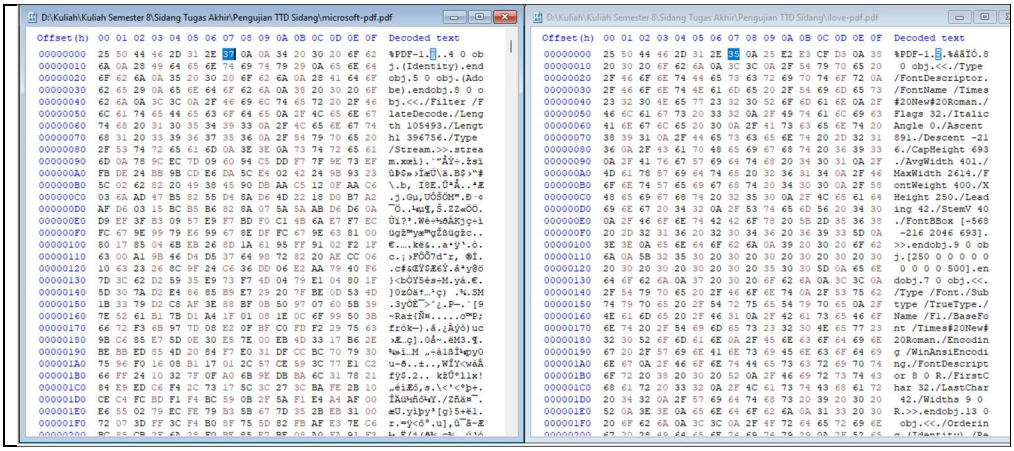
Gambar 4.24 Hasil verifikasi dokumen PDF Microsoft dengan tanda tangan digital *renderer* PDF lain



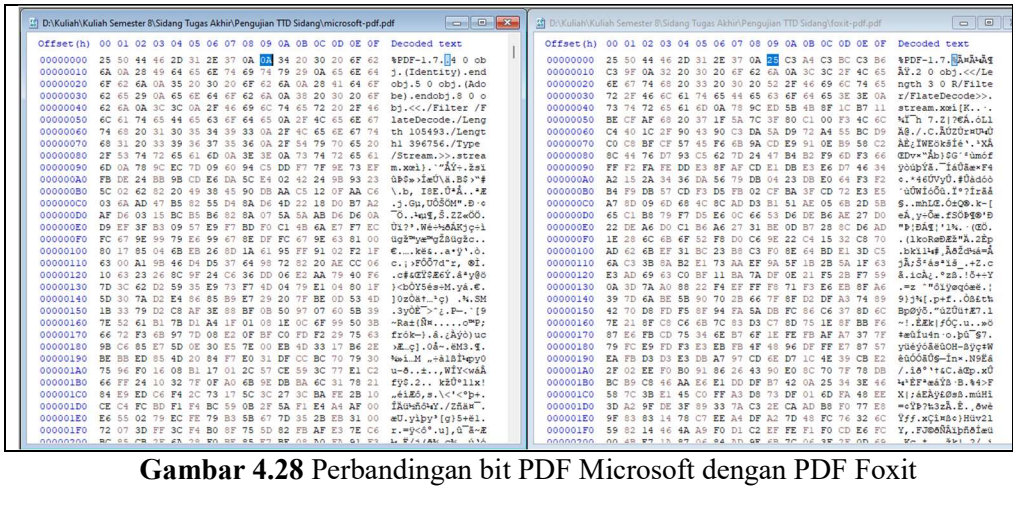
Gambar 4.25 Perbandingan bit PDF Microsoft dengan PDF Nitro



Gambar 4.26 Perbandingan bit PDF Microsoft dengan PDF Smallpdf



Gambar 4.27 Perbandingan bit PDF Microsoft dengan PDF Ilovepdf

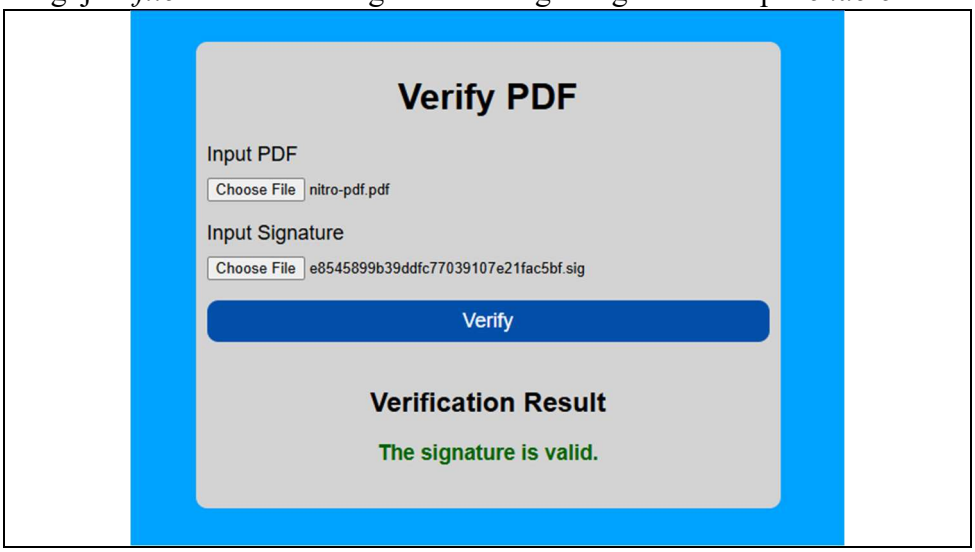


Gambar 4.28 Perbandingan bit PDF Microsoft dengan PDF Foxit

Hasil verifikasi dokumen PDF Microsoft menunjukan bahwa dokumen PDF Microsoft dinyatakan valid hanya ketika diverifikasi menggunakan tanda tangan digital dari PDF Microsoft itu sendiri. Meskipun dibuat menggunakan *file* DOCX yang sama, dokumen PDF Microsoft memiliki bit yang berbeda dengan dokumen PDF yang dibuat menggunakan *renderer* PDF lainnya.

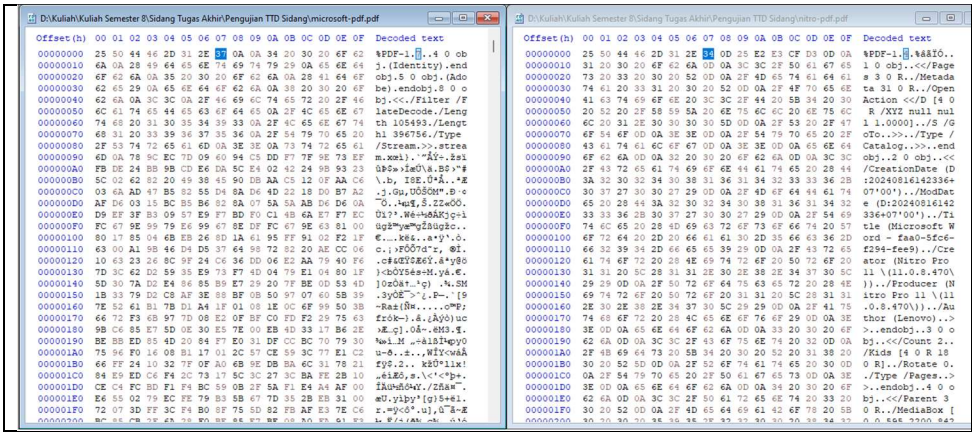
Hal tersebut dibuktikan dengan perbandingan bit yang ditunjukkan pada gambar 4.25 sampai 4.28. Dokumen PDF Microsoft memiliki perbedaan bit yang signifikan terhadap dokumen PDF yang dibuat dengan *renderer* PDF lain.

2) Pengujian *file* PDF Nitro dengan tanda tangan digital beberapa *renderer* PDF

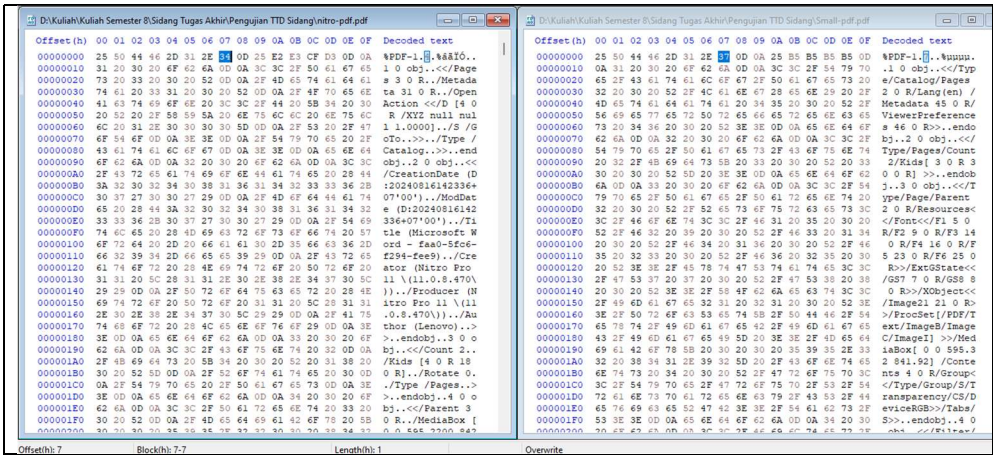


Gambar 4.29 Hasil verifikasi dokumen PDF Nitro dengan tanda tangan digital PDF Nitro

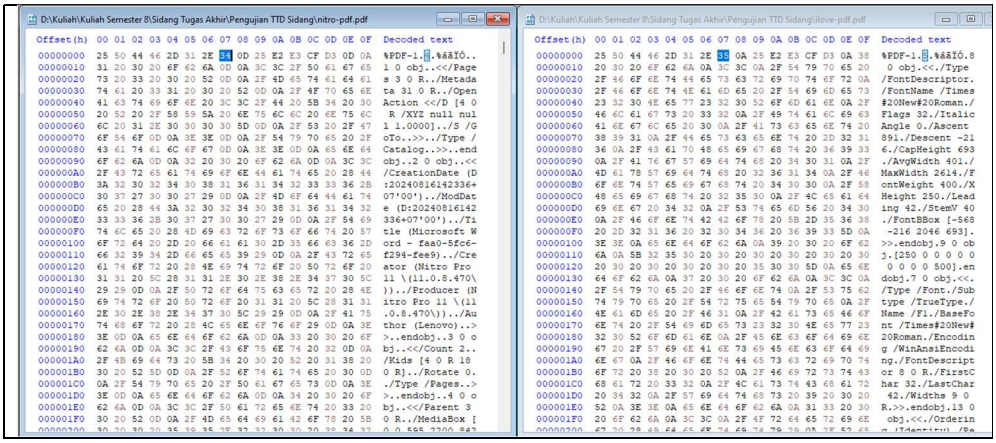
Gambar 4.30 Hasil verifikasi dokumen PDF Nitro dengan tanda tangan digital *renderer* PDF lain



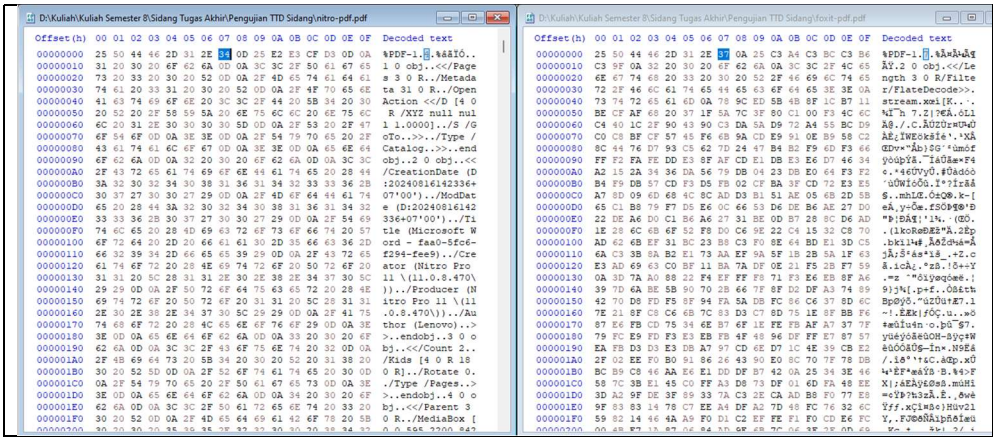
Gambar 4.31 Perbandingan bit PDF Nitro dengan PDF Microsoft



Gambar 4.32 Perbandingan bit PDF Nitro dengan PDF Smallpdf



Gambar 4.33 Perbandingan bit PDF Nitro dengan PDF Ilovepdf

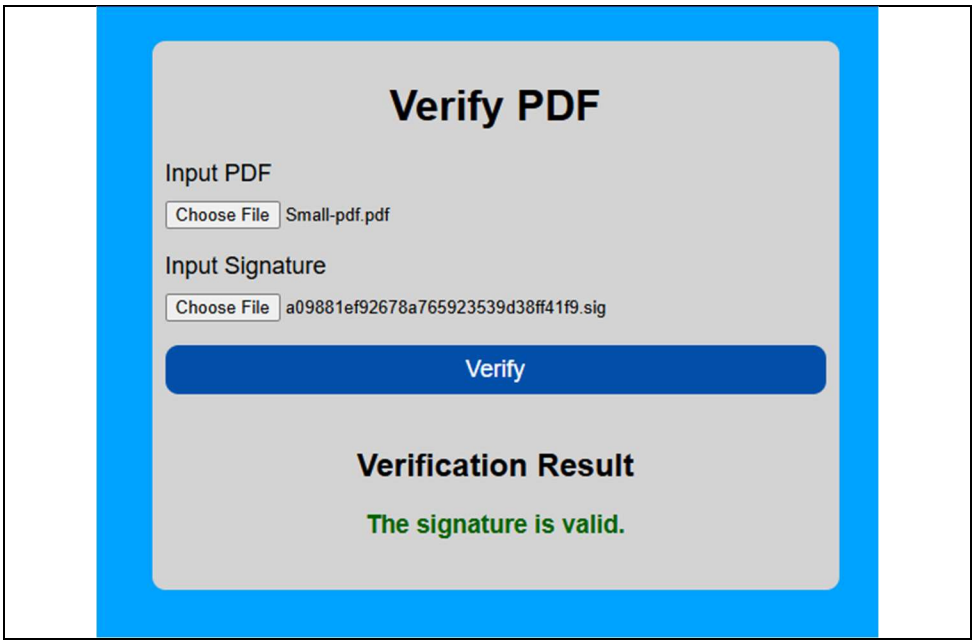


Gambar 4.34 Perbandingan bit PDF Nitro dengan PDF Foxit

Hasil verifikasi dokumen PDF Nitro menunjukan bahwa dokumen PDF Nitro dinyatakan valid hanya ketika diverifikasi menggunakan tanda tangan digital dari PDF Nitro itu sendiri. Meskipun dibuat menggunakan *file* DOCX yang sama, dokumen PDF Nitro memiliki bit yang berbeda dengan dokumen PDF yang dibuat menggunakan *renderer* PDF lainnya.

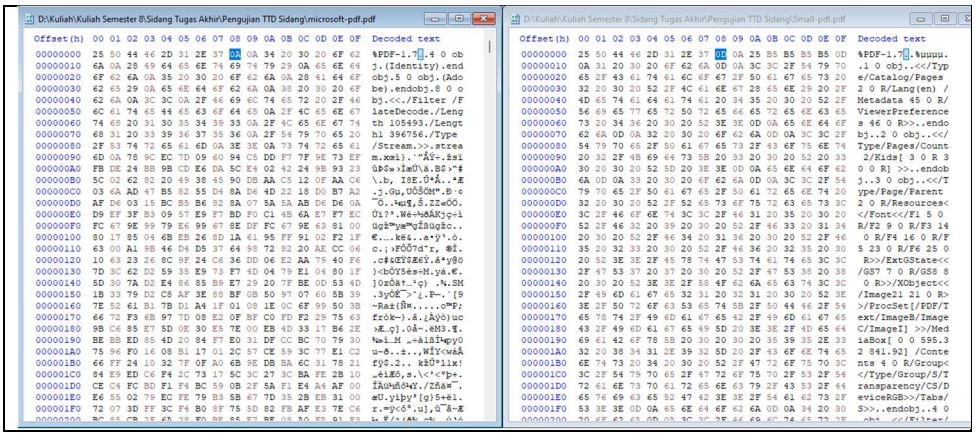
Hal tersebut dibuktikan dengan perbandingan bit yang ditunjukkan pada gambar 4.31 sampai 4.34. Dokumen PDF Nitro memiliki perbedaan bit yang signifikan terhadap dokumen PDF yang dibuat dengan *renderer* PDF lain.

3) Pengujian *file* PDF Smallpdf.com dengan tanda tangan digital beberapa *renderer* PDF

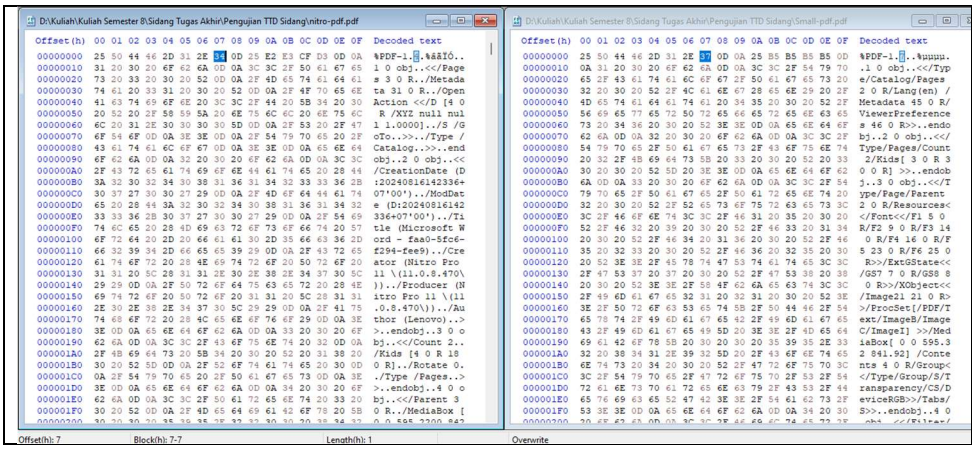


Gambar 4.35 Hasil verifikasi dokumen PDF Smallpdf dengan tanda tangan digital PDF Smallpdf

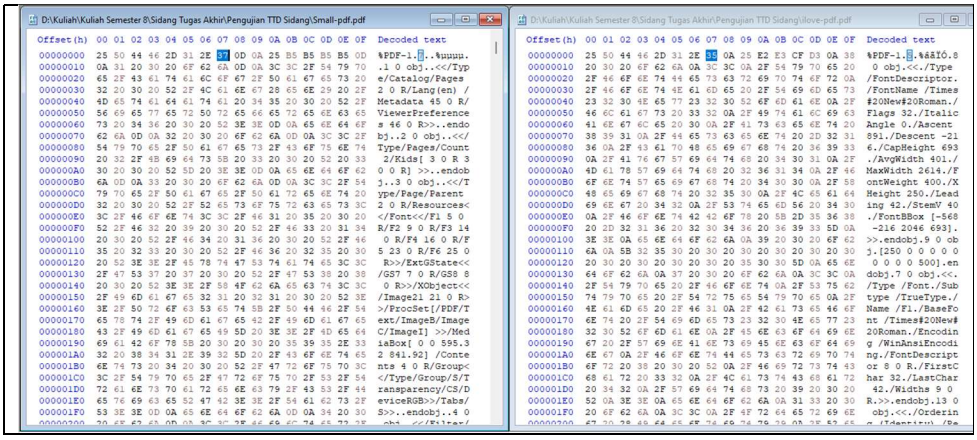
Gambar 4.36 Hasil verifikasi dokumen PDF Smallpdf dengan tanda tangan digital *renderer* PDF lain



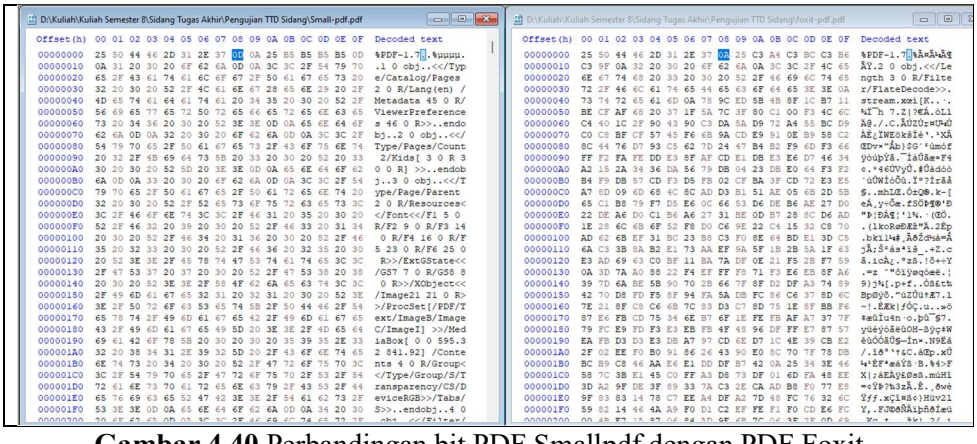
Gambar 4.37 Perbandingan bit PDF Smallpdf dengan PDF Microsoft



Gambar 4.38 Perbandingan bit PDF Smallpdf dengan PDF Nitro



Gambar 4.39 Perbandingan bit PDF Smallpdf dengan PDF Ilovepdf



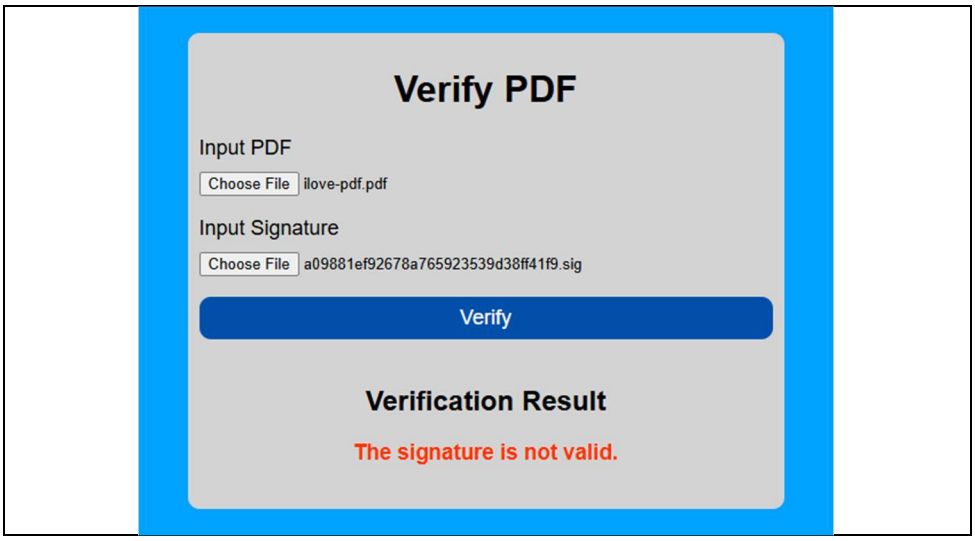
Gambar 4.40 Perbandingan bit PDF Smallpdf dengan PDF Foxit

Hasil verifikasi dokumen PDF Smallpdf menunjukan bahwa dokumen PDF Smallpdf dinyatakan valid hanya ketika diverifikasi menggunakan tanda tangan digital dari PDF Smallpdf itu sendiri. Meskipun dibuat menggunakan *file* DOCX yang sama, dokumen PDF Smallpdf memiliki bit yang berbeda dengan dokumen PDF yang dibuat menggunakan *renderer* PDF lainnya.

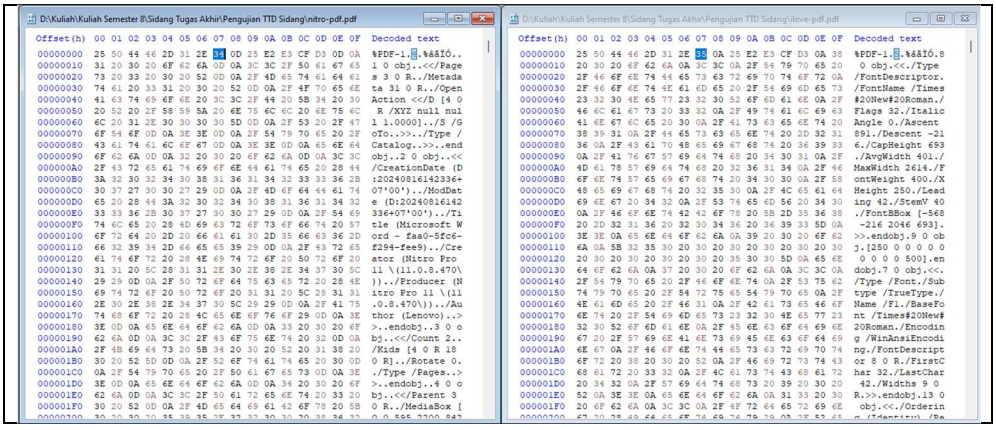
Hal tersebut dibuktikan dengan perbandingan bit yang ditunjukkan pada gambar 4.37 sampai 4.40. Dokumen PDF Smallpdf memiliki perbedaan bit yang signifikan terhadap dokumen PDF yang dibuat dengan *renderer* PDF lain.

4) Pengujian *file* PDF Ilovepdf.com dengan tanda tangan digital beberapa *renderer* PDF

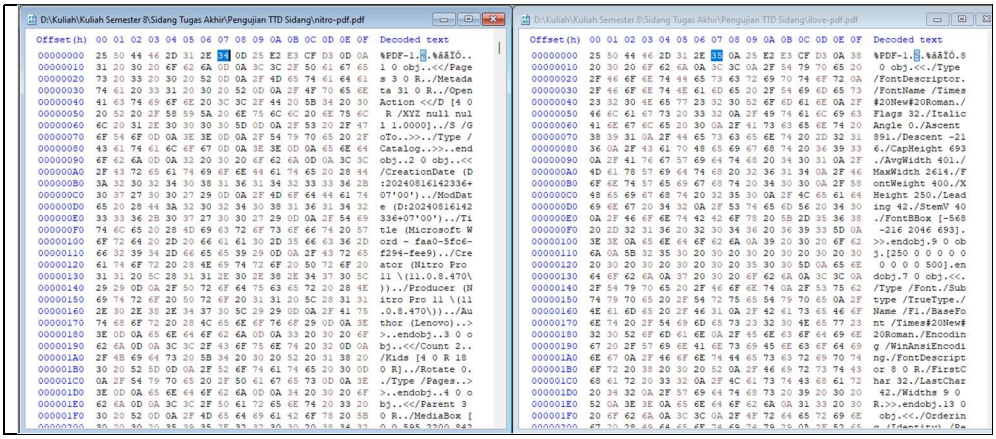
Gambar 4.41 Hasil verifikasi dokumen PDF Ilovepdf dengan tanda tangan digital PDF Ilovepdf



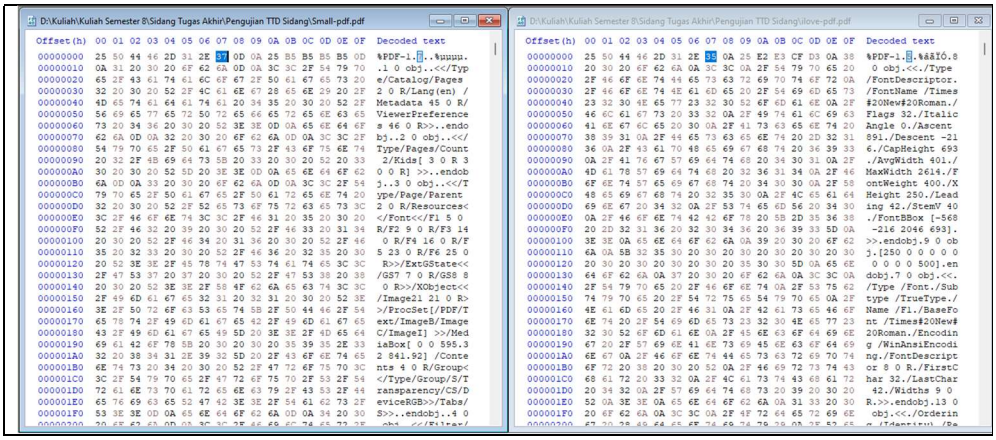
Gambar 4.42 Hasil verifikasi dokumen PDF Ilovepdf dengan tanda tangan digital *renderer* PDF lain



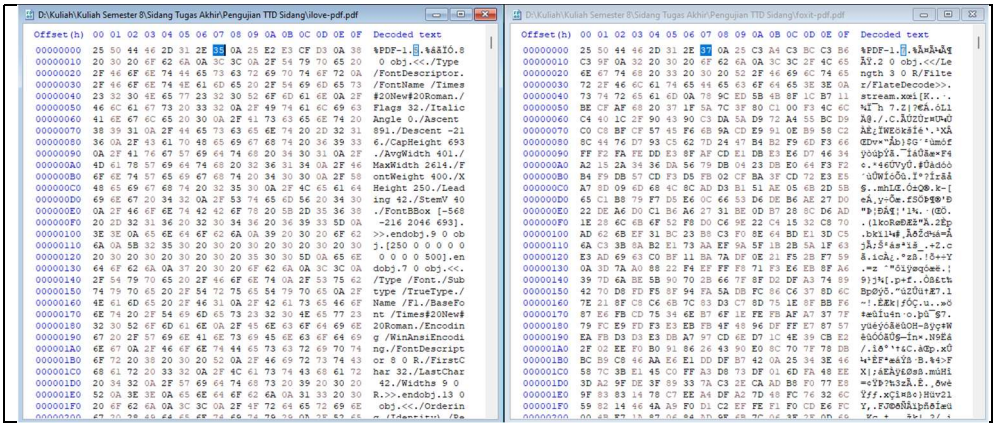
Gambar 4.43 Perbandingan bit PDF Ilovepdf dengan PDF Microsoft



Gambar 4.44 Perbandingan bit PDF Ilovepdf dengan PDF Nitro



Gambar 4.45 Perbandingan bit PDF Ilovepdf dengan PDF Smallpdf

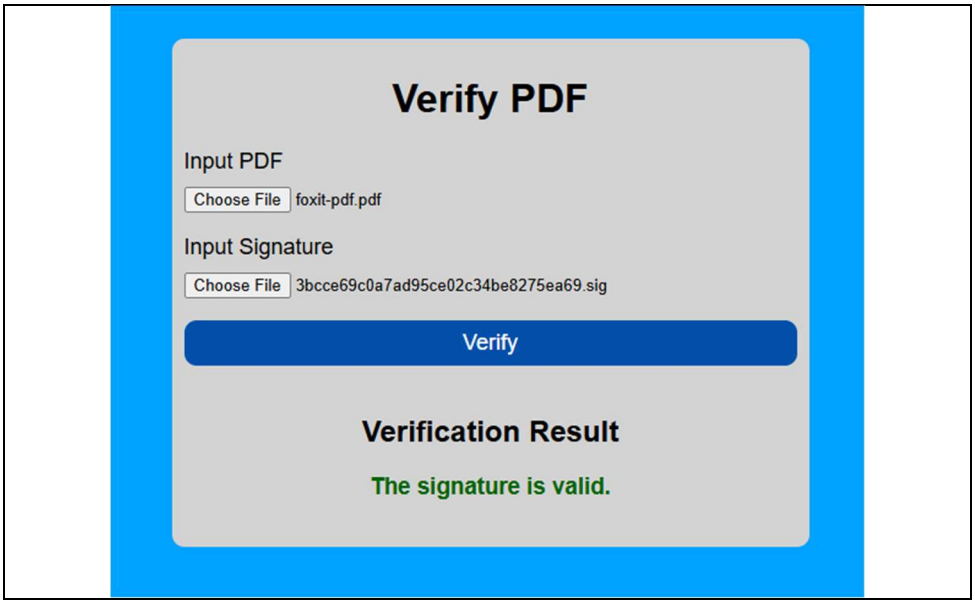


Gambar 4.46 Perbandingan bit PDF Ilovepdf dengan PDF Foxit

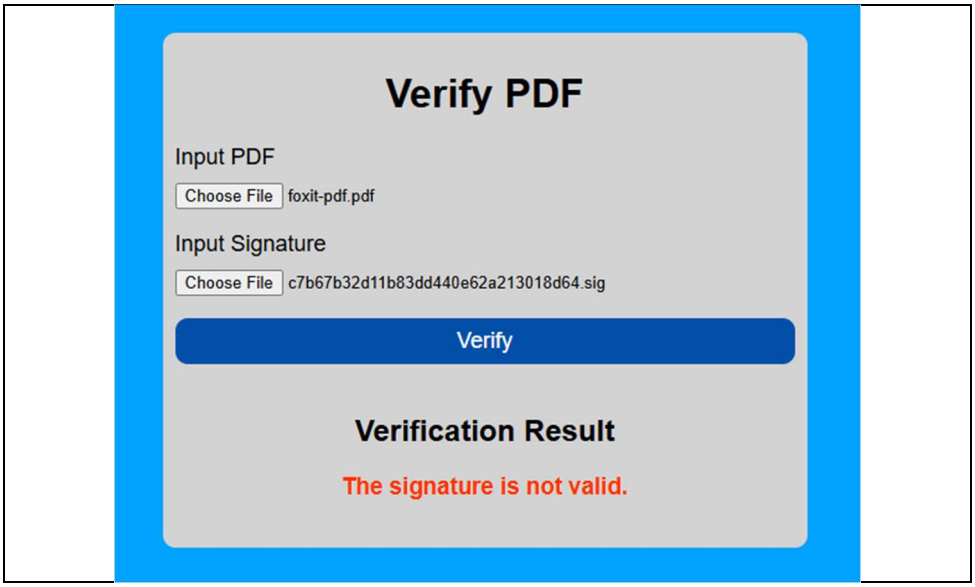
Hasil verifikasi dokumen PDF Ilovepdf menunjukan bahwa dokumen PDF Ilovepdf dinyatakan valid hanya ketika diverifikasi menggunakan tanda tangan digital dari PDF Ilovepdf itu sendiri. Meskipun dibuat menggunakan *file* DOCX yang sama, dokumen PDF Ilovepdf memiliki bit yang berbeda dengan dokumen PDF yang dibuat menggunakan *renderer* PDF lainnya.

Hal tersebut dibuktikan dengan perbandingan bit yang ditunjukkan pada gambar 4.42 sampai 4.46. Dokumen PDF Ilovepdf memiliki perbedaan bit yang signifikan terhadap dokumen PDF yang dibuat dengan *renderer* PDF lain.

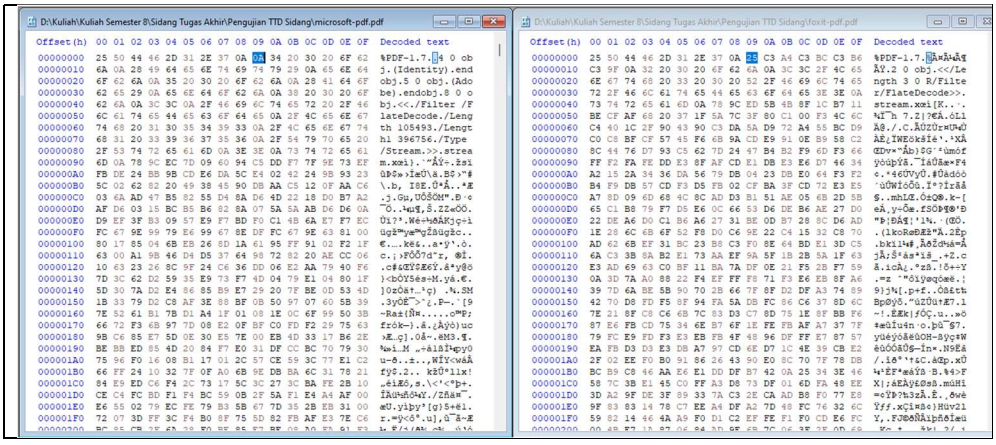
5) Pengujian *file* PDF Foxit.com dengan tanda tangan digital beberapa *renderer* PDF



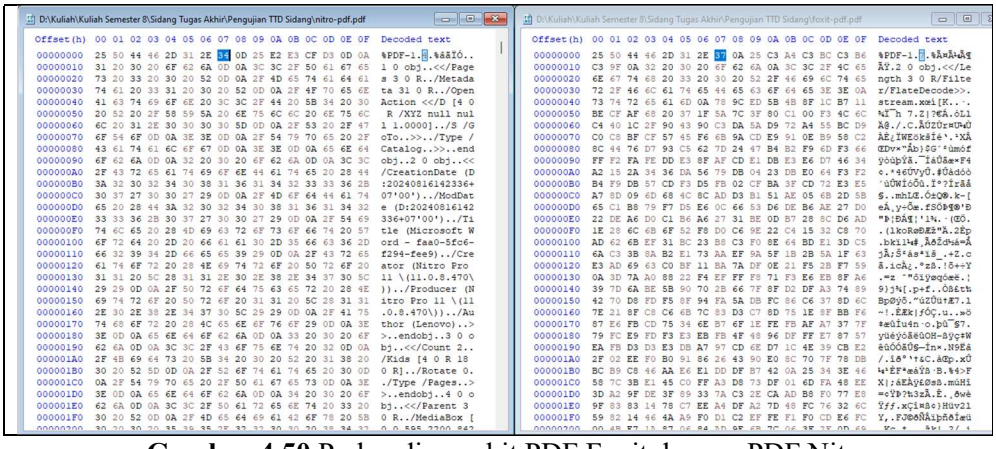
Gambar 4.47 Hasil verifikasi dokumen PDF Foxit dengan tanda tangan digital PDF Foxit



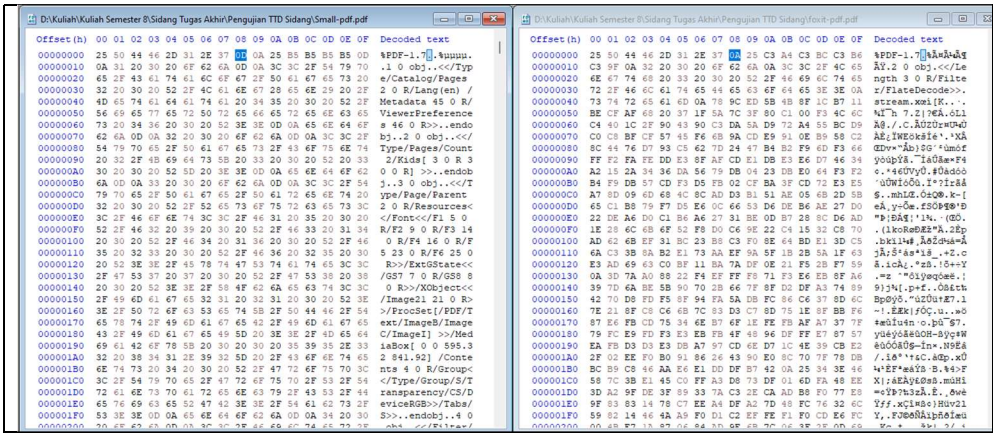
Gambar 4.48 Hasil verifikasi dokumen PDF Foxit dengan tanda tangan digital *renderer* PDF lain



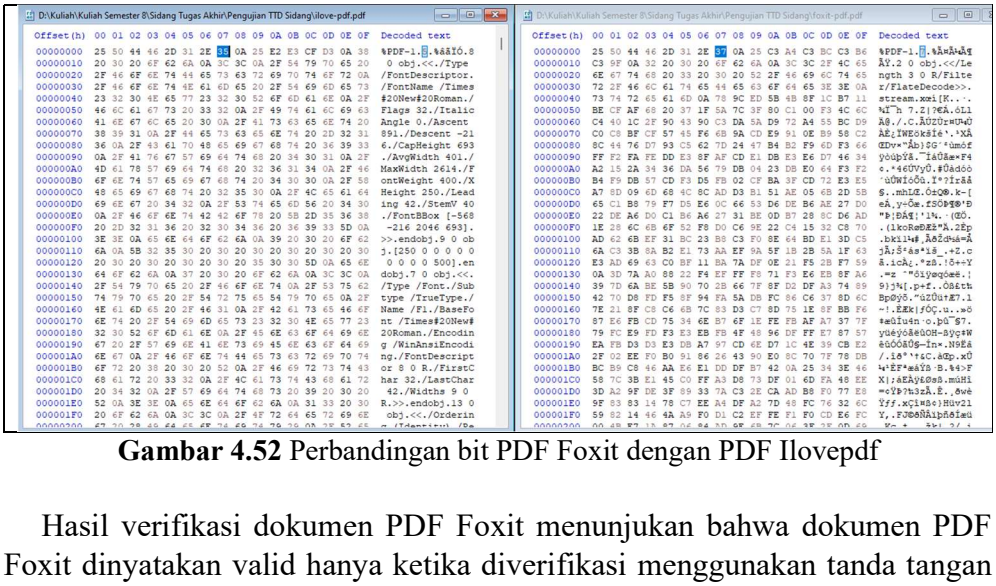
Gambar 4.49 Perbandingan bit PDF Foxit dengan PDF Microsoft



Gambar 4.50 Perbandingan bit PDF Foxit dengan PDF Nitro



Gambar 4.51 Perbandingan bit PDF Foxit dengan PDF Smallpdf



Gambar 4.52 Perbandingan bit PDF Foxit dengan PDF Ilovepdf

Hasil verifikasi dokumen PDF Foxit menunjukan bahwa dokumen PDF Foxit dinyatakan valid hanya ketika diverifikasi menggunakan tanda tangan digital dari PDF Foxit itu sendiri. Meskipun dibuat menggunakan *file* DOCX yang sama, dokumen PDF Foxit memiliki bit yang berbeda dengan dokumen PDF yang dibuat menggunakan *renderer* PDF lainnya.

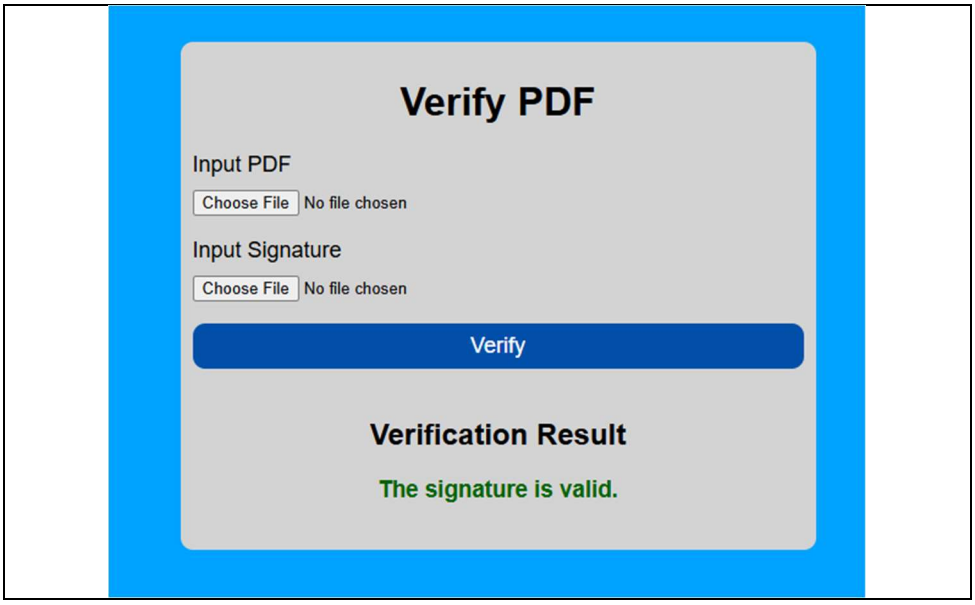
Hal tersebut dibuktikan dengan perbandingan bit yang ditunjukkan pada gambar 4.49 sampai 4.52. Dokumen PDF Foxit memiliki perbedaan bit yang signifikan terhadap dokumen PDF yang dibuat dengan *renderer* PDF lain.

Setelah melakukan berbagai skenario pengujian pembuatan PDF dari *File* DOCX yang sama dari beberapa *renderer* PDF, dibuatlah tabel ringkasan hasil pengujian sebagai berikut

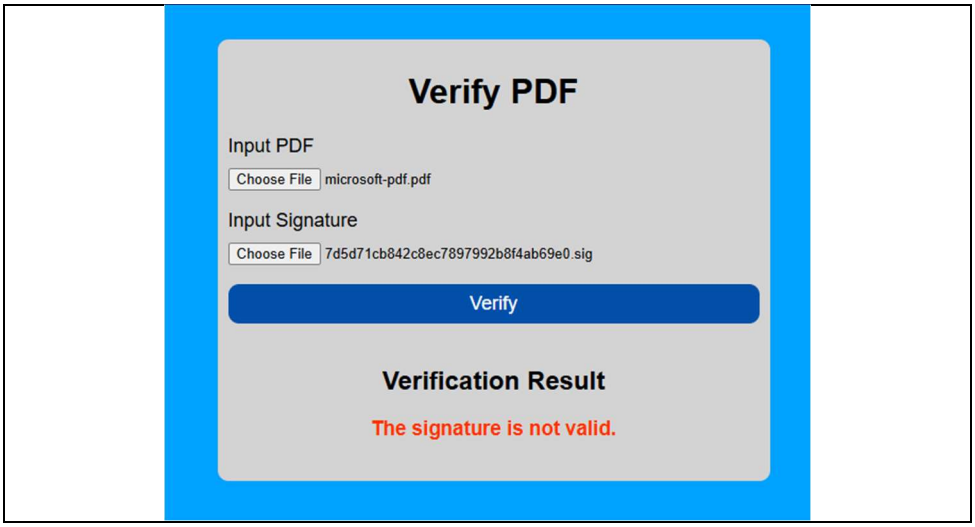
Tabel 4.2 Hasil pengujian perbandingan tanda tangan digital dokumen PDF dari *File* DOCX yang sama dari beberapa *renderer* PDF

Tanda tangan digital / *File* PDF	Microsoft Print to PDF	Nitro PDF	Smallpdf.com	Ilovepdf.com	Foxit.com
Microsoft Print to PDF	Valid	Tidak Valid	Tidak Valid	Tidak Valid	Tidak Valid
Nitro PDF	Tidak Valid	Valid	Tidak Valid	Tidak Valid	Tidak Valid
Smallpdf.com	Tidak Valid	Tidak Valid	Valid	Tidak Valid	Tidak Valid
Ilovepdf.com	Tidak Valid	Tidak Valid	Tidak Valid	Valid	Tidak Valid
Foxit.com	Tidak Valid	Tidak Valid	Tidak Valid	Tidak Valid	Valid

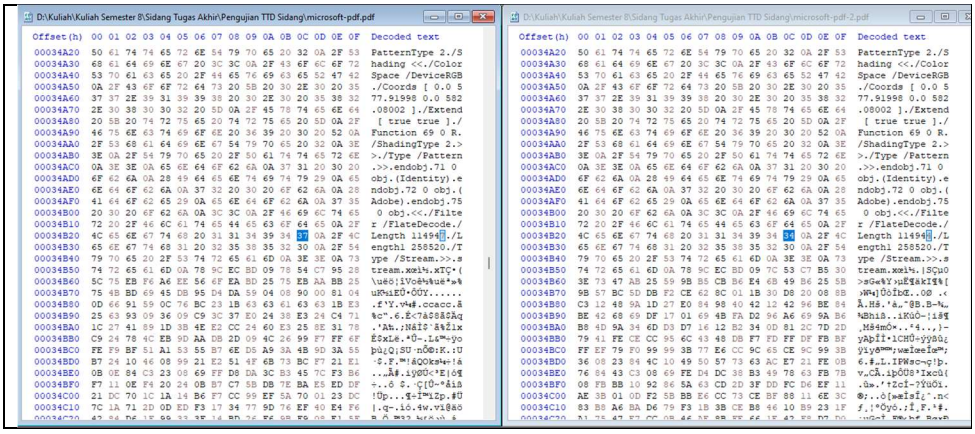
3. Pengujian Pembuatan Dokumen PDF dari *File* DOCX yang Sama dengan 3 Kali Iterasi dari Masing Masing *Renderer* PDF
   1) Microsoft Print to PDF



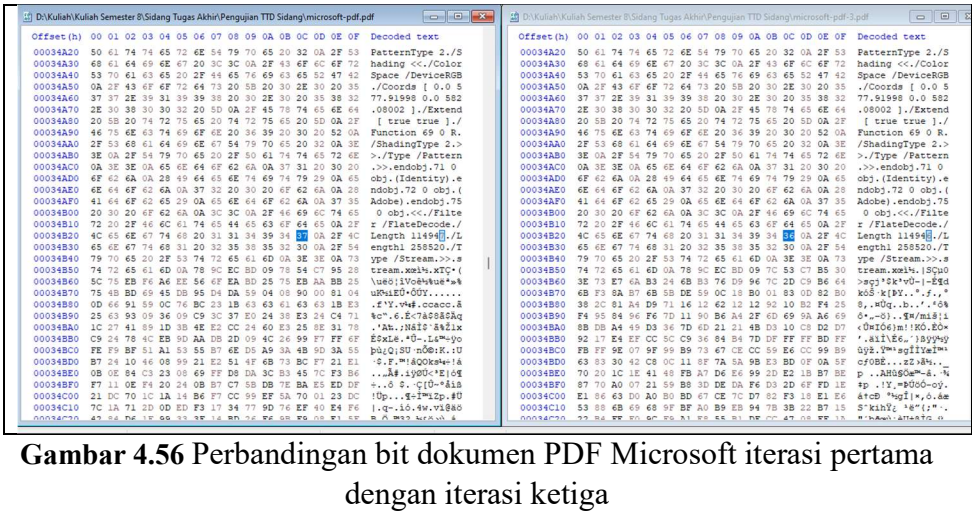
Gambar 4.53 Hasil verifikasi dokumen PDF iterasi pertama dengan tanda tangan digital iterasi pertama



Gambar 4.54 Hasil verifikasi dokumen PDF Microsoft iterasi pertama dengan tanda tangan digital iterasi berikutnya



Gambar 4.55 Perbandingan bit dokumen PDF Microsoft iterasi pertama dengan iterasi kedua

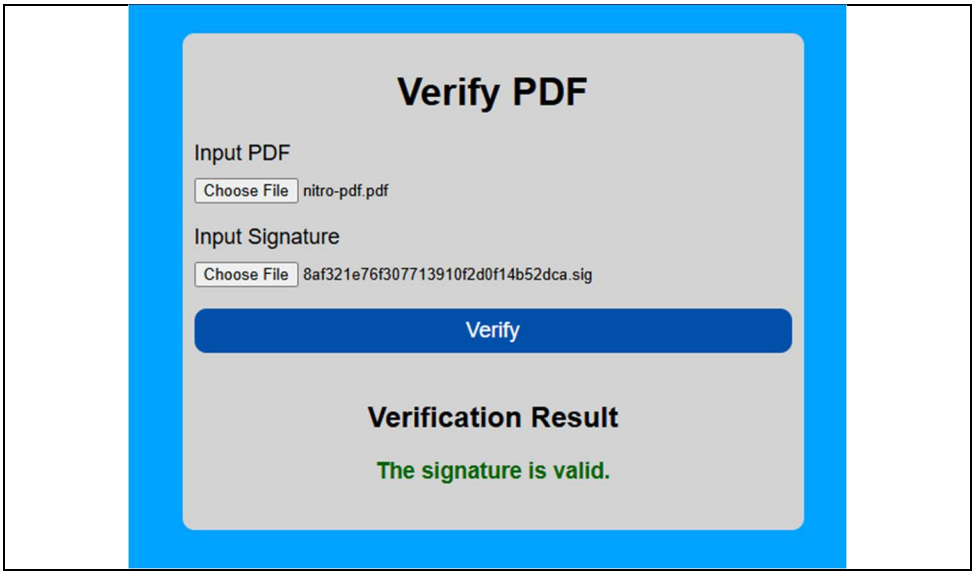


Gambar 4.56 Perbandingan bit dokumen PDF Microsoft iterasi pertama dengan iterasi ketiga

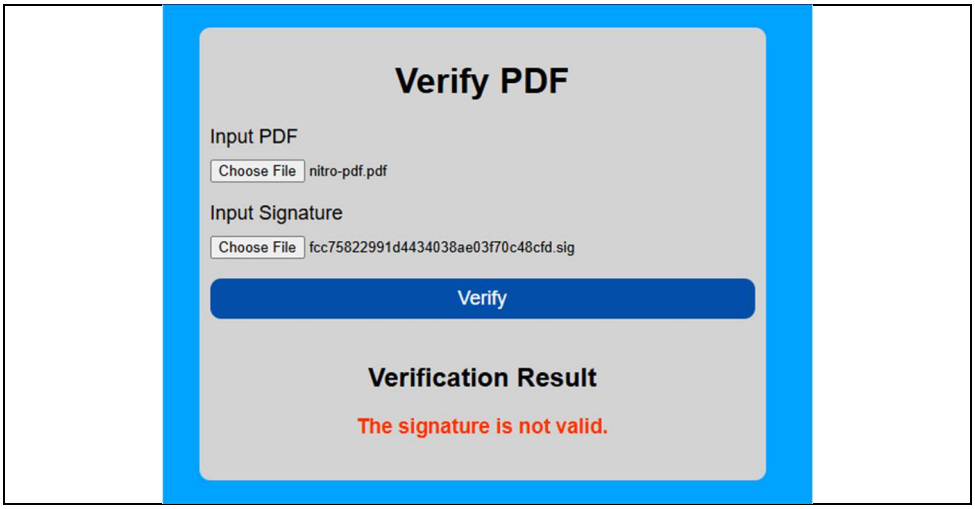
Hasil verifikasi dokumen PDF Microsoft untuk setiap iterasi menunjukan bahwa dokumen PDF Microsoft dinyatakan valid hanya ketika diverifikasi menggunakan tanda tangan digital iterasi pertama. Hal tersebut dibuktikan dengan perbandingan bit yang ditunjukkan pada gambar 4.55 dan 4.56.

Setiap iterasi pembuatan PDF Microsoft memiliki sedikit perbedaan ukuran bit *file* sekitar 16-32 bit. Secara keseluruhan, masing masing dokumen PDF tersebut memiliki perbedaan isi bit yang cukup signifikan.

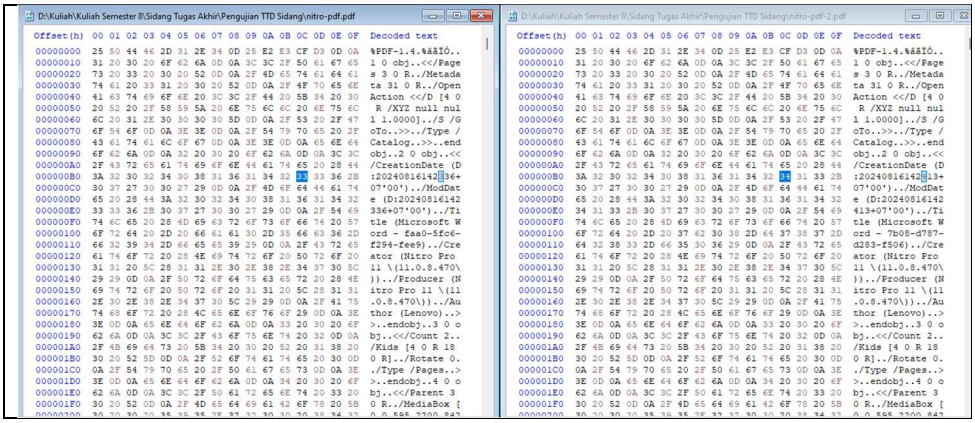
2) Nitro PDF



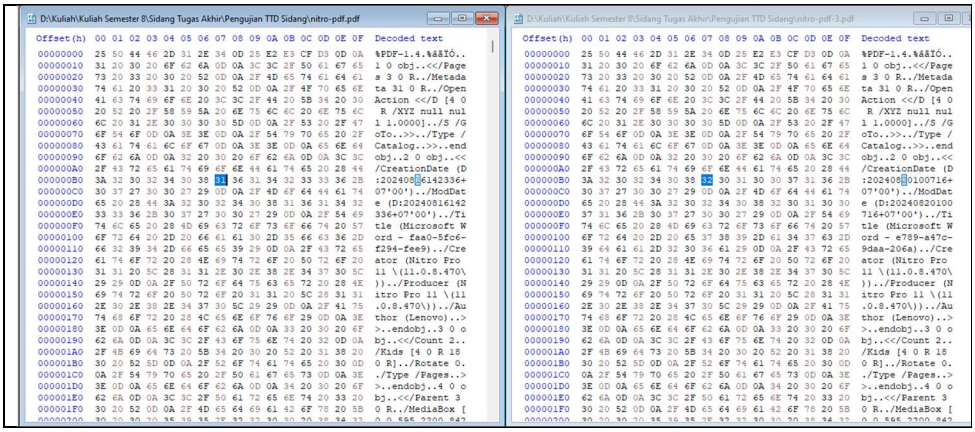
Gambar 4.57 Hasil verifikasi dokumen PDF Nitro iterasi pertama dengan tanda tangan digital iterasi pertama



Gambar 4.58 Hasil verifikasi dokumen PDF Nitro iterasi pertama dengan tanda tangan digital iterasi berikutnya



Gambar 4.59 Perbandingan bit dokumen PDF Nitro iterasi pertama dengan iterasi kedua

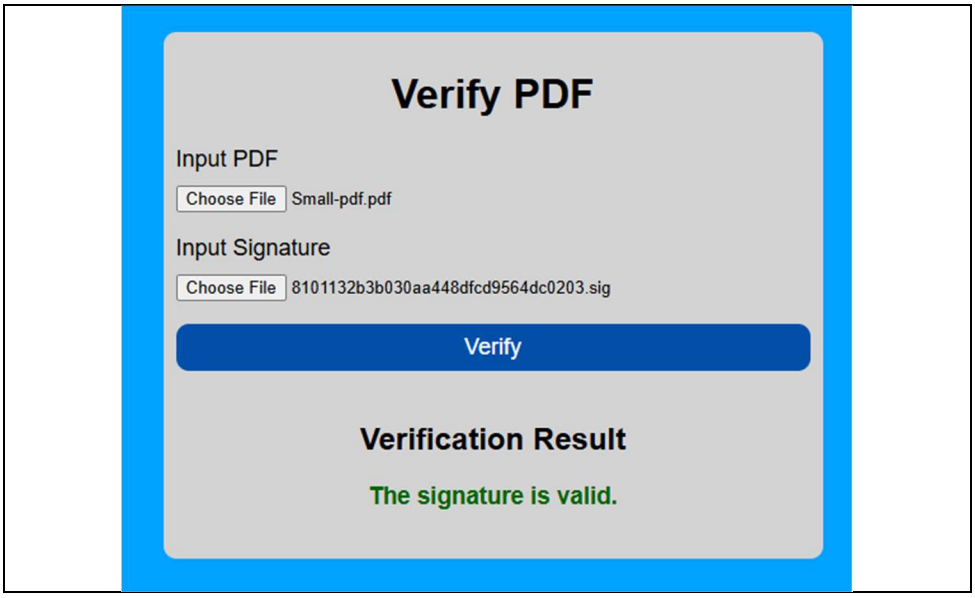


Gambar 4.60 Perbandingan bit dokumen PDF Nitro iterasi pertama dengan iterasi ketiga

Hasil verifikasi dokumen PDF Nitro untuk setiap iterasi menunjukan bahwa dokumen PDF Nitro dinyatakan valid hanya ketika diverifikasi menggunakan tanda tangan digital iterasi pertama. Hal tersebut dibuktikan dengan perbandingan bit yang ditunjukkan pada gambar 4.59 dan 4.60.

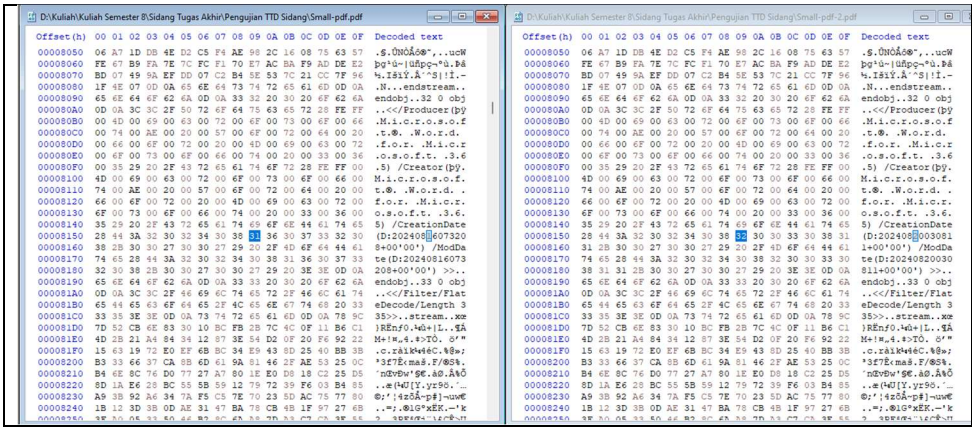
Setiap iterasi pembuatan PDF Nitro memiliki ukuran bit *file* yang sama persis. Secara keseluruhan, perbedaan yang dimiliki dari ketiga dokumen PDF tersebut terletak pada bagian metadata tanggal pembuatan dan *uuid* dokumen PDF.

3) Smallpdf.com

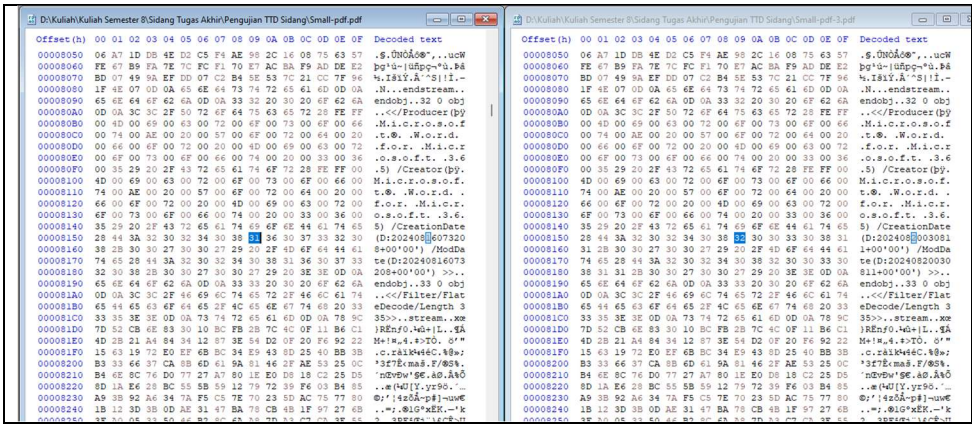


Gambar 4.61 Hasil verifikasi dokumen PDF Smallpdf iterasi pertama dengan tanda tangan digital iterasi pertama

Gambar 4.62 Hasil verifikasi dokumen PDF Smallpdf iterasi pertama dengan tanda tangan digital iterasi berikutnya



Gambar 4.63 Perbandingan bit dokumen PDF Smallpdf iterasi pertama dengan iterasi kedua

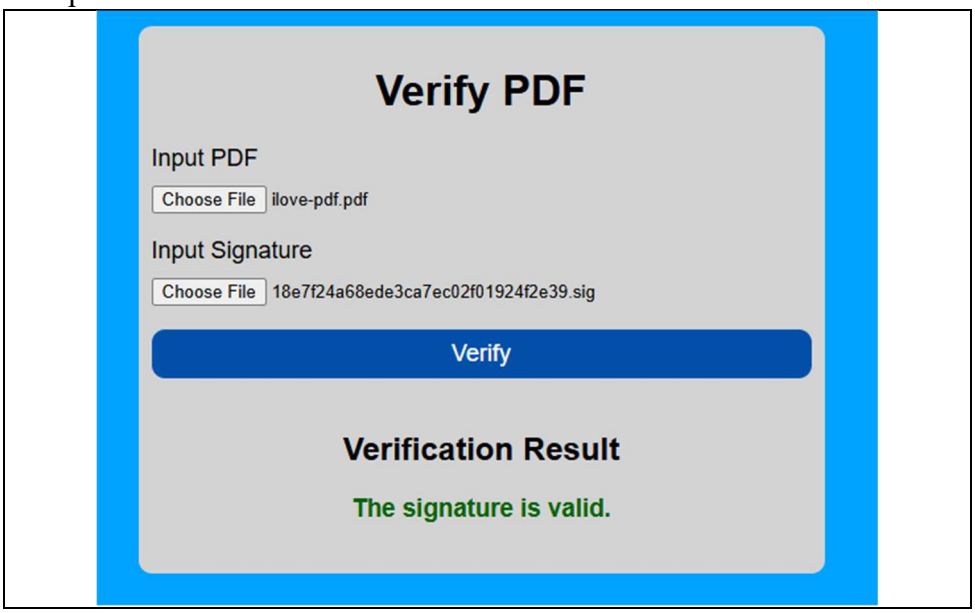


Gambar 4.64 Perbandingan bit dokumen PDF Smallpdf iterasi pertama dengan iterasi berikutnya

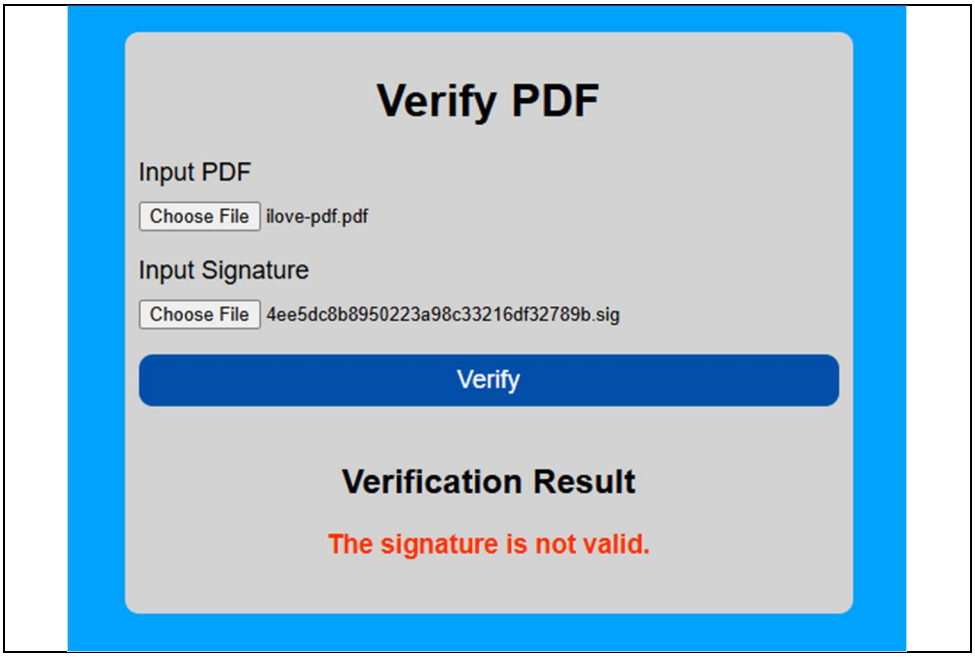
Hasil verifikasi dokumen PDF Smallpdf untuk setiap iterasi menunjukan bahwa dokumen PDF Smallpdf dinyatakan valid hanya ketika diverifikasi menggunakan tanda tangan digital iterasi pertama. Hal tersebut dibuktikan dengan perbandingan bit yang ditunjukkan pada gambar 4.63 dan 4.64.

Setiap iterasi pembuatan PDF Smallpdf memiliki ukuran bit *file* yang sama persis. Secara keseluruhan, perbedaan yang dimiliki dari ketiga dokumen PDF tersebut terletak pada bagian metadata tanggal pembuatan dan *uuid* dokumen PDF.

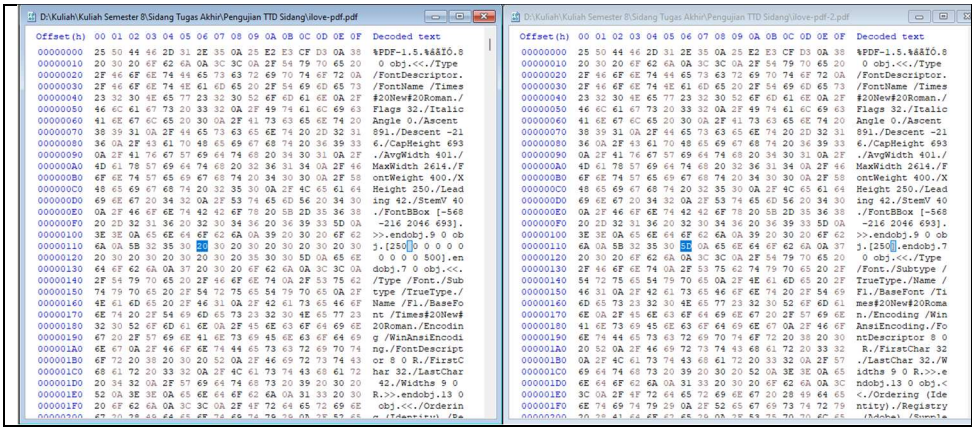
4) Ilovepdf.com



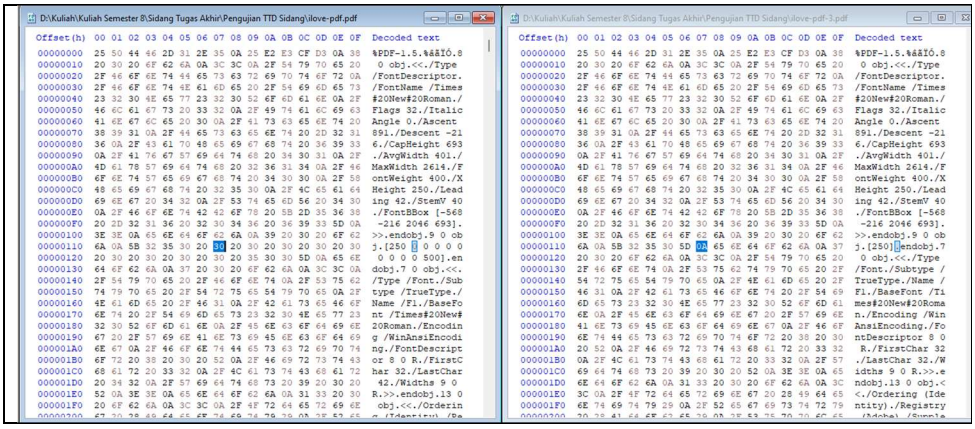
Gambar 4.65 Hasil verifikasi dokumen PDF Ilovepdf iterasi pertama dengan tanda tangan digital iterasi pertama



Gambar 4.66 Hasil verifikasi dokumen PDF Ilovepdf iterasi pertama dengan tanda tangan digital iterasi berikutnya



Gambar 4.67 Perbandingan bit dokumen PDF Ilovepdf iterasi pertama dengan iterasi kedua

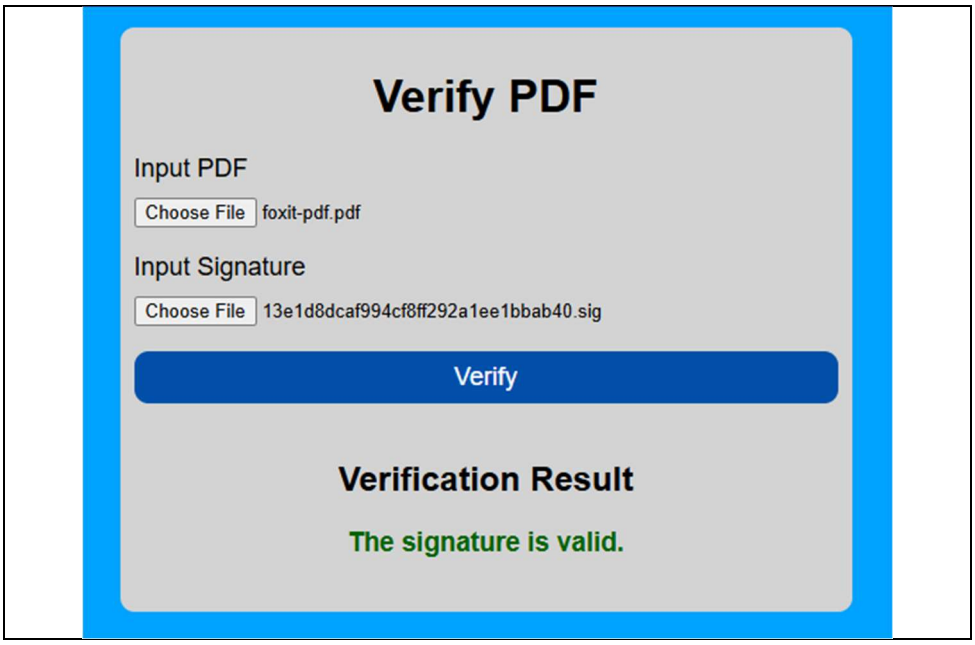


Gambar 4.68 Perbandingan bit dokumen PDF Ilovepdf iterasi pertama dengan iterasi ketiga

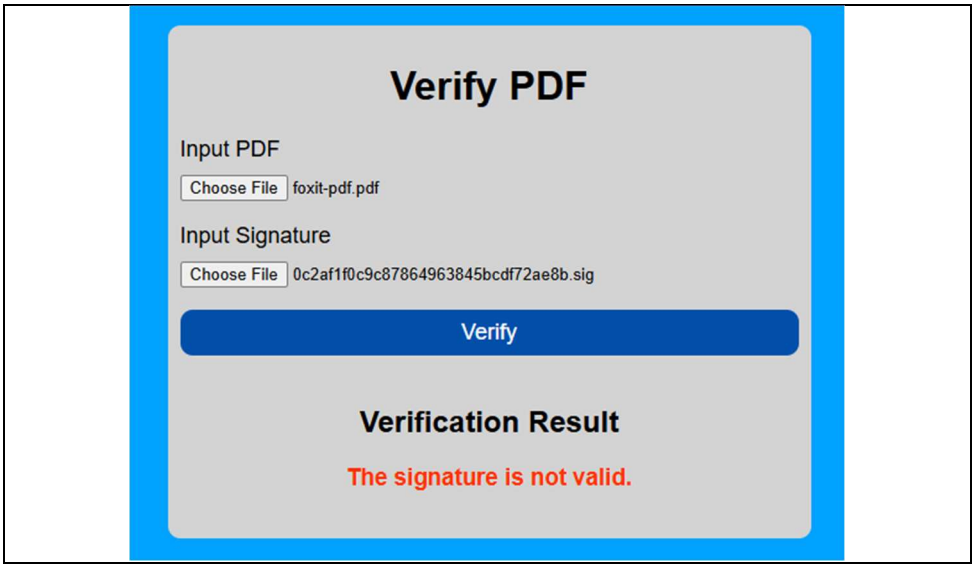
Hasil verifikasi dokumen PDF Ilovepdf untuk setiap iterasi menunjukan bahwa dokumen PDF Ilovepdf dinyatakan valid hanya ketika diverifikasi menggunakan tanda tangan digital iterasi pertama. Hal tersebut dibuktikan dengan perbandingan bit yang ditunjukkan pada gambar 4.67 dan 4.68.

Setiap iterasi pembuatan PDF Smallpdf memiliki ukuran bit *file* yang cukup berbeda. Secara keseluruhan, perbedaan isi bit dari ketiga dokumen PDF tersebut cukup signifikan.

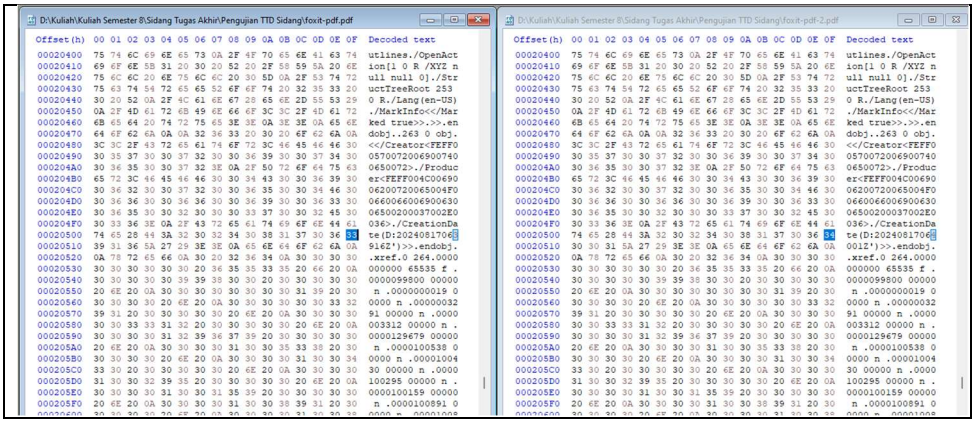
5) Foxit.com



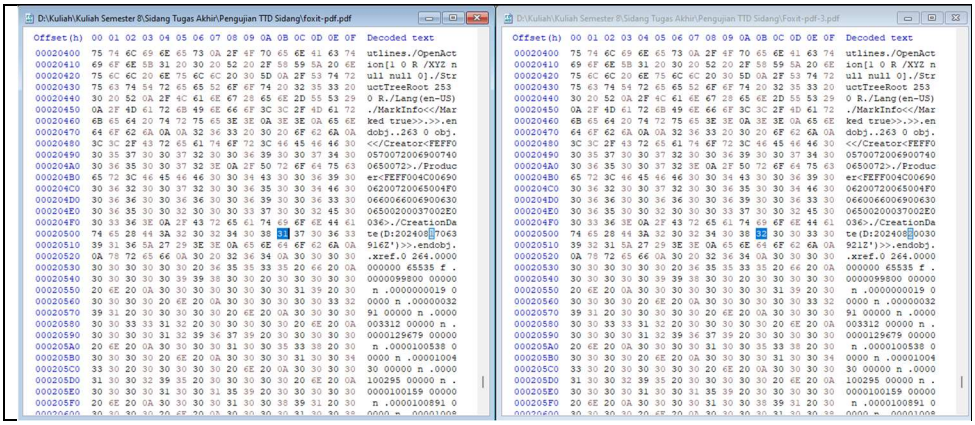
Gambar 4.69 Hasil verifikasi dokumen PDF Foxit iterasi pertama dengan tanda tangan digital iterasi pertama



Gambar 4.70 Hasil verifikasi dokumen PDF Foxit iterasi pertama dengan tanda tangan digital iterasi berikutnya



Gambar 4.71 Perbandingan bit dokumen PDF Foxit iterasi pertama dengan iterasi kedua



Gambar 4.72 Perbandingan bit dokumen PDF Foxit iterasi pertama dengan iterasi ketiga

Hasil verifikasi dokumen PDF Foxit untuk setiap iterasi menunjukan bahwa dokumen PDF Foxit dinyatakan valid hanya ketika diverifikasi menggunakan tanda tangan digital iterasi pertama. Hal tersebut dibuktikan dengan perbandingan bit yang ditunjukkan pada gambar 4.63 dan 4.64.

Setiap iterasi pembuatan PDF Foxit memiliki ukuran bit *file* yang sama persis. Secara keseluruhan, perbedaan yang dimiliki dari ketiga dokumen PDF tersebut terletak pada bagian metadata tanggal pembuatan dan *uuid* dokumen PDF.

Setelah melakukan berbagai skenario pengujian pembuatan PDF dari DOCX dengan 3 kali iterasi dari masing masing *renderer* PDF, dibuatlah tabel ringkasan hasil pengujian sebagai berikut

Tabel 4.3 Hasil pengujian tanda tangan digital dokumen PDF dari *file* DOCX yang sama dengan 3 kali iterasi dari masing masing *renderer* PDF

Iterasi	Microsoft Print to PDF	Nitro PDF	Smallpdf.com	Ilovepdf.com	Foxit.com
1	Valid	Valid	Valid	Valid	Valid
2	Tidak Valid	Tidak Valid	Tidak Valid	Tidak Valid	Tidak Valid
3	Tidak Valid	Tidak Valid	Tidak Valid	Tidak Valid	Tidak Valid

4.1.4. Pengujian Sistem Perangkat Lunak

Setelah melakukan pengujian tanda tangan digital, tahapan selanjutnya adalah melakukan pengujian sistem perangkat lunak. Tahapan ini dilakukan agar sistem yang dibangun berfungsi sesuai dengan spesifikasi awal yang telah ditentukan. Dalam penelitian ini, pengujian sistem dilakukan dengan mengukur fungsionalitas dari sistem yang dibuat. Detail hasil pengujian sistem adalah sebagai berikut

1) *Input file* bertipe .pdf pada kolom *input* .pdf

Skenario pengujian pertama yaitu pengguna melakukan *input file* bertipe pdf pada kolom *input* pdf.

Gambar 4.73 Hasil pengujian *input file* bertipe pdf

Pengguna berhasil melakukan *input file* bertipe PDF seperti yang ditampilkan pada gambar 4.73 diatas.

2) Melakukan *input file* selain bertipe .pdf pada kolom *input* .pdf

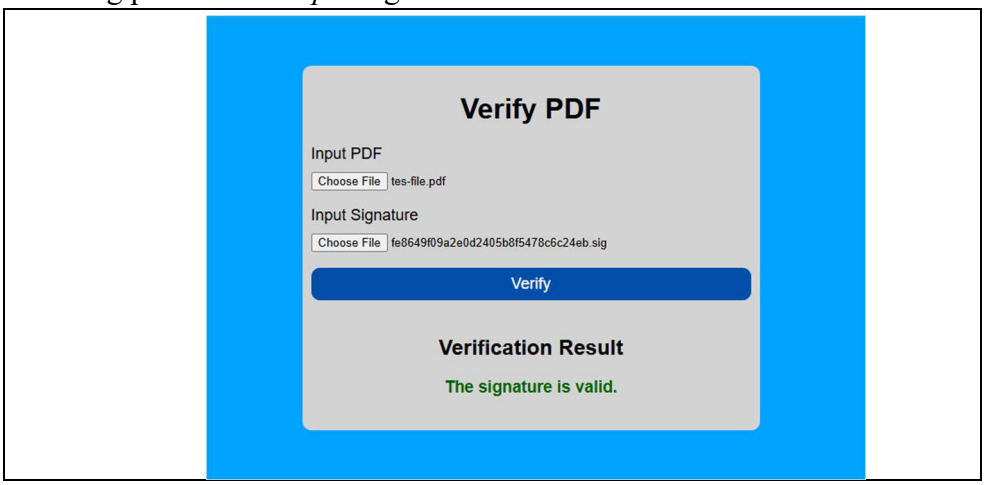
Skenario pengujian berikutnya yaitu pengguna melakukan *input file* bertipe selain pdf pada kolom *input* pdf.

Gambar 4.74 Hasil pengujian *input file* selain bertipe pdf

Pengguna gagal melakukan *input file* bertipe PDF seperti yang ditampilkan pada gambar 4.74 diatas.

3) *Input file* bertipe .sig pada kolom *input* .sig

Skenario pengujian berikutnya yaitu pengguna melakukan *input file* bertipe selain sig pada kolom *input* sig.



Gambar 4.75 Hasil pengujian *input file* bertipe sig

Pengguna berhasil melakukan *input file* bertipe sig seperti yang ditampilkan pada gambar 4.75 diatas.

4) *Input file* selain bertipe .sig pada kolom *input* .sig

Skenario pengujian berikutnya yaitu pengguna melakukan *input file* bertipe selain sig pada kolom *input* sig.

Gambar 4.76 Hasil pengujian *input file* selain bertipe sig

Pengguna gagal melakukan *input file* bertipe selain sig seperti yang ditampilkan pada gambar 4.76 diatas.

5) Menekan tombol *Sign* tanpa melakukan *input file*

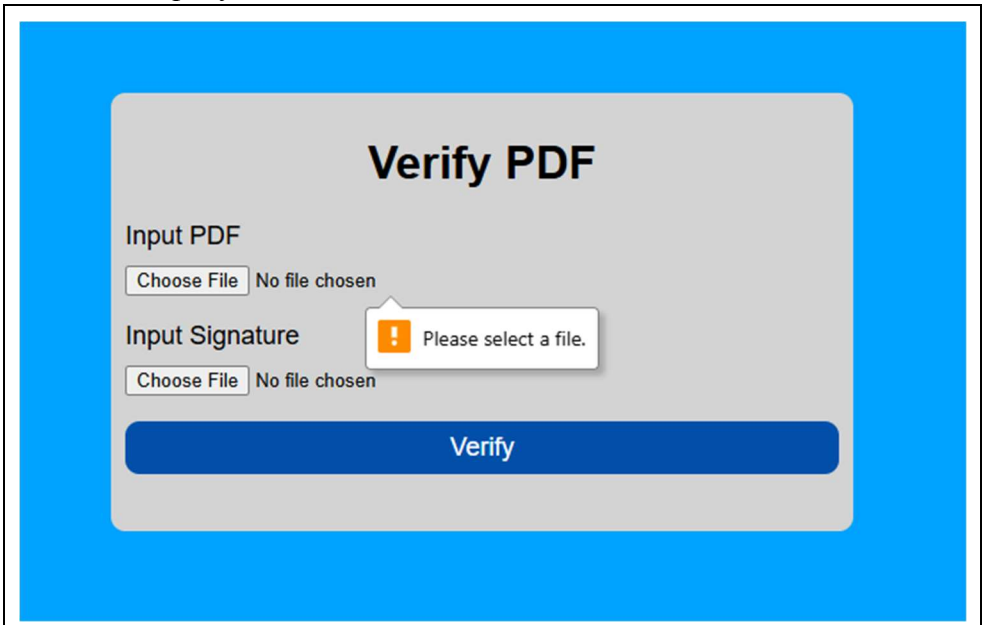
Skenario pengujian berikutnya yaitu pengguna menekan tombol *sign* tanpa melakukan *input file*.

Gambar 4.77 Hasil pengujian menekan tombol *sign* tanpa melakukan *input file*

Pengguna gagal melakukan *sign* tanpa meng*input file* seperti yang ditampilkan pada gambar 4.77 diatas.

6) Menekan tombol *Verify* tanpa melakukan *input file*

Skenario pengujian terakhir yaitu pengguna menekan tombol *verify* tanpa melakukan *input file*.



Gambar 4.78 Hasil pengujian menekan tombol *verify* tanpa melakukan *input file*

Pengguna gagal melakukan *verify* tanpa meng*input file* seperti yang ditampilkan pada gambar 4.78 diatas.

Setelah melakukan berbagai skenario pengujian pengujian diatas, dibuatlah tabel ringkasan hasil pengujian sebagai berikut

Tabel 4.4 Hasil pengujian sistem perangkat lunak

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil yang Didapatkan
1	*Input file* bertipe .pdf pada kolom *input* .pdf	Berhasil	Berhasil
2	*Input file* selain bertipe .pdf pada kolom *input* .pdf	Gagal	Gagal
3	*Input file* bertipe .sig pada kolom *input* .sig	Berhasil	Berhasil
4	*Input file* selain bertipe .sig pada kolom *input* .sig	Gagal	Gagal
5	Menekan tombol *Sign* tanpa melakukan *input file*	Gagal	Gagal
6	Menekan tombol *Verify* tanpa melakukan *input file*	Gagal	Gagal

4.2 Pembahasan

Pada bagian pembahasan ini akan dijelaskan hasil yang diperoleh untuk masalah penelitian yang diangkat. Masalah pada penelitian ini yaitu seberapa jauh kemampuan algoritma RSA dan SHA-256 dalam mendeteksi perbedaan pada dokumen PDF. Selain itu, penelitian ini juga menambahkan skenario skenario pengujian lanjutan yang belum dilakukan pada penelitian sebelumnya oleh (Andika et al., 2021) dan (Fachrul et al., 2022).

Penelitian ini menggunakan tanda tangan digital sebagai sebuah parameter pengecekan dalam mendeteksi perbedaan pada dokumen PDF. Tanda tangan digital dibuat dengan mengambil nilai *hash* SHA-256 dari dokumen PDF kemudian mengenkripsi nilai *hash* tersebut dengan kunci privat RSA. Kemudian tanda tangan digital akan diverifikasi dengan mendekripsinya menggunakan kunci publik RSA dan membandingkan nilai *hash* SHA-256 hasil dekripsi dengan dokumen PDF yang di*input*kan. Perubahan bit sedikit saja pada dokumen PDF akan mengakibatkan perubahan nilai *hash* SHA-256 secara drastis

Tanda tangan digital dibuat berdasarkan *input* dokumen PDF pada halaman *Sign* PDF. Tanda tangan digital yang telah dibuat akan disimpan kedalam *file* tersendiri dengan format sig. Kemudian *file* sig tersebut akan dicocokan dengan *input file* PDF pada halaman *verify* PDF. Jika cocok maka akan memberikan hasil tanda tangan valid, tetapi jika tidak cocok akan memberikan hasil tanda tangan tidak valid

Penelitian ini menguji validitas tanda tangan digital dengan 3 bentuk pengujian. Pengujian modifikasi dokumen PDF, pengujian pembuatan dokumen PDF dari *file* DOCX yang sama menggunakan beberapa *renderer* PDF, dan pengujian pembuatan PDF dari DOCX dengan 3 kali iterasi dari masing masing *renderer* PDF.

Pengujian modifikasi dokumen PDF menggunakan 7 skenario modifikasi dokumen PDF dengan menggunakan *tools* Nitro untuk modifikasinya. Hasil pengujian ini yaitu setiap modifikasi terhadap sebuah dokumen PDF akan menghasilkan perubahan bit dari dokumen PDF tersebut kecuali modifikasi nama *file* PDF.

Pengujian pembuatan dokumen PDF dari *file* DOCX yang sama menggunakan 5 *renderer* PDF yaitu Microsoft, Nitro, Smallpdf, Ilovepdf, dan Foxit. Hasil pengujian ini yaitu setiap *renderer* PDF memiliki ciri tersendiri dalam membuat dokumen PDF sehingga memiliki ukuran *file* dan isi bit yang berbeda-beda meskipun dibuat dari *file* DOCX yang sama.

Pengujian pembuatan PDF dari DOCX yang sama dengan 3 kali iterasi ini menggunakan 5 *renderer* PDF yaitu Microsoft, Nitro, Smallpdf, Ilovepdf, dan Foxit. Hasil pengujian ini yaitu setiap iterasi dari pembuatan dokumen PDF pasti menghasilkan isi bit yang berbeda meskipun memiliki ukuran *file* yang sama. Terdapat 3 *renderer* PDF yang menghasilkan ukuran *file* yang sama untuk setiap iterasinya yaitu Nitro, Smallpdf, dan Foxit meskipun isi bit masing masing dokumen PDF tersebut berbeda di bagian tanggal pembuatan dan *uuid* pada masing masing iterasi. Sementara itu, *renderer* PDF Microsoft dan Ilovepdf menghasilkan ukuran *file* dan isi bit yang berbeda untuk setiap iterasinya.

Selain itu, penelitian ini juga menguji sistem perangkat lunak yang telah dibuat. Pengujian sistem perangkat lunak ini dilakukan dengan 6 skenario pengujian. Berdasarkan pengujian tersebut, seluruh skenario pengujian memberikan hasil sesuai dengan yang diharapkan.

BAB V

PENUTUP

5.1. Kesimpulan

Berdasarkan penelitian yang telah dilakukan, didapatkan kesimpulan yang dapat dinyatakan dalam beberapa poin berikut ini:

1) Tanda tangan digital dapat mendeteksi perubahan dokumen PDF dengan mengecek kecocokan dari nilai *hash* SHA-256 dari dokumen PDF. Perubahan bit sekecil apapun pada dokumen PDF dapat mengakibatkan perubahan nilai *hash* SHA-256 secara drastis.
2) Berdasarkan hasil pengujian, algoritma *hash* SHA-256 dan algoritma RSA dapat mendeteksi hampir seluruh perbedaan dari dokumen PDF termasuk mendeteksi perbedaan dokumen PDF yang dibuat dari *file* DOCX yang sama tetapi menggunakan *renderer* PDF yang berbeda atau dibuat menggunakan *renderer* PDF yang sama tetapi pada waktu yang berbeda. Meskipun demikian, algoritma *hash* SHA-256 dan algoritma RSA tidak dapat mendeteksi perbedaan dari nama *file* PDF.
3) Algoritma *hash* SHA-256 dan algoritma RSA dirasa efektif dalam mendeteksi perbedaan dari dokumen PDF.

5.2. Saran

Berdasarkan penelitian yang telah dilakukan, terdapat beberapa hal yang dapat dikembangkan untuk penelitian selanjutnya. Adapun saran yang dapat diberikan adalah sebagai berikut:

1) Memasukkan tanda tangan digital kedalam *file* PDF.
2) Menguji perubahan dokumen PDF menggunakan aplikasi PDF editor lainnya

DAFTAR PUSTAKA

Andika, R. A., Putradana, A. G., & Pahlevi, R. R. (2021). Aplikasi Enterprise Document Digital Signature menggunakan RSA dan SHA256 untuk WFH di Era Pandemi COVID-19. *E-Proceeding of Engineering*, *8*(5), 10179–10186.

Anshori, Y., & Erwin Dodu, A. Y. (2019). Implementasi Algoritma Kriptografi Rivest Shamir Adleman (RSA) pada Tanda Tangan Digital Implementation of Rivest Shamir Adleman (RSA) Cryptography Algorithm On Digital Signatures. *Techno.Com Jurnal Teknologi Informasi*, *18*(2), 110–121.

Arisandi, D., Sukri, & Bahrudin Yusuf, M. (2020). PEMERIKSAAN INTEGRITAS DOKUMEN DENGAN DIGITAL SIGNATURE ALGORITHM. *JOISIE Journal Of Information System And Informatics Engineering*, *16*(1), 73–82. https://doi.org/10.55601/jsm.v16i1.180

Aryasanti, A., Hardjianto, M., Brotosaputro, G., & Roeswidiah, R. (2022). Implementasi Tanda Tangan Digital Menggunakan Algoritme RSA dan SHA-512 dengan Salt Berbasis Web. *Ticom: Technology of Information and Communication*, *10*(3), 181–186.

Cahyo Prabowo, E., & Afrianto, I. (2017). Penerapan Digital Signature Dan Kriptografi PadaOtentikasi Sertifikat Tanah Digital - Teknik Informatika Universitas Komputer Indonesia. *Jurnal Ilmiah Komputer Dan Informatika (KOMPUTA)*, *6*(2).

Dermawan Ikhsan, J. (2021). *Implementasi Algoritma RSA dan Hash SHA-256 untuk Tanda Tangan Digital dalam Membangkitkan Kode QR Akses Masuk Kampus*. 2–3.

Endelina. (2021). Implementasi Digital Signature Pada *File* Audio Menerapkan Metode SHA-256. *Journal of Informatics Management and Information Technology*, *1*(2), 60–67. http://www.hostjournals.com/jimat/article/view/98

Fachrul, M., Sutardi, S., Tajidun, L. M., & Aksara, L. B. (2022). Penerapan Konsep Digital Signature Terhadap Verifikasi Keaslian Dokumen Transkrip Nilai Mahasiswa Menggunakan Enkripsi Rivest Shamir Adleman. *SemanTIK*, *8*(1), 45. https://doi.org/10.55679/semantik.v8i1.21608

Hutagalung, J., Ramadhan, P. S., & Sihombing, S. J. (2023). Keamanan Data Menggunakan Secure *Hash*ing Algorithm (SHA)-256 dan Rivest Shamir Adleman (RSA) pada Digital Signature. *Jurnal Teknologi Informasi Dan Ilmu Komputer*, *10*(6), 1213–1222. https://doi.org/10.25126/jtiik.1067319

Nuraeni, F., Agustin, H., Muharam, I. M., Informatika, T., & Tasikmalaya, T. (2018). Implementasi Tanda Tangan Digital Menggunakan RSA dan SHA-512 Pada Proses Legalisasi Ijazah. *Konferensi Nasional Sistem Informasi (KNSI) 2018*, 864–869.

Putri, Y., & Manullang, E. (2018). Implementasi Kriptografi Digital Signature Menggunakan Secure *Hash* Algorithm (Sha-1). *Jurnal Teknologi Informasi* , *6*(1), 1–8.

Rangkuti, F. M., Budi Nugroho, N., & Panjaitan, Z. (2019). Implementasi Digital Signature

Pada E-Invoice Di Uniqa Digital Invitation Menggunakan Algoritma SHA-256 (Secure *Hash* Algorithm-256) Dan RSA (Rivest Shamir Adleman). *Jurnal CyberTech*, *x. No.x*(x). https://ojs.trigunadharma.ac.id/

Refialy, L., Sediyono, E., & Setiawan, A. (2015). Pengamanan Sertifikat Tanah Digital menggunakan Digital Signature SHA-512 dan RSA. *Jurnal Teknik Informatika Dan Sistem Informasi*, *1*(3), 229–234. https://doi.org/10.28932/jutisi.v1i3.400

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A Method for Obtaining Digital Signatures and Publik-Key Cryptosystems. *Communications of the ACM*, *21*(2), 120–126. https://doi.org/10.1145/359340.359342

Saputra, I., & Nasution, S. D. (2019). Analisa Algoritma SHA-256 Untuk Mendeteksi Orisinalitas Citra Digital. *Prosiding Seminar Nasional Riset Information Science (SENARIS)*, *1*(September), 164. https://doi.org/10.30645/senaris.v1i0.20

Saputra, I., & Nasution, S. D. (2022). Perbandingan Performa Algoritma Md5 Dan Sha-256 Dalam Membangkitkan Identitas *File*. *Jurnal Sains Komputer & Informatika (J-SAKTI)*, *6*(1), 172–187.

Zaatsiyah, N., & Djuniadi. (2021). *IMPLEMENTING DIGITAL SIGNATURE WITH RSA AND MD5 IN SECURING E-INVOICE DOCUMENT*. *5*, 129–140.

Andika, R. A., Putradana, A. G., & Pahlevi, R. R. (2021). Aplikasi Enterprise Document Digital Signature menggunakan RSA dan SHA256 untuk WFH di Era Pandemi COVID-19. *E-Proceeding of Engineering*, *8*(5), 10179–10186.

Anshori, Y., & Erwin Dodu, A. Y. (2019). Implementasi Algoritma Kriptografi Rivest Shamir Adleman (RSA) pada Tanda Tangan Digital Implementation of Rivest Shamir Adleman (RSA) Cryptography Algorithm On Digital Signatures. *Techno.Com Jurnal Teknologi Informasi*, *18*(2), 110–121.

Arisandi, D., Sukri, & Bahrudin Yusuf, M. (2020). PEMERIKSAAN INTEGRITAS DOKUMEN DENGAN DIGITAL SIGNATURE ALGORITHM. *JOISIE Journal Of Information System And Informatics Engineering*, *16*(1), 73–82. https://doi.org/10.55601/jsm.v16i1.180

Aryasanti, A., Hardjianto, M., Brotosaputro, G., & Roeswidiah, R. (2022). Implementasi Tanda Tangan Digital Menggunakan Algoritme RSA dan SHA-512 dengan Salt Berbasis Web. *Ticom: Technology of Information and Communication*, *10*(3), 181–186.

Cahyo Prabowo, E., & Afrianto, I. (2017). Penerapan Digital Signature Dan Kriptografi PadaOtentikasi Sertifikat Tanah Digital - Teknik Informatika Universitas Komputer Indonesia. *Jurnal Ilmiah Komputer Dan Informatika (KOMPUTA)*, *6*(2).

Dermawan Ikhsan, J. (2021). *Implementasi Algoritma RSA dan Hash SHA-256 untuk Tanda Tangan Digital dalam Membangkitkan Kode QR Akses Masuk Kampus*. 2–3.

Endelina. (2021). Implementasi Digital Signature Pada *File* Audio Menerapkan Metode SHA-256. *Journal of Informatics Management and Information Technology*, *1*(2), 60–67. http://www.hostjournals.com/jimat/article/view/98

Fachrul, M., Sutardi, S., Tajidun, L. M., & Aksara, L. B. (2022). Penerapan Konsep Digital Signature Terhadap Verifikasi Keaslian Dokumen Transkrip Nilai Mahasiswa Menggunakan Enkripsi Rivest Shamir Adleman. *SemanTIK*, *8*(1), 45. https://doi.org/10.55679/semantik.v8i1.21608

Hutagalung, J., Ramadhan, P. S., & Sihombing, S. J. (2023). Keamanan Data Menggunakan Secure *Hash*ing Algorithm (SHA)-256 dan Rivest Shamir Adleman (RSA) pada Digital Signature. *Jurnal Teknologi Informasi Dan Ilmu Komputer*, *10*(6), 1213–1222. https://doi.org/10.25126/jtiik.1067319

Nuraeni, F., Agustin, H., Muharam, I. M., Informatika, T., & Tasikmalaya, T. (2018). Implementasi Tanda Tangan Digital Menggunakan RSA dan SHA-512 Pada Proses Legalisasi Ijazah. *Konferensi Nasional Sistem Informasi (KNSI) 2018*, 864–869.

Putri, Y., & Manullang, E. (2018). Implementasi Kriptografi Digital Signature Menggunakan Secure *Hash* Algorithm (Sha-1). *Jurnal Teknologi Informasi* , *6*(1), 1–8.

Rangkuti, F. M., Budi Nugroho, N., & Panjaitan, Z. (2019). Implementasi Digital Signature Pada E-Invoice Di Uniqa Digital Invitation Menggunakan Algoritma SHA-256 (Secure *Hash* Algorithm-256) Dan RSA (Rivest Shamir Adleman). *Jurnal CyberTech*, *x. No.x*(x). https://ojs.trigunadharma.ac.id/

Refialy, L., Sediyono, E., & Setiawan, A. (2015). Pengamanan Sertifikat Tanah Digital menggunakan Digital Signature SHA-512 dan RSA. *Jurnal Teknik Informatika Dan Sistem Informasi*, *1*(3), 229–234. https://doi.org/10.28932/jutisi.v1i3.400

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A Method for Obtaining Digital Signatures and Publik-Key Cryptosystems. *Communications of the ACM*, *21*(2), 120–126. https://doi.org/10.1145/359340.359342

Saputra, I., & Nasution, S. D. (2019). Analisa Algoritma SHA-256 Untuk Mendeteksi Orisinalitas Citra Digital. *Prosiding Seminar Nasional Riset Information Science (SENARIS)*, *1*(September), 164. https://doi.org/10.30645/senaris.v1i0.20

Saputra, I., & Nasution, S. D. (2022). Perbandingan Performa Algoritma Md5 Dan Sha-256 Dalam Membangkitkan Identitas *File*. *Jurnal Sains Komputer & Informatika (J-SAKTI)*, *6*(1), 172–187.

Zaatsiyah, N., & Djuniadi. (2021). *IMPLEMENTING DIGITAL SIGNATURE WITH RSA AND MD5 IN SECURING E-INVOICE DOCUMENT*. *5*, 129–140.